University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Albert Gutowski

Regular Path Queries and Modal Constraints

Decidability of Query Containment under a Graph Database Schema

Doctoral dissertation

Supervisor

dr hab. Filip Murlak
Institute of Informatics
University of Warsaw

Warsaw, 2025

Abstract

This dissertation solves the problem of query containment under constraints for unions of conjunctive regular path queries (UCRPQs) and constraints expressed using an extension of the description logic $\mathcal{ALC}$. The logic $\mathcal{ALC}$ can be seen as a notational variant of (multi-)modal logic; we consider the logic $\mathcal{ALCO}_{\text{reg}}$, extending $\mathcal{ALC}$ with constants and regular expressions.

This theoretical research lies on the border of database theory – graph databases in particular – and the field of knowledge representation and reasoning. We study the problem under the standard set semantics, in the natural – and more challenging – setting of only finite databases. The obtained result significantly narrows the decidability gap for this problem, as it appears to be the first decidability result for the full class of conjunctive regular path queries in this context, and there is a known undecidability result by Rudolph (2016) concerning the same query language and another extension of the description logic $\mathcal{ALC}$, known as $\mathcal{ALCOIF}$.

The core idea of the proof of the main result is based on several papers published by the author; the techniques have been heavily reworked, unified, and generalized for this dissertation. In the most technical parts of the thesis we work in the setting of directed graphs, using bisimilarity types of trees (partial unravellings) to represent modal constraints, and using homomorphism types of graphs to reason about queries. This is non-standard, as the problem is usually studied either in the setting of databases (in database theory) or in the setting of interpretations (in knowledge representation and reasoning); we describe in detail how to reduce the problem from the latter setting to ours.

One of the main constructions in the proof is based on a technique for solving the *finite query entailment problem* in the presence of transitivity by Gogacz, Ibañez-García, and Murlak (2018), which can be seen as constructing a tree automaton recognizing decompositions of graphs into strongly connected components; the main challenge is to represent the components abstractly, to keep the alphabet of the automaton finite, while being able to distribute information about constraints and queries over these components. Some other constructions presented in this thesis bear strong similarities to several other techniques existing in the literature; most notably, a little-known construction on deterministic finite automata (Hesse 2003; Bojańczyk 2009), and the *finite chase* procedure by Rosati (2006).

The focus of the dissertation is describing the developed techniques in an accessible way. The solution is organized as a chain of reductions to simpler problems. A crucial part of the result is describing the setting of *ranked graphs*, showing that the main problem can be reduced to reasoning in this setting, and generalizing the techniques to this setting. However, this is done only in Chapter 8; nearly half of this dissertation is devoted to introducing the core techniques in simpler settings, to make them easier to understand. When moving to increasingly difficult settings, we make an effort to discuss why simpler approaches fail, and what the intuitions behind their more advanced variants are.

We strive to provide full, formal proofs. The advantage of this approach is that it forces us to develop notions that are suitable not just for high-level descriptions, but also for low-level arguments. A notable downside is a large number of new definitions, which is amplified by the fact that we work in the setting of graphs, non-standard for this problem. To circumvent this, a lot of effort has been put into keeping the definitions as simple as possible; the author believes that the introduced concepts are, in the vast majority, natural and well-suited to the considered problems.

Streszczenie

Niniejsza rozprawa rozwiązuje problem zawierania zapytań w obecności więzów, dla zapytań UCRPQ (ang. *unions of conjunctive regular path queries*) i więzów wyrażonych przy użyciu rozszerzenia logiki deskrypcyjnej $\mathcal{ALC}$, która może być postrzegana jako wariant notacyjny logiki (multi)modalnej; główny wynik pracy dotyczy logiki $\mathcal{ALCO}_{\text{reg}}$, rozszerzającej $\mathcal{ALC}$ o stałe i wyrażenia regularne.

Przedstawione badania teoretyczne leżą na pograniczu teorii baz danych – w szczególności grafowych baz danych – oraz reprezentacji wiedzy i wnioskowania. Problem jest rozważany w standardowej semantyce zbiorowej, z naturalnym – i bardziej wymagającym – założeniem, że rozważane bazy danych są skończone. Uzyskany wynik znacząco zawęża lukę pomiędzy rozstrzygalnością a nierozstrzygalnością dla tego problemu: jest to, jak się zdaje, pierwszy pozytywny wynik obejmujący pełną klasę zapytań CRPQ w tym kontekście, a istnieje znany wynik nierozstrzygalności (Rudolph 2016) dotyczący tego samego języka zapytań i innego rozszerzenia logiki $\mathcal{ALC}$, znanego jako $\mathcal{ALCOIF}$.

Kluczowe idee dowodu bazują na moich publikacjach; używane techniki zostały gruntownie przeprojektowane, ujednolicone i uogólnione na potrzeby tej rozprawy. W najbardziej technicznych częściach rozprawy pracuję z grafami skierowanymi, używając typów bisymilarności drzew (częściowych rozwinięć grafu) do reprezentacji więzów modalnych, oraz typów homomorfizmu grafów do rozumowania o zapytaniach. Jest to niestandardowe: rozważany problem jest zwykle badany albo w kontekście baz danych (w teorii baz danych), albo w kontekście interpretacji (w reprezentacji wiedzy i wnioskowaniu); szczegółowo opisuję redukcję problemu do rozważanego wariantu.

Jedna z głównych konstrukcji w dowodzie opiera się na metodzie stworzonej do rozwiązania *problemu wynikania zapytań w modelach skończonych* w obecności przechodnich relacji (Gogacz, Ibañez-García, and Murlak 2018). Metoda ta może być postrzegana jako konstrukcja automatu na drzewach, rozpoznającego dekompozycje grafów na silnie spójne składowe; głównym wyzwaniem jest reprezentacja tych składowych w abstrakcyjny sposób, tak żeby alfabet automatu był skończony, ale umożliwiał rozłożenie informacji o więzach i zapytaniach pomiędzy składowymi. Wprowadzam też kilka innych konstrukcji podobnych do technik występujących w literaturze; w szczególności, do mało znanej konstrukcji na automatach skończonych (Hesse 2003; Bojańczyk 2009), i do procedury *finite chase* (Rosati 2006).

Moim priorytetem było przedstawienie opracowanych metod w przystępny sposób. Rozwiązanie problemu opisuję jako łańcuch redukcji do coraz prostszych problemów. Kluczową częścią wyniku są *grafy z rangami*: ich definicja, uogólnienie używanych konstrukcji w celu uwzględnienia rang, i redukcja głównego problemu do kontekstu grafów z rangami. Definiuję je jednak dopiero w rozdziale 8; niemal połowę rozprawy poświęcam wprowadzeniu głównych technik w prostszych kontekstach, żeby ułatwić ich zrozumienie. Przy przechodzeniu do trudniejszych kontekstów, wyjaśniam dlaczego wcześniejsze podejścia zawodzą, i jakie intuicje stoją za bardziej zaawansowanymi wariantami używanych technik.

Staram się przedstawiać pełne, formalne dowody. Zaletą tego podejścia jest to, że wymusza definiowanie pojęć odpowiednich nie tylko do wysokopoziomowego opisu rozwiązania, ale także do niskopoziomowych dowodów. Istotną wadą jest duża liczba nowych definicji, co jest potęgowane przez pracę w kontekście grafów, niestandardowym dla tego problemu. Aby temu zaradzić, włożyłem dużo wysiłku w opracowanie jak najprostszych definicji; uważam, że wprowadzone przeze mnie pojęcia są w zdecydowanej większości naturalne i dobrze dopasowane do rozważanych problemów.

Contents

Chapter 1

Introduction

One of the most fundamental problems in theoretical computer science is the *satisfiability problem*, which asks if there exists a structure in which a given formula is satisfied. This problem for propositional logic, known as *SAT*, is one of the most famous examples of NP-complete problems, with multiple practical applications through the use of *SAT solvers*. For first-order logic, the satisfiability problem is undecidable, which motivated a search for formalisms that admit decidability: syntactic fragments of classical logics, such as the *two-variable fragment* and the *guarded fragment* of first-order logic, as well as non-classical logics, such as *modal logic* and related *description logics*. Following the identification of several fundamental formalisms for which the problem is decidable, multiple extensions of these formalisms were studied, to understand the limits of decidability. One of the approaches was to extend the basic formalisms with natural features, such as counting, constants, transitivity, and regular expressions; this approach, on the one hand, makes it possible to identify very expressive formalisms that are decidable, and on the other hand, provides high-level insights into which combinations of features lead to undecidability and should be avoided in practical applications.

As mentioned above, SAT solvers are a prominent example of practical applications of the satisfiability problem for propositional logic, but some real-life tasks require reasoning in more expressive logics. Such tasks can be encountered, for example, in software/hardware verification, database query optimization, or knowledge representation and reasoning. Contrary to pure mathematics, in computer science the focus is typically on finite structures, and on the corresponding *finite* satisfiability problem. In databases and in knowledge representation and reasoning, one often works with a combination of two formalisms: one for expressing *queries*, and another for expressing a *database schema* or an *ontology*. There is an ongoing effort to understand the limits of decidability of the finite satisfiability problem for such combinations. This thesis contributes to these efforts; the main result is formulated as a decidability result for *query containment under constraints* in graph databases, where queries are expressed as unions of *conjunctive regular path queries*, and constraints are expressed using description logics.

The main contribution is a novel technique for working with regular path queries that combines a little-known construction on deterministic automata with several existing strategies. This seems to be the first technique handling arbitrary conjunctive regular path queries in this context.

The rest of this introduction is structured as follows. We begin with a gentle introduction to the topic of this thesis. Next, we discuss the state of research on this topic. Finally, we discuss the main contributions of this thesis. At the end, we describe the content of each chapter.

1.1 Setting the scene

Rigorous logical reasoning about the real world typically involves creating a mathematical model of a fragment of the world, formalizing concepts and relations, making some assumptions, and drawing conclusions. To explain the subject of this thesis, we present an example of logical reasoning that leads to an unlikely conclusion. Then, we discuss which parts of this process we are interested in, how we represent the assumptions, and what kind of questions we want to answer.

1.1.1 Automated reasoning

Consider the following assumptions.

1. There exists at least one human.
2. Every human has a biological mother, who is also a human.
3. The number of humans who have ever lived is finite.

By a simple argument, we can conclude that there is a human who is their own ancestor[1]. A slightly informal argument can be described as follows. Consider an arbitrary human (who exists – by the first assumption), their mother, maternal grandmother, great-grandmother, and so on; all of them exist and are humans (by the second assumption). This way, we can obtain an arbitrarily long chain of ancestors; because the number of humans is finite (by the third assumption), some human appears more than once in such a chain, which implies that they are their own ancestor.

The conclusion that someone is their own ancestor seems unlikely, so it is natural to suspect that either some assumption is false, or the presented reasoning is incorrect. We leave evaluating the truthfulness of the assumptions to others – in the case of these particular assumptions, to philosophers, biologists, and cosmologists. In this thesis, we are only interested in the reasoning itself, and in the formal language used to express the assumptions and the conclusion.

To minimize the possibility that the reasoning is incorrect, we formulate rigorous mathematical proofs, using well-established notions and theorems; for example, the argument presented earlier can be written as follows. Let N be the number of humans who have ever lived, and let h_0 be an arbitrary human. Let $(h_0, h_1, \ldots, h_N)$ be a sequence of humans such that h_i is a mother of h_{i-1} for each $i = 1, 2, \ldots, N$. Because this sequence has length $N + 1$ and there are only N humans, by the pigeonhole principle, there are some numbers $i \neq j$ such that $h_i = h_j$, which means that h_i is their own ancestor.

In computer science, we want to write useful automated tools – computer programs; for example, we could write a program that, given a set of assumptions and a supposed conclusion expressed in a particular formal language, decides if the conclusion can be reached from the assumptions. In *theoretical* computer science, we do not care if these tools are actually useful, as long as the solved problem is interesting – to the point where we often do not bother to implement them, and instead we

[1] By *ancestors* we mean parents, grandparents, great-grandparents, and so on.

just describe how they could be implemented. Even though the algorithm described in this thesis would be too slow for practical applications, the worth of theoretical research lies elsewhere: both the result itself, and the theoretical framework used to prove it, can impact further theoretical research, which, eventually, might impact practical applications. Because this influence is not direct, the justification for considering this particular problem is also indirect: in Section 1.2 we argue that problems of this type and the chosen formalisms are strongly grounded in theoretical computer science, and we show similarities to some practical applications; in Section 1.3 we argue that the developed theoretical framework is a significant contribution to the field.

1.1.2 Graph databases

Below we build upon the example of reasoning presented earlier, to introduce the main concepts needed to define the problem solved in this thesis. Note that our goal is not to define an accurate model of the world; some inaccuracies are introduced intentionally, to show how the obtained result can help in identifying imperfections of the model.

A family tree[2] can be seen as a simple example of a graph database instance: it contains information about some people and the family relations between them. We encode information using colors: there are green arrows from children to their mothers, and blue arrows from children to their fathers. Additionally, we assume that a family tree contains information about the biological sex of each person, also encoded as a color. An example of a family tree is pictured in Figure 1.1.

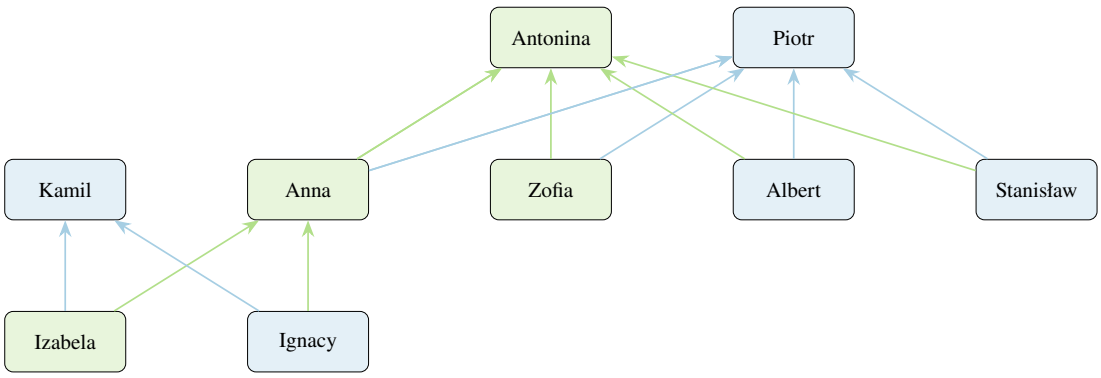


Figure 1.1: An example of a family tree.

Consider the informal notion of the *maximal* family tree, containing all humans that have ever lived. Its exact shape depends on the definition of a *human* and genetic exceptions, where the biological sex or the relation of being a biological parent is not clear. However, we are not interested in the exact definitions of these notions in the real world; the crux of our investigation is the following: given some formal constraints on these notions, we want to reason about their logical coherence. We will analyze the maximal family tree from a mathematical point of view, without any implicit assumptions, by modelling it with a set representing all humans that have ever lived, subsets representing biological sexes, and two relations: child-to-mother and child-to-father. Additionally, we assume that the family

[2]We note that a *family tree* is not, in graph terminology, a *tree*.

tree in Figure 1.1 is a fragment of the maximal family tree. Below we introduce the crucial concepts of this thesis, such as *queries* and *constraints*, by providing examples and non-examples related to the maximal family tree, treated as an instance of a graph database.

1.1.3 Queries

In databases, one of the fundamental operations is executing *queries* on the database instance. A *query* is a broad term that typically includes *requests* to modify the data; however, we are interested exclusively in queries that can be intuitively understood as *questions*. We work with *conjunctive queries* (CQs) and *conjunctive regular path queries* (CRPQs). Table 1.1 contains examples of queries concerning the maximal family tree, and their answers – or partial answers, based on the fragment presented in Figure 1.1. For each query we say whether it is a CQ and/or a CRPQ, and if not, we point out the main reason; below we provide a brief discussion of these reasons, along with an intuitive explanation for the terms *conjunctive*, *regular* and *path* in the names *conjunctive queries* and *conjunctive regular path queries*.

#	Query	Answer	CQ	CRPQ	Key property
1	Is there a female human?	✓	✓	✓	
2	Is there a male human with a female sibling?	✓	✓	✓	
3	Who is Zofia's father?	Piotr	✓	✓	
4	Who are the female ancestors of Ignacy?	Anna, Antonina, ...	×	✓	non-locality
5	Who has the most children?	?	×	×	counting
6	Is there a human without children?	?	×	×	negation
7	Do all humans have a mother?	?	×	×	universal condition

Table 1.1: Examples of queries about the maximal family tree.

The first query is one of the simplest possible examples of a conjunctive query.

The second query can be expressed as a CQ or as a CRPQ, but this requires expressing the relation of being a *sibling* using the relations and concepts available in the database, for example as the query "Are there humans h_1, h_2, h_3, h_4 such that h_1 is male, h_2 is female, h_3 is a mother of h_1 and of h_2, and h_4 is a father of h_1 and of h_2?". Note that this is a *conjunction* of several conditions. This example prompts us to make two observations.

- The wording "a mother" instead of "the mother" is intentional: we do not assume that everyone in the maximal family tree has exactly one mother (which is discussed later), and it more accurately corresponds to what can be expressed by a conjunctive query.
- Because of the inability of conjunctive queries to express inequality, everyone is considered to be their own sibling by the above interpretation.

The first two queries are *Boolean* (yes-no questions), in contrast to the third query, which could be written formally as follows: "What is the set of all humans h such that h is a father of Zofia?". Note that it refers to an individual (Zofia); importantly, we treat names as a way to refer to specific humans, but they play a different role than the "colors", such as the biological sex of a human.

The fourth query is similar to the previous one, but multiple humans satisfy the specified conditions.

The query is not a CQ, because it uses a non-local notion of an *ancestor*, which can be arbitrarily far away in the tree; in contrast, CRPQs can use *regular expressions*, such as `(mother|father)+`, to define relations that involve arbitrarily long sequences of arrows, called *paths* in graph terminology.

The last three queries cannot be answered based on the provided fragment of the maximal family tree, and are neither CQs nor CRPQs. The fifth query involves counting, which is closely related to inequality. The last two queries involve negation ("without children") or a universal condition ("all humans"), which can also be seen as a form of negation, and cannot be expressed by CQs and CRPQs.

1.1.4 Constraints

Based on our knowledge about the world, we can specify some *constraints* that the database instance should satisfy. An example of such a constraint is the statement "every human has a mother, who is also a human". We will describe some of them as *modal constraints*. The precise relation of modal constraints to modal logic requires some explanation; importantly, it is not based on the original semantics of modal logic, but rather on the underlying formal language(s).

Modal logic is described in Stanford Encyclopedia of Philosophy (Garson 2024) as follows.

> A modal is an expression (like 'necessarily' or 'possibly') that is used to qualify the truth of a judgement. Modal logic is, strictly speaking, the study of the deductive behavior of the expressions 'it is necessary that' and 'it is possible that'. However, the term 'modal logic' may be used more broadly for a family of related systems. These include logics for belief, for tense and other temporal expressions, for the deontic (moral) expressions such as 'it is obligatory that' and 'it is permitted that', and many others.

The notions of necessity and possibility in modal logic are well-suited to express properties of states of a nondeterministic process. A crucial property of modal logic is its *internal* perspective: rather than expressing a global property of the whole process, a modal logic formula is evaluated in a particular state; this state either satisfies or does not satisfy the formula. This internal perspective is reflected in the underlying formal language by the lack of variables.

Translating the language of modal logic to the maximal family tree example, a modal formula represents some description of a human, such as "[this human] has a mother who is female". The formula can talk about colors (the biological sex) and can follow the outgoing arrows (to mothers and fathers), combining the following conditions: universal ("every [mother/father]"), existential ("some [mother/father]"), conjunction ("and"), disjunction ("or") and negation ("not"). The formulas can be nested: for example, "[this human] has some mother whose every father is male". Importantly, the universal and existential quantifiers can only refer to outgoing arrows; we cannot say "every human" or "some human" within a modal formula. This condition corresponds to *guarded* fragments of classical logics.

The nesting results in a branching structure of a formula, and the lack of variables has very important consequences: intuitively, different "branches" of the formula do not interact with each other. In the basic modal logic we cannot say "[this human] has a mother and a father who are the same person", nor use a self-reference, such as "[this human] is their own father". This results in the basic modal logic having the *tree model property*: if a formula is satisfied in some graph, then it is satisfied in a

tree, which has a similarly branching structure. However, in some extensions of modal logic, special constructors are introduced (e.g. for counting, or for self-reference) that make the above properties expressible, but the interaction between branches is still strictly limited by the available constructors. As a result, even some very expressive description logics have properties analogous to the tree model property.

To express global constraints on the whole structure using modal formulas, we simply say that a formula is satisfied globally (e.g. "every human is male").

We present examples of constraints in Table 1.2, marking which ones are modal; as the term *modal logic* encompasses a family of logics, some constraints might be treated as modal or not, depending on the chosen formalism, which is denoted by a question mark in the table.

Constraint	**Modal**	**Key property**
Every human is either male or female.	✓	
Every human has a mother.	✓	
No human has a grandfather.	✓	
No human has a mother and a father who are siblings.	×	equality (of grandparents)
Every human has a female ancestor.	?	non-locality
Every human has exactly one mother.	?	counting / functionality
Every human has a child.	?	reverse arrow direction
Every human has Albert as a father.	?	referring to an individual

Table 1.2: Examples of constraints in a graph database. Note that not all presented constraints are satisfied in the maximal family tree.

In real databases, such constraints are usually part of the *database schema*, and their main purpose is to protect against accidental modifications that would violate the integrity of the data. In the problem we consider, the constraints play a larger role, because instead of working with one database instance, we will ask questions about all possible instances that satisfy the given constraints, as explained below.

1.1.5 Query entailment

Note that, in Table 1.1 on page 4, we listed queries concerning the whole maximal family tree, and we were unable to provide some answers based on the available data. However, if we take into account some knowledge about the maximal family tree, in the form of constraints, we might be able to answer them. As a trivial example, if we assume that every human has a mother, then we can give a positive answer to the query "Do all humans have a mother?". The query "Does Stanisław have a grandmother?" can also be answered positively under these constraints, which involves some simple reasoning.

The above questions are examples of the problem of *query entailment* in the field of knowledge representation and reasoning: based on fragmentary data and constraints on the full database instance, we ask whether we can be sure that a given Boolean query (yes-no question) has a positive answer in the whole database instance. Equivalently, we ask if the query is satisfied in every possible database instance that contains the given fragment and satisfies the constraints.

In the context of databases, we usually assume that, by definition, a database instance must be finite. However, the problem of query entailment is often considered without this assumption, allowing for infinite instances as well, which often makes the problem easier to solve, but might be questionable for representing the real world. This is briefly discussed below, after introducing the problem of query containment under constraints and rephrasing the earlier reasoning example. In this thesis, we work with finite instances.

1.1.6 Query containment under constraints

Consider a non-Boolean query, such as "Who are the ancestors of Izabela?". In Table 1.1 on page 4, we gave a *list* of ancestors as the answer to a similar query, which is how a real database would likely represent it. In theoretical work, the order of elements in this list usually does not matter. However, there are two approaches to duplicate elements: either no duplicates are allowed (under *set semantics*), or relations and answers to queries are defined as *bags* (*multisets*), where duplicates are allowed (under *bag semantics*). We take the first approach, which means that we treat answers to queries as sets.

The problem of *query containment under constraints* is similar to the query entailment problem, but instead of fragmentary data we consider another query. That is, we are given two queries q_1, q_2 and some constraints, and we ask whether the set of answers to q_1 is always guaranteed to be contained in the set of answers to q_2, under the given constraints. Below we provide two examples.

- Under given constraints, is it guaranteed that every mother is female? More formally: is the set of all mothers contained in the set of all females?
- If x is a sibling of y, is it true that x and y have a common ancestor? More formally: is the set of pairs of siblings contained in the set of pairs of humans with a common ancestor?

In the case of Boolean queries, *containment* can be understood as *implication* or *entailment*. In many settings, including the one we consider, the finite query entailment problem can be seen as a special case of the problem of query containment under constraints, where one of the queries corresponds to the fragmentary data.

Let us see how to express the initial reasoning example as an instance of the problem of CRPQ containment under modal constraints.

- Query q_1: Is there at least one human?
- Query q_2: Is there a human who is their own ancestor?
- Constraints: Every human has a mother, who is also a human.

The answer to this instance of the problem of query containment under constraints is positive: if the constraints are satisfied and q_1 has a positive answer, then q_2 also has a positive answer, as discussed earlier. Note that the implicit assumption that every database instance is finite is crucial here.

In this case, if we are unwilling to accept someone being their own ancestor, solving the above problem tells us that there is some mistake in the way we modelled the world. The most reasonable explanation is that the assumption that "every human has a mother, who is also a human" is incorrect; for every precise definition of a human, one can point out the first human in history, whose mother is not considered human. However, the math behind the reasoning provides alternative solutions: maybe the first human had no parents at all; maybe there were infinitely many people in the past; or maybe time is

cyclic, and a person can be their own ancestor. Note that similar ideas can be found in philosophy and religious beliefs.

Once again we highlight the difference between our setting, in which we consider only finite database instances, and a common setting of the query entailment problem, where the considered instances might be infinite. The finiteness assumption is more natural for modelling the real world, and the above reasoning is an example of this – although infinity is often a convenient simplification.

1.1.7 A formal statement of the result

As the main result of this thesis, we prove that the problem of query containment under constraints is *decidable* for CRPQs and modal constraints. That is, we describe a computer program that correctly answers any question of the form presented above.

The problem of query containment under constraints is defined as follows, for fixed formalisms in which queries and constraints are expressed.

Query containment under constraints

Input: Constraints $\mathcal{S}$ and queries q_1, q_2.

Question: Is the set of answers to q_1 contained in the set of answers to q_2 in every database instance that satisfies $\mathcal{S}$?

Instead of CRPQs, we work with *unions* of conjunctive regular path queries (UCRPQs), which effectively means that we can use disjunction ("or") in the queries. We express modal constraints using a *terminological box* (TBox) of a description logic $\mathcal{ALCO}_{\text{reg}}$. The basic modal logic corresponds to the description logic $\mathcal{ALC}$ (*Attributive Concept Language with Complements*). Other description logics are obtained by extending $\mathcal{ALC}$ with various features; in this case, regular expressions, denoted by the subscript "reg", and *nominals* – the ability to refer to individuals – denoted by the letter $\mathcal{O}$. The main result of this thesis is the following theorem.

Theorem 1.1. The problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox is decidable.

It follows that both the finite query entailment problem and the problem of query containment under constraints are decidable for CRPQs and $\mathcal{ALC}$, which has not been known before.

Different definitions of the problem

Below we mention other definitions of the problem of query containment under constraints that appear in the literature.

Calvanese, Eiter, and Ortiz (2009) define the problem of query containment under constraints in the context of arbitrary models, potentially infinite ones. As shown with the example discussed earlier, the restriction to finite models significantly changes the problem. In the unrestricted case, Calvanese, Eiter, and Ortiz solve the problem for a very expressive query language (UC2RPQs) and very expressive description logics ($\mathcal{ZOI}$, $\mathcal{ZOQ}$); in (Calvanese, Eiter, and Ortiz 2014) the (unrestricted) query entailment problem is also solved for the same query language and the description logic $\mathcal{ZIQ}$.

Bienvenu, Lutz, and Wolter (2012) consider a more general variant of the problem of query containment under constraints, where each query comes with a separate set of constraints. This variant of the problem is undecidable even for conjunctive queries and the description logic $\mathcal{ALCF}$, extending $\mathcal{ALC}$ with the ability to declare some roles as functional.

Under bag semantics, even the problem of query containment *without constraints* is undecidable for unions of conjunctive queries (Ioannidis and Ramakrishnan 1995).

1.2 The state of research on query containment under constraints

In this section we discuss the state of research related to the solved problem. Because this involves certain results for the finite satisfiability problem and the finite query entailment problem, we first provide some context for these problems and their relation to the problem of query containment under constraints. Next, we present the current state of research, in two parts: the first part concerns "local" formalisms – unions of conjunctive queries and extensions of modal logic without transitivity or regular expressions – and the second part discusses "non-local" formalisms. Along the way, we justify the importance of the considered problems and formalisms.

1.2.1 Relation to finite satisfiability and query entailment

Finite satisfiability

The satisfiability problem goes back to the roots of computer science – from the *Entscheidungsproblem*, through Gödel's incompleteness theorems, to Church and Turing formalizing the notion of an algorithm and showing that the satisfiability problem is undecidable for first-order logic in 1936-1937. The *finite* satisfiability problem for first-order logic was also proved to be undecidable by Trakhtenbrot in 1950; the difference in reasoning about finite models and unrestricted models gave rise to the field of finite model theory, which is naturally close to computer science, since computers ultimately work with finite objects. The decidability of the (finite) satisfiability problem is regained either by considering other logics, or by restricting the class of structures considered. Some well-known logics for which the finite satisfiability problem is decidable are fragments of first-order logic, multiple variants of modal logic and description logics, and fixed-point logics. These include the two-variable fragment (Scott 1962), the guarded fragment (Andréka, Németi, and van Benthem 1998), the fluted fragment (Quine 1969; Pratt-Hartmann, Szwast, and Tendera 2019), the unary negation fragment (ten Cate and Segoufin 2013), the guarded negation fragment (Bárány, ten Cate, and Segoufin 2015), modal μ-calculus (Scott and de Bakker 1969; Kozen 1982), and guarded fixed-point logic (Grädel and Walukiewicz 1999). When the class of considered structures is restricted to graphs of bounded treewidth, the finite satisfiability problem becomes decidable even for monadic second-order logic (Courcelle 1990).

In this thesis, we restrict the considered class of structures to graphs – more precisely, to interpretations over unary and binary predicates, and constants. This is a natural assumption, because the formalisms we use – regular path queries and description logics (and modal logic) – are typically not defined for relations of higher arities. It is worth noting that Trakhtenbrot's theorem also holds when restricted to such interpretations. We also note that there exists a standard way of representing relations of higher

arities in a graph, called *reification*; however, in this context it typically requires expressing inverses and functionality in the constraints, which is not considered in this thesis, but it is a natural next step. The setting we consider is heavily inspired by graph databases: the structure of data and the type of query languages considered are very similar. The main differences are that we do not support *data types* (numbers, strings, etc.) in a meaningful way, and that we use description logics to specify the schema, which is not standard, as graph databases often do not have a schema at all; the relation to two existing graph schema languages is briefly discussed later.

Recall that the problem of query containment under constraints asks, for constraints $\mathcal{T}$ and queries q_1, q_2, if the set of answers to q_1 is contained in the set of answers to q_2 in every database instance that satisfies the constraints $\mathcal{T}$. In the setting we consider, as we will see in Chapter 3, without loss of generality we can assume that the queries are Boolean, which means that the question can be phrased as "Is the formula $(\mathcal{T} \wedge q_1) \Rightarrow q_2$ true in every finite graph?". By negating this question, we get "Is there a finite graph in which the formula $\mathcal{T} \wedge q_1 \wedge \neg q_2$ is satisfied?".

Finite query entailment

As mentioned earlier, the finite query entailment problem can be seen in many settings, including the one we consider, as a special case of the problem of query containment under constraints. For constraints $\mathcal{T}$, fragmentary data $\mathcal{D}$ (seen as a conjunction of atomic sentences), and a Boolean query q, the finite query entailment problem asks if the formula $(\mathcal{T} \wedge \mathcal{D}) \Rightarrow q$ is satisfied in all finite graphs, which is dual to checking if there exists a finite graph in which the formula $\mathcal{T} \wedge \mathcal{D} \wedge \neg q$ is satisfied.

On the other hand, the problem of query containment under constraints can often be reduced to the finite query entailment problem, and the reduction, while not necessarily trivial, is typically not the main difficulty in solving the problem. For this reason, in the context of describing the current state of research, we can essentially treat the two problems as equivalent.

1.2.2 Local formalisms: conjunctive queries and modal constraints

When describing the current state of research, we focus on extensions of conjunctive queries and constraints related to modal logic. This is justified by the fact that conjunctive queries are well-behaved theoretically and form the core of many practical queries, while modal logic arises naturally in multiple different contexts and has good computational properties. Below we justify the importance of modal logic and (unions of) conjunctive queries in more detail. Next, we present the current state of research on the problem of query containment under constraints for these formalisms and their extensions, but without non-local features, which are discussed later.

We pay special attention to one particular description logic, $\mathcal{ALCQI}$, extending $\mathcal{ALC}$ with counting ($\mathcal{Q}$) and inverses ($\mathcal{I}$); even though our result does not concern this logic, one of the important advantages of the techniques developed in this thesis over the existing techniques is that they might be extendable to handle this logic as well, which is argued in Section 1.3.

Theoretical importance

Conjunctive queries can be characterized as the select-project-join fragment of relational algebra (intuitively, select-from-where queries in SQL, without inequalities), or as a syntactic fragment of first-order logic, using only existential quantifiers and conjunction. Similarly, *unions* of conjunctive queries are essentially the existential positive fragment of first-order logic. These notions are closely related to the fundamental notion of a *homomorphism* between two structures: a first-order formula is preserved under homomorphisms if and only if it is logically equivalent to an existential positive first-order formula.

Modal logic is even older than computer science, as it comes from philosophy. Due to its long history, multidisciplinary character, and rediscoveries in different contexts, there are multiple different perspectives on modal logic; the main two can be described in a simplified way as follows.

- Extensions of classical logics (e.g. of the propositional logic) with operators describing *modalities* – originally, *possibility* and *necessity*, but later applied also to obligation, time, computation, and many others. The formulas are evaluated in *Kripke structures*, consisting of *possible worlds* and an *accessibility relation* between them.
- Fragments of classical logics (e.g. of first-order logic), in which the accessibility relation is treated as a relation in the classical sense, and modalities correspond to existential or universal quantifiers, but restricted to a special "guarded" form.

Using the second perspective, modal logic can be very precisely related to classical logics through the notion of *bisimulation*; van Benthem (1977) identifies modal logic with the bisimulation-invariant fragment of first-order logic. In this context, it is also worth mentioning an analogous relation between monadic second-order logic and modal μ-calculus (Janin and Walukiewicz 1996).

We treat description logics as variants of modal logic, although this connection was not obvious at first: it was observed by Schild (1991) that the description logic $\mathcal{ALC}$ can be seen as a syntactic variant of the basic modal logic with multiple modalities (*multi-modal logic K*), and that the description logic $\mathcal{ALC}_{\text{reg}}$ corresponds to the Propositional Dynamic Logic.

Inspiration from practice

The query containment problem most notably occurs in relational databases in the context of optimizing queries in the presence of views. Several of the most popular database engines, according to their documentation, support automatic query optimization based on the created views, even if the query does not explicitly mention the view; for example, see (Oracle 2025). Deciding whether a query can be answered using a view is very closely related to the problem of query containment. It seems that, at least in some cases, integrity constraints are taken into account in this process – for example, foreign keys, `NOT NULL` and `UNIQUE` constraints are taken into account when reasoning about joins – which justifies thinking of it as the problem of query containment under constraints.

Applications of modal logic can be found in hardware and software verification (particularly in the form of *temporal logic* and *dynamic logic*), and in knowledge representation and reasoning. Description logics are the basis of the Web Ontology Language (OWL) – a W3C recommendation. Two prominent examples of ontologies used in practice and conforming to some extent to the OWL standard are

SNOMED CT and the Gene Ontology. However, we should note that there are three *profiles* in the current version of OWL, and the mentioned ontologies are most closely related to the simplest one, to the author's knowledge. Nevertheless, even the simplest profile supports transitive roles, and the more complex profiles support other discussed features, such as inverses and functionality. Implementations of reasoners for OWL that support query entailment exist: (Matentzoglu et al. 2015) studied 35 OWL reasoners, 11 of which support query answering, which, at least in theory, is typically equivalent to query entailment. A more recent survey (Abicht 2023) analyzes 73 OWL reasoners, and deems 25 as usable and actively maintained.

The description logic $\mathcal{ALCQI}$ was identified over two decades ago as a suitable description logic for data modelling, due to its ability to express the Entity-Relationship Model (Calvanese, Lenzerini, and Nardi 1998), as well as UML class diagrams (Berardi, Calvanese, and De Giacomo 2005). In particular, $\mathcal{ALCQI}$ can express some important types of integrity constraints considered in database theory, such as unary inclusion dependencies and unary functional dependencies. As mentioned earlier, in the context of the problem of query containment under constraints, expressing inverses and counting (more precisely, functionality) in the constraints can be used to represent relations of higher arities in the graph setting.

Relevant results

The relevant results come from several fields: database theory, knowledge representation and reasoning, and studies of fragments of classical logics. We try to present the results in the chronological order. Recall that for now we are interested in "local" formalisms, without transitivity and regular expressions. We highlight three techniques that we will refer to later: *canonical databases*, the *finite chase procedure*, and *cycle reversing*. For simplicity, in the descriptions below we assume that the queries are Boolean.

For the problem of query containment without constraints, the foundational result is the NP-completeness of the CQ containment (Chandra and Merlin 1977), and its connection to homomorphisms between database instances. The solution can be described as evaluating one of the queries in the *canonical database* of the other query. The canonical database can be seen as a "minimal" database instance in which the query is satisfied, where the minimality is defined based on the homomorphism order.

In database theory, constraints are often expressed as *inclusion dependencies* and *functional dependencies* – or, more generally, *tuple-generating dependencies*. Maier, Mendelzon, and Sagiv (1979) developed the *chase procedure* in the context of the problem of *constraint implication*: deciding if one set of constraints implies another set of constraints. Johnson and Klug (1982) adjusted the chase procedure to the problem of CQ containment under constraints for inclusion and functional dependencies; intuitively, the procedure extends the canonical database of one of the queries to construct a minimal (in the homomorphism order) database instance that additionally satisfies the constraints. However, the resulting database instance might be infinite; the authors identify a restricted class of dependencies for which the answer is the same for finite and unrestricted databases – this property is called *finite controllability*.

Cosmadakis, Kanellakis, and Vardi (1990) developed a technique that is sometimes called *cycle reversing*, in the context of constraint implication for functional dependencies and unary inclusion

dependencies. The technique provides a way to "saturate" the set of constraints with all dependencies that need to be satisfied in finite models; the name comes from the fact that this saturation process identifies constraints that enforce the existence of a cycle, and introduces constraints corresponding to the same cycle traversed in the opposite direction.

Rosen (1997) proved that modal logic can be seen as the bisimulation-invariant fragment of first-order logic in finite structures, and Otto (2004) proved analogous results for several variants of modal logic, including modal logic with inverses, using constructions based on groups of high girth. Even though the results are not directly related to queries, obtaining models of high girth is often a crucial step in solving the problem of query containment under constraints.

Rosati (2006) proved that the problem of UCQ containment under constraints is decidable for arbitrary inclusion dependencies, by developing the *finite chase procedure*. In the finite case, a minimal database instance that satisfies the constraints and one of the queries typically does not exist, but, for a given positive integer k, the finite chase procedure constructs a database instance that is "locally minimal" – intuitively, its substructures of size at most k are minimal. This property is closely related to high girth.

In the realm of knowledge representation and reasoning, Rosati (2008) considered the finite query entailment problem for some variants of the description logic *DL-Lite*, and solved it using the cycle reversing technique. Later, the same technique has been applied in the context of Horn fragments of description logics by Ibañez-García, Lutz, and Schneider (2014) and Boneva et al. (2023).

Pratt-Hartmann (2009) proved that the finite UCQ entailment problem is decidable for the two-variable guarded fragment of first-order logic with counting ($\mathcal{GC}^2$), which subsumes the description logic $\mathcal{ALCQI}$. Pratt-Hartmann reduces the problem to the finite satisfiability problem for $\mathcal{GC}^2$, using the fact that $\mathcal{GC}^2$ can express tree-shaped UCQs and their negations, and by describing combinatorial constructions that, intuitively, produce structures of high girth, which allows focusing on tree-shaped queries – although the construction distinguishes between different types of cycles. It is worth noting that $\mathcal{GC}^2$ and $\mathcal{ALCQI}$ are not finitely controllable.

For the guarded fragment of first-order logic, Bárány, Gottlob, and Otto (2010) proved finite controllability, by generalizing the finite chase procedure by Rosati (2006). This result implies the finite controllability for a wide range of description logics, up to $\mathcal{ALCHOIb}$, which extends $\mathcal{ALC}$ with role inclusions ($\mathcal{H}$), nominals ($\mathcal{O}$), inverses ($\mathcal{I}$), and "safe" Boolean combinations of roles (b). The decidability of the finite UCQ entailment problem follows, because the unrestricted UCQ entailment problem is decidable even for the description logic $\mathcal{ALCHOIQb}$ that includes counting ($\mathcal{Q}$) as well, as proved by Rudolph and Glimm (2010). We note that, in very expressive description logics, the interaction between certain features is often restricted to maintain decidability.

Otto (2012) showed that the guarded fragment of first-order logic can be characterized as the guarded bisimulation invariant fragment of first-order logic in finite structures, generalizing the techniques from (Otto 2004) and constructing *highly acyclic Cayley graphs*.

Later, ten Cate and Segoufin (2013) introduced the unary negation fragment of first-order logic (UNFO) and proved that its finite satisfiability problem is decidable, using results from (Otto 2004). Because UNFO subsumes $\mathcal{ALCI}$ and can express UCQs and their negations, this result can be used as an

alternative proof of the decidability of the finite UCQ entailment problem for $\mathcal{ALCI}$. However, UNFO extended with functionality has undecidable finite satisfiability – the proof is implicit in (ten Cate and Segoufin 2013) and explicit in (Danielski and Kieroński 2019) – so this approach cannot be extended to $\mathcal{ALCQI}$.

Amarilli and Benedikt (2015) solve the finite query entailment problem for CQs and combinations of unary inclusion dependencies and arbitrary functional dependencies, by proving finite controllability. They adapt several techniques discussed above: the cycle reversing technique from (Cosmadakis, Kanellakis, and Vardi 1990), the finite chase procedure from (Rosati 2006), and the construction from (Otto 2004) based on groups of high girth.

1.2.3 Non-locality: transitivity and regular expressions

The main result of this thesis concerns unions of conjunctive regular path queries. This is, to the author's best knowledge, the first decidability result in this context that handles full CRPQs. Below we discuss existing results concerning *non-local* features: transitivity or regular expressions, appearing either in the queries or in the constraints. As before, we first justify the importance of these features, and then present the current state of research.

Theoretical importance

Regular languages are one of the most beautiful notions, with roots in the very beginnings of computer science. They can be characterized in multiple different ways: by finite automata, regular expressions, monadic second-order logic, monoids. Regular languages are widely used in practice, most directly in pattern matching and text search, usually in the form of extended versions of regular expressions.

Conjunctive regular path queries can be seen as a natural syntactic combination of conjunctive queries and regular expressions in the context of graphs. Importantly, CRPQs are not a fragment of first-order logic, but rather a fragment of first-order logic with transitive closure, or, more broadly, of monadic second-order logic. They can be seen as the basic form of graph queries considered in the literature, with other considered formalisms usually being more expressive, so it makes sense to consider CRPQs first in the context of proving decidability results. For a survey of other graph query languages, see (Angles et al. 2017); we mention unions of conjunctive *two-way* regular path queries (UC2RPQs), where inverse roles are allowed inside regular expressions, and a very expressive formalism of *regular queries* that has good algebraic properties, and, citing (Reutter, Romero, and Vardi 2017), "seems to be the most expressive fragment of first-order logic with transitive closure that is known to have an elementary containment problem".

Inspiration from practice

Even though the most popular graph query language, Cypher, cannot express all regular path queries (Gheerbrant et al. 2025), the standardized languages can express all UCRPQs: SPARQL (a W3C recommendation), GQL (an ISO standard), and an extension of SQL, SQL/PGQ (an ISO standard).

The suitability of $\mathcal{ALCQI}$ for data modelling can also be argued in the context of graph databases, as $\mathcal{ALCQI}$ is able to express a large fragment of PG-Schema – a recent proposal for a standard graph

schema language (Angles et al. 2023). Another schema language, SHACL (a W3C recommendation), can be encoded in an extension of $\mathcal{ALCQI}$ (Leinberger et al. 2020).

Perhaps the most relevant practical application is *Ontology-Based Data Access* (OBDA), used to provide an abstract, unified view of multiple data sources, presenting a more user-friendly access to the data. For more information we refer the reader to the survey on OBDA (Xiao et al. 2018), which includes examples of implemented systems and examples of adoption in industrial projects. According to this survey, one of the profiles of the OWL standard, *OWL 2 QL*, is designed specifically for OBDA, and SPARQL has been adopted as the de facto standard query language. Thus, OBDA combines description logics and navigational queries, and the fundamental problem of query rewriting is closely related to the query entailment problem.

Relevant results

Florescu, Levy, and Suciu (1998) proved that the problem of CRPQ containment (without constraints) is decidable, and Calvanese et al. (2000) generalized the result to C2RPQs. The approach can be described similarly to the one used for conjunctive queries by Chandra and Merlin (1977): one of the queries is evaluated in canonical databases of the other query. However, this time we need to consider infinitely many canonical databases, so automata techniques are used to reason about them, or to show that only finitely many of them are relevant.

Most of the results for fragments of classical logics cannot be extended to include non-local properties; when the logic is extended with transitivity, the finite satisfiability problem becomes undecidable for the two-variable fragment, the guarded fragment, the two-variable guarded fragment, and the fluted fragment of first-order logic (Grädel, Otto, and Rosen 1999; Grädel 1999; Tendera 2005; Pratt-Hartmann and Tendera 2019). As we will see later, the situation is different in the case of the unary negation fragment.

Deutsch and Tannen (2002)[3] considered the problem of query containment under constraints for some syntactically defined fragments of CRPQs, and variants of tuple-generating dependencies (called *embedded dependencies* in the paper). The positive results concern very restricted fragments of CRPQs: essentially, unions of conjunctive queries with a special role denoting reachability (because we are interested in decidability, we ignore constructors that can be omitted at the cost of increasing the size of the query). Coincidentally, this is the query language we consider in Chapter 7 as an intermediate result; such queries cannot express even a transitive closure of a specific role. The results concern a restricted form of tuple-generating dependencies; in particular, they need to be *full* – without existential quantification – which makes them incomparable with the description logics we consider.

Rudolph (2016) proved several undecidability results in the context of description logics. The most relevant undecidability result concerns the problem of finite query entailment for UCRPQs and the description logic $\mathcal{ALCOIF}$; more precisely, the result is stated for two-way regular path queries (2RPQs), in which inverses can be used inside regular expressions, but the query used in the proof can be expressed as a UCRPQ as well, or even as an extension of unions of conjunctive queries with the

[3]We note that there is a known minor error in a hardness result in (Deutsch and Tannen 2002), identified and corrected in (Figueira et al. 2020), but it does not influence this discussion.

transitive closure of roles. Rudolph also proves that the finite query entailment problem is undecidable for UCQs and the description logic $\mathcal{SHOIF}$ (Rudolph 2016); the letter $\mathcal{S}$ denotes an extension of $\mathcal{ALC}$ with the ability to declare some roles as transitive. In the context of the finite query entailment problem, transitivity ($\mathcal{S}$) and role inclusions ($\mathcal{H}$) essentially allow expressing the transitive closure of a role.

Gogacz, Ibañez-García, and Murlak (2018) introduced an important technique, which served as the basis for research leading to the results presented in this thesis. They solved the finite query entailment problem for UCQs and description logics $\mathcal{SOI}$, $\mathcal{SOF}$ and $\mathcal{SIF}$. In the paper, the technique takes the form of a tree automaton recognizing decompositions of a graph into strongly connected components (SCCs), and distributing the constraints and fragments of the query over these SCCs; we discuss in more detail how we build on this idea in the next section. The authors also define the *coloured blocking principle*, based on ideas introduced by Gogacz and Marcinkowski (2013). This principle is a very powerful member of a family of techniques used to transform an infinite model into a finite one, called *blocking techniques*. In the paper (Gogacz et al. 2020), co-authored by the author of this thesis, we use the two techniques mentioned above to solve the finite query entailment problem for UCQ$^+$s and description logics $\mathcal{ALCOI}^+$ and $\mathcal{ALCOQ}^+$, where the plus sign denotes the ability to express transitive closure of roles. Notably, using the ideas developed by Gogacz et al. (2019), we allow counting over transitive roles, which is typically forbidden, because it easily leads to undecidability (Kazakov, Sattler, and Zolin 2007).

Danielski and Kieroński (2019) proved that the finite satisfiability of the unary negation fragment of first-order logic (*UNFO*) extended with transitivity ($\mathcal{S}$), nominals ($\mathcal{O}$) and role inclusions ($\mathcal{H}$) is decidable. As noted by the authors, this implies the decidability of the finite entailment problem for UCQs and the description logic $\mathcal{SHOI}^{\sqcap}$, where $\sqcap$ denotes role conjunctions. It seems that this statement can be strengthened by replacing UCQs with a significant fragment of UC2RPQs that includes unions of roles under the Kleene star[4]; however, concatenation under the Kleene star does not seem to be expressible in this setting. In this context, it is also worth mentioning that Jung et al. (2018) proved the decidability of the *unrestricted* satisfiability of the logic UNFO$^{\text{reg}}$, extending UNFO with regular expressions; it is not known if the finite variant of the satisfiability problem is decidable[5] – a positive answer would essentially encompass the main result of this thesis.

Bednarczyk and Kieroński (2022) proved the finite controllability for UCQs and description logics $\mathcal{ZOI}$ and $\mathcal{ZOQ}$, where $\mathcal{Z}$ stands for $\mathcal{ALCb}^{\text{Self}}_{\text{reg}}$; the superscript "Self" denotes the ability to refer to self-loops.

Boneva et al. (2023) proved that the containment of UC2RPQs in acyclic UC2RPQs modulo schemas expressed in the Horn fragment of the description logic $\mathcal{ALCIF}$ is decidable. The acyclicity of UCRPQs is a major restriction, and it is crucial for the techniques used in the paper – it is justified by the considered setting of *transformations* of graph databases corresponding to an evolving schema, and by the desired time complexity.

[4]In (Gutiérrez-Basulto et al. 2024) we solved the problem of query containment under constraints for this fragment of UC2RPQs and the description logic $\mathcal{ALCQ}$.

[5]Contrary to the claim in (Pareti et al. 2020); the mistake has been confirmed with the authors.

1.3 The contribution

The main result of this thesis is the decidability of the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox. This seems to be the first decidability result for full UCRPQs in this context. The result is close to the undecidability frontier: the finite query entailment problem for UCRPQs and the description logic $\mathcal{ALCOIF}$ is undecidable (Rudolph 2016).

Relation to published papers

We published the most important part of the contribution – the decidability of the finite query entailment problem for UCRPQs and the description logic $\mathcal{ALC}$ – in (Gutiérrez-Basulto et al. 2022), with the author of this thesis being the main author. In (Gutiérrez-Basulto et al. 2024) we extended the results to the query containment problem, and to description logics $\mathcal{ALCQ}$ and $\mathcal{ALCI}$. These extensions give hope that the techniques can be adjusted to handle the full description logic $\mathcal{ALCQI}$, which, as we argued, is suitable to express graph database schemas.

Most of the techniques presented in this thesis have been heavily reworked compared to the papers. The results including nominals or regular expressions in the logic have not been published before.

Techniques

The techniques developed in this thesis build on multiple techniques mentioned in the previous section, and introduce new ones. Below we briefly discuss the most important ones.

To handle UCRPQs, we incorporated into the algorithm a simple but powerful construction on deterministic finite automata. This construction has already been (re)discovered by Hesse (2003) and Bojańczyk (2009), although in completely different contexts; we are not aware of it being used before to prove a decidability result. On the other hand, as noted by Bojańczyk (2009), the construction can serve a similar purpose to the Factorization Forest Theorem by Simon (1990), which has likely been used in some decidability results.

We split the proof into three main combinatorial constructions; intuitively, one for reducing the problem of query containment under constraints to the finite query entailment problem; one for solving the finite satisfiability problem for constraints, ignoring the query; and one for modifying the graph satisfying the constraints to be "locally minimal", which is related to having high girth – although these conditions need to be adjusted to non-local queries.

The first construction, for the reduction between the problems, has a lot in common with the discussed solution of the CRPQ containment problem (without constraints), where canonical databases of one of the queries are considered. The way we described this reduction in (Gutiérrez-Basulto et al. 2024) is particularly similar: we essentially build a tree automaton that recognizes canonical databases of one of the queries, with some edges represented abstractly. However, the overall reduction as presented in (Gutiérrez-Basulto et al. 2024) cannot be used for the full description logic $\mathcal{ALCQI}$; in this thesis we modify the construction in an attempt to make it more general and easier to describe.

We rework the techniques developed in (Gogacz, Ibañez-García, and Murlak 2018); instead of tree

automata recognizing decompositions of a graph into SCCs, we describe it in a way similar to the cycle reversing technique, as follows. We extend the language of constraints, such that whenever a constraint refers to some edge, it specifies whether this edge belongs to some cycle (SCC) or not. For a given set of constraints, we consider all corresponding sets of constraints in the extended (*SCC-annotated*) language; this can be seen as a nondeterministic variant of the "saturation" process in the cycle reversing technique. It is more difficult to solve the finite satisfiability problem for the SCC-annotated constraints (compared to the original constraints), but the additional information about SCCs works well with the adjusted notion of a graph being "locally minimal".

The interaction between the SCC-annotation technique and the construction on deterministic finite automata is very complicated; to describe it in a clear way, we generalize the SCC-annotation technique to *ranked graphs*, where a rank (a positive integer) is assigned to each edge. We introduce a generalized notion of reachability: for a rank r, $r^{\uparrow}$*-reachability* means reachability through edges of ranks at least r. We prove that these notions are well-behaved and induce a natural notion of *ranked SCCs*. The SCC-annotation technique easily generalizes to the case of ranked SCCs; however, solving the finite satisfiability problem for SCC-annotated constraints in the ranked setting is a major technical challenge.

We introduce a toolbox of techniques for each of the three main combinatorial constructions; most notably, to construct "locally minimal" graphs, we define *ravelling constructions*: a simple and adjustable family of constructions that preserve the satisfaction of modal constraints. They can be used in particular to obtain graphs of a very similar structure to the ones obtained by the finite chase procedure by Rosati (2006). Combining these constructions with an intuitive notion of the *Last Edge Appearance Record* of a walk in a graph, we obtain a powerful tool that allows us to avoid working with infinite models and complicated recursive constructions.

1.4 The structure of this thesis

> When pure mathematicians pull out the messiness of reality from their problems, it's not just out of laziness. I mean, it's a little bit laziness, but it's not just that, and it's not out of apathy for applications. It's because distilling a problem into its core essence can expose hidden connections, and it's through those connections that mathematicians make progress. A hard problem in one context, where it might seem deeply confusing, sometimes can look clearer in another setting, but the analogy may not be obvious, and a lot of the progress of math through history has looked like developing a richer and richer web of hidden connections.
>
> Grant Sanderson, 3Blue1Brown (Sanderson 2025)

The longer format of a PhD thesis is used to move the focus from *how* the proof works to *why* it works. For this purpose, we consider increasingly difficult variants of the problem of query containment under constraints, starting from conjunctive queries, through conjunctive queries with reachability, to conjunctive regular path queries, showing how the constructions need to change along the way, and why the previous ones do not suffice. This shift of focus influences the way of presenting the constructions: instead of hiding the details in an appendix, as it is common especially in conference papers, we purposefully bring them to the surface.

Another difference from papers is a much more rigorous presentation of constructions and proofs. This,

combined with the previous point, prompted the use of a unified, graph-theoretic language, instead of the syntax-heavy definitions of description logics and queries. Defining the notions in a graph setting and translating the problem requires additional work, but the author believes that it is worth it, because it allows formulating general constructions and lemmas reused throughout the thesis.

Because of the shift of priorities explained above, we purposefully focus on the decidability rather than the complexity of the developed algorithms, since technical difficulties that accompany the pursuit of the optimal time complexity can easily overshadow the main ideas of the decidability proofs, which, in the author's opinion, are interesting enough on their own to deserve a proper presentation. However, because the complexity of the resulting algorithms is non-elementary, we briefly discuss in Chapter 11 the main idea for achieving an elementary complexity; for a discussion of the exact complexity of closely related problems, we refer the reader to (Gutiérrez-Basulto et al. 2022, 2024).

Another goal of the taken approach is to make it easier for the reader to detect possible similarities between the techniques developed in this thesis and other techniques used in the literature, even if the language and the setting differ significantly. Already during the process of writing this thesis, the author found two likely connections: several techniques used in (Danielski and Kieroński 2019) seem to have a lot in common with some constructions presented in this thesis, and some techniques developed in (Canavoi and Otto 2017) might also be related. Unfortunately, both papers involve very intricate combinatorial constructions, so verifying these conjectures is a difficult task, but it could result in a fruitful transfer of knowledge between the fields.

The description of chapters

We begin by introducing the setting we work with in Chapter 2: labelled graphs, *modal profiles* used to express modal constraints, and decision problems corresponding to the problem of query containment under constraints and to the finite query entailment problem. In Chapter 3 we define the problem of CQ containment modulo an $\mathcal{ALC}$ TBox and translate it to the graph setting; the translation is straightforward, using some well-known techniques. Chapter 4 contains a reduction from the problem of query containment under constraints to the finite query entailment problem for conjunctive queries. In Chapter 5 we present the toolbox for working in the graph setting, including the core computational problem: the *minimal downsets problem* parametrized by a class of graphs, and a general reduction from the finite query entailment problem to the minimal downsets problem. In Chapters 6 to 9 we solve increasingly difficult variants of the minimal downsets problem, with the main technical contribution of this thesis contained in Chapters 7 to 9: building the toolbox to solve the minimal downsets problem in the presence of reachability constraints (through SCC-annotations), defining ranked graphs, generalizing the introduced techniques to ranked graphs, and reducing the minimal downsets problem for UCRPQs to the case of ranked graphs. Chapter 10 contains a reduction from the problem of query containment under constraints to the finite query entailment problem for UCRPQs. In Chapter 11 we define the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox and translate it to the graph setting – to the problem solved in the previous chapter; additionally, we present a technique that improves the time complexity of developed algorithms, and discuss further optimizations. In Chapter 12 we present the conclusions and briefly discuss the related results in (Gutiérrez-Basulto et al. 2024) that include counting or inverses in the description logic.

Index and hyperlinks

For the readers of the printed version of this thesis, an index is provided at the end of the document, with a list of key notions and references to their definitions. In the electronic version, all occurrences of key notions are hyperlinked to their definitions[6]; in some PDF readers, hovering over the hyperlink will show a preview of the definition.

The research presented in this thesis was supported by the National Science Centre, Poland, under grant number 2018/30/E/ST6/00042.

[6]This is achieved by the `knowledge` package by Thomas Colcombet.

Chapter 2

Preliminaries

In this chapter we define graphs, modal constraints, and two decision problems corresponding to query containment under constraints and finite query entailment.

Technical assumptions for computability

Because we work with decision problems, we need to make some technical assumptions, to make sure that the inputs can be appropriately encoded. Let $\mathbb{U}$ be the smallest set that contains all non-negative integers, all finite subsets of $\mathbb{U}$ and all finite sequences of elements in $\mathbb{U}$. We omit some details, such as the exact model of computation, the input alphabet and the exact encoding, as they do not affect decidability, barring some very unusual choices. The set $\mathbb{U}$ is *computable*: there exists an algorithm that decides whether an input is an encoding of an element in $\mathbb{U}$. Moreover, there are algorithms that perform standard operations on elements of $\mathbb{U}$, such as comparing two numbers, enumerating the elements of sets and sequences, extending sets and sequences, and so on. We use the set $\mathbb{U}$ in the definitions to ensure that the corresponding objects can be encoded as inputs to decision problems.

2.1 Graphs

We work exclusively with **finite** directed graphs in which each node and each edge has a single label assigned. We refer to them simply as *graphs*; we emphasize here that they are finite, as overlooking this assumption might cause some confusion.

A *graph* G is a triple $(V_G, E_G, \mathsf{Label}_G)$, where $V_G \subseteq \mathbb{U}$ is a finite set of *nodes*, E_G is a finite set of *edges* – triples (u, a, v), where $u, v \in V_G$, and $a \in \mathbb{U}$ is the *label* of the edge – and $\mathsf{Label}_G : V_G \to \mathbb{U}$ is a function assigning a *label* to each node. We denote by Γ_G the set of labels of all nodes in G, and by Σ_G the set of labels of all edges in G. An example of a graph is shown in Figure 2.1.

We sometimes refer to *node labels* and *edge labels* outside the context of a fixed graph. Formally, we mean elements of $\mathbb{U}$; informally, we use these notions to inform the reader how these elements will be used. For example, we might define a *ranked edge label* as an edge label of the form (a, r), where r is a positive integer.

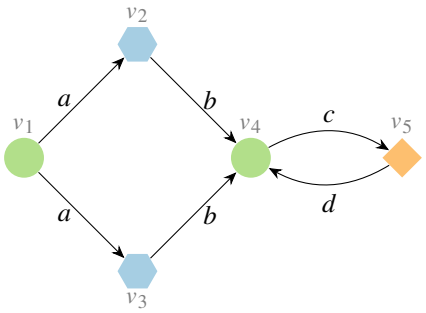


Figure 2.1: An example of a graph G, with nodes $V_G = \{v_1, v_2, v_3, v_4, v_5\}$ and edges $E_G = \{(v_1, a, v_2), (v_1, a, v_3), (v_2, b, v_4), (v_3, b, v_4), (v_4, c, v_5), (v_5, d, v_4)\}$. We represent node labels by their shapes and colors; for example, $\mathsf{Label}_G(v_1) =$ ● and $\mathsf{Label}_G(v_5) =$ ◆. The color of a node is determined by its shape, to improve accessibility.

A *subgraph* H of a graph G is a graph with nodes $V_H \subseteq V_G$ and edges $E_H \subseteq E_G$, with preserved node labels: for each $v \in V_H$, $\mathsf{Label}_H(v) = \mathsf{Label}_G(v)$. The subgraph of G *induced* by a set $V \subseteq V_G$ is the subgraph with nodes V and edges $\{(u, a, v) \in E_G : u, v \in V\}$. The relation of being a subgraph induces a natural (reflexive) partial order on graphs, and allows us to talk about a *maximal subgraph* satisfying some conditions.

Consider an edge (u, a, v) in a graph. We say it is an edge from u to v with label a, *outgoing* from u, *incoming* to v, *between* u and v. The node v is a *successor* of u, and u is a *predecessor* of v. For a node u in a graph G and an edge label a, the set of *a-successors* of u in G, denoted $\mathsf{Succ}_G(u, a)$, is the set of nodes v such that $(u, a, v) \in E_G$.

A graph is *(weakly) connected* if it is impossible to split its nodes into two non-empty sets such that there is no edge between nodes in different sets. A *connected component* of a graph is its maximal connected subgraph.

Two graphs are *disjoint* if their sets of nodes are disjoint. The *union* of a finite set $\mathcal{G}$ of pairwise disjoint graphs is the graph with nodes $\bigcup_{G \in \mathcal{G}} V_G$ and edges $\bigcup_{G \in \mathcal{G}} E_G$, with each node v having the same label as in the unique graph $G \in \mathcal{G}$ such that $v \in V_G$.

The *disjoint union* of a finite set $\mathcal{G}$ of graphs is the graph with nodes $\{(G, u) : G \in \mathcal{G}, u \in V_G\}$ and edges $\{((G, u), a, (G, v)) : G \in \mathcal{G}, (u, a, v) \in E_G\}$; the label of a node (G, u) is $\mathsf{Label}_G(u)$.

Graph mappings

Consider a graph G and a function f from V_G to an arbitrary set, such that f is *label-sensitive*: if nodes u, v have different labels in G, then $f(u) \neq f(v)$. By a slight abuse of notation, we denote by $f(G)$ the *graph-image* of f: the graph with nodes $\{f(v) : v \in V_G\}$ and edges $\{(f(u), a, f(v)) : (u, a, v) \in E_G\}$, in which a node $f(v)$ has the label of v in G; because f is label-sensitive, this label is well-defined.

For two graphs G, H, a *homomorphism* from G to H is a label-sensitive function $h : V_G \to V_H$ such that $h(G)$ is a subgraph of H. The homomorphism h is an *isomorphism* if it is a bijection, and $h(G) = H$. These notions coincide with their standard definitions for labelled directed graphs.

A graph G *maps homomorphically* to a graph H, denoted $G \to H$, if there is a homomorphism from G to H. Two graphs G, H are *homomorphically equivalent*, denoted $G \leftrightarrow H$, if $G \to H$ and $H \to G$. The *homomorphism type* of a graph G, denoted $\mathsf{HomType}(G)$, is the equivalence class of G in the relation consisting of all pairs of homomorphically equivalent graphs. Two graphs G, H are *isomorphic* if there exists an isomorphism between them. The *isomorphism type* of a graph is its equivalence class in the relation consisting of all pairs of isomorphic graphs. Examples of graphs are shown in Figure 2.2, along with a discussion of homomorphisms and isomorphisms between them.

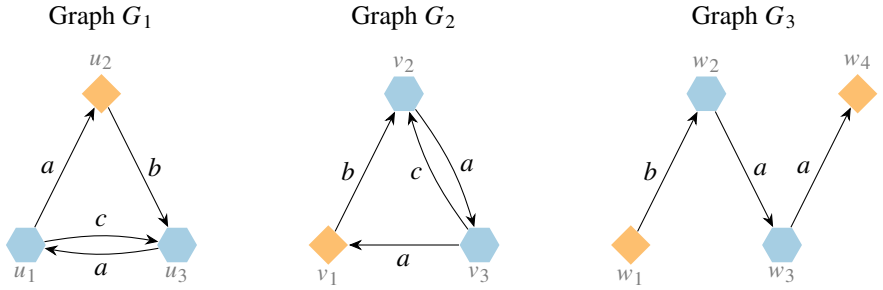


Figure 2.2: Examples of graphs G_1, G_2, G_3. Graphs G_1 and G_2 are isomorphic: there is an isomorphism from G_1 to G_2 mapping u_1 to v_3, u_2 to v_1, and u_3 to v_2. Visually, two graphs are isomorphic if the nodes can be repositioned and renamed in such a way that both graphs look exactly the same. There are homomorphisms from G_3 to both G_1 and G_2; an example of a homomorphism from G_3 to G_2 maps w_1 and w_4 to v_1, w_2 to v_2, and w_3 to v_3. Visually, a graph G maps homomorphically to a graph H if nodes of G can be repositioned, renamed, and possibly merged, to look like a subgraph of H. Graphs G_1 and G_2 do not map homomorphically to G_3: a graph containing a cycle cannot map homomorphically to an acyclic graph.

In the context of decision problems, homomorphism types and isomorphism types of graphs can be represented as an arbitrary graph of this type. Under this representation, checking if two types are equal requires checking if two graphs are isomorphic or homomorphically equivalent.

Homomorphisms between graphs induce a partial order $\preccurlyeq$ on homomorphism types of graphs: for two graphs G, H, $\mathsf{HomType}(G) \preccurlyeq \mathsf{HomType}(H)$ iff $G \to H$; this partial order is correctly defined, as homomorphisms induce a transitive and reflexive relation on graphs. The *downward closure* of a set $\mathcal{T}$ of homomorphism types of graphs is the set of all homomorphism types τ of graphs such that $\tau \preccurlyeq \tau'$ for some $\tau' \in \mathcal{T}$.

Walks

A *walk* in a graph G is a finite, non-empty sequence π of alternating nodes and edges in G, beginning and ending with a node, such that, with $\pi = [v_0, e_1, v_1, e_2, v_2, \ldots, e_m, v_m]$, for each $i \in \{1, \ldots, m\}$, $e_i = (v_{i-1}, a_i, v_i)$ for some edge label a_i. We say that π is a walk *from* v_0 *to* v_m. The sequence $(e_1, \ldots, e_m)$ is the sequence of edges *traversed* by π. The sequence $(v_0, v_1, \ldots, v_m)$ is the sequence of nodes *visited* by π. The sequence $(a_1, \ldots, a_m)$ is called the *edge label sequence* of π. The *length* of π, denoted $|\pi|$, is the length of the sequence of edges traversed by π. A walk π is *trivial* if its length is 0; that is, $\pi = [v_0]$. A walk is *closed* if it is a walk from a node to itself; that is, $v_0 = v_m$. A *cycle* is a

non-trivial closed walk in which all visited nodes are different, except that the first and last nodes are the same. We denote by $\mathsf{LastNode}(\pi)$ the last node in the walk π.

Note that we use square brackets to denote walks; this is done to clearly distinguish a trivial walk $[v]$ from the node v, and a walk $[u, e, v]$ of length 1 from an edge (u, a, v).

The notions of *prefixes*, *infixes* and *suffixes* of walks coincide with these notions for sequences – note that we require them to be correct walks, so they must begin and end with a node. If a walk π_1 is a prefix of a walk π_2, then π_2 is an *extension* of π_1. For a walk $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$ and an edge $e = (v_m, a, v)$ outgoing from v_m, the *extension of π by the edge e*, denoted $\pi \cdot e$, is the walk $[v_0, e_1, v_1, \ldots, e_m, v_m, e, v]$. For a walk π in a graph G and an edge label a, let $\mathsf{Extensions}_G(\pi, a)$ denote the set of all walks in G that are extensions of π by an edge with label a.

In a graph G, a node v is *reachable* from a node u if there is a walk from u to v in G.

2.2 Modal constraints

To express modal constraints in graphs, we will talk about (partial) unravellings of graphs, and their bisimilarity types. Because these are central notions for this thesis, we carefully introduce the notions and explain them with examples.

2.2.1 Trees

A *tree* (also known in this setting as an *arborescence*) is a connected graph in which one node, called the *root*, has no incoming edges, and each other node has exactly one incoming edge. Equivalently, a tree is a graph in which there is a node r, called the root, such that every node is reachable from r by exactly one walk. A *leaf* in a tree T is a node with no outgoing edges.

The *depth* of a tree T is the maximal length of a walk in T. The depth of a node in T is the length of the walk from the root to this node. For a non-negative integer k, we denote by $T|_k$ the subgraph of T induced by all nodes of depth at most k; this subgraph is a tree as well.

We denote by $\mathsf{Root}(T)$ the root of T, and by $\mathsf{RootLabel}(T)$ its label. In trees, successors are called *children*, and predecessors are called *parents*.

A crucial property of trees is their recursive structure. For a node u in a tree T, the *subtree of T rooted in u*, denoted $\mathsf{Subtree}_T(u)$, is the subgraph of T induced by all nodes reachable from u; this subgraph is also a tree. For an edge label a, we define $\mathsf{Subtrees}(T, a)$ as the set of subtrees of T rooted in nodes connected to the root by an edge with label a: $\{\mathsf{Subtree}_T(v) : v \in \mathsf{Succ}_T(\mathsf{Root}(T), a)\}$. Because we work with finite trees, for every tree T:

- for each edge label a, each tree in $\mathsf{Subtrees}(T, a)$ has depth strictly smaller than the depth of T;
- $\mathsf{Subtrees}(T, a)$ is finite for every edge label a, and non-empty for finitely many edge labels a.

The first condition, corresponding to *finite depth*, is particularly important for performing inductive reasoning and for recursive definitions. The second condition corresponds to *finite branching*.

2.2.2 Unravellings

For a node u in a graph G and a non-negative integer n, the *n-step unravelling* of G from u, denoted $\mathsf{Unravel}(n, G, u)$, is the tree in which the nodes are all walks in G that begin in u and have length at most n, and there is an edge (π_1, a, π_2) iff $\pi_2 \in \mathsf{Extensions}_G(\pi_1, a)$. The label of a node π is the label of $\mathsf{LastNode}(\pi)$ in G. An example of a graph and its 3-step unravelling is shown in Figure 2.3.

The set of *n-step unravellings of G* is the set consisting of the n-step unravellings of G from all nodes.

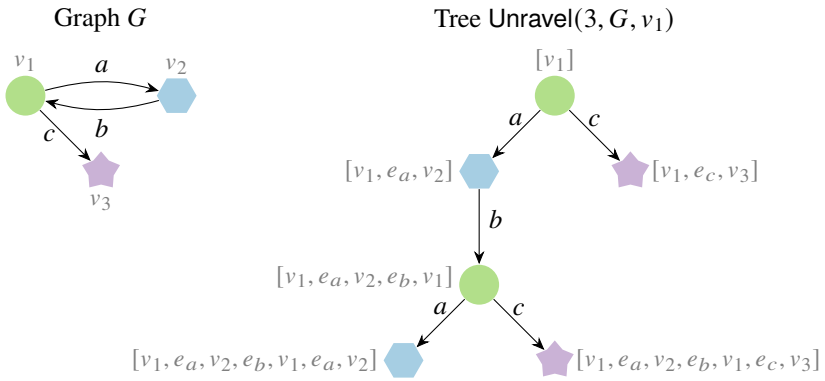


Figure 2.3: A graph G and its 3-step unravelling from v_1. Nodes in the tree are walks in G; we denote by e_a the unique edge in G with label a, and similarly for labels b and c.

Let us look at the recursive structure of a tree $T = \mathsf{Unravel}(n, G, u)$. If $n \geq 1$, the set $\mathsf{Subtrees}(T, a)$ corresponds to the set $\{\mathsf{Unravel}(n-1, G, v) : v \in \mathsf{Succ}_G(u, a)\}$, up to renaming nodes. This provides a "local" view on unravellings, which is crucial for inductive proofs. On the other hand, the walk-based definition of unravellings clearly shows their "global" structure, which is very useful for defining combinatorial constructions on graphs that preserve some properties of unravellings. The possibility of switching between these two views is an important tool in our arsenal.

2.2.3 Bisimilarity

Our goal is to define the *bisimilarity type* of a tree. We begin by providing a standard definition of bisimilarity, followed by a simple recursive characterization of bisimilarity types of trees, which we use throughout the thesis.

The standard definition

The notion of bisimilarity is usually defined for *labelled transition systems*, which correspond to *rooted graphs* in our setting: graphs with a single distinguished node. Two rooted graphs (G_1, r_1), (G_2, r_2) are *bisimilar* if there exists a relation $R \subseteq V_{G_1} \times V_{G_2}$ such that $(r_1, r_2) \in R$ and, for each $(u_1, u_2) \in R$ and each edge label a:

- $\mathsf{Label}_{G_1}(u_1) = \mathsf{Label}_{G_2}(u_2)$;
- for each $v_1 \in \mathsf{Succ}_{G_1}(u_1, a)$ there is some $v_2 \in \mathsf{Succ}_{G_2}(u_2, a)$ such that $(v_1, v_2) \in R$;
- for each $v_2 \in \mathsf{Succ}_{G_2}(u_2, a)$ there is some $v_1 \in \mathsf{Succ}_{G_1}(u_1, a)$ such that $(v_1, v_2) \in R$.

Bisimilarity is an equivalence relation on rooted graphs: it is reflexive, symmetric and transitive.

In the context of bisimilarity, we treat a tree T as a rooted graph $(T, \mathsf{Root}(T))$; that is, two trees T_1, T_2 are *bisimilar* if $(T_1, \mathsf{Root}(T_1))$ and $(T_2, \mathsf{Root}(T_2))$ are bisimilar.

Fact 2.1. For two trees T_1, T_2:

- if T_1, T_2 are isomorphic, then T_1, T_2 are bisimilar;
- if T_1, T_2 are bisimilar, then T_1, T_2 are homomorphically equivalent.

Remark. Two trees are homomorphically equivalent iff they are *simulation equivalent*, in the language of transition systems.

Consider the trees in Figure 2.4. The trees T_1 and T_3 are bisimilar, which is witnessed by the relation $R = \{(w_0, u_0), (w_1, u_1), (w_2, u_2), (w_3, u_1), (w_4, u_2)\}$. The trees T_1 and T_2 are not bisimilar, intuitively, because the node v_3 in T_2 has label ⬣ and no outgoing edges, while the only node with label ⬣ in T_1 is u_1, and it has an outgoing edge.

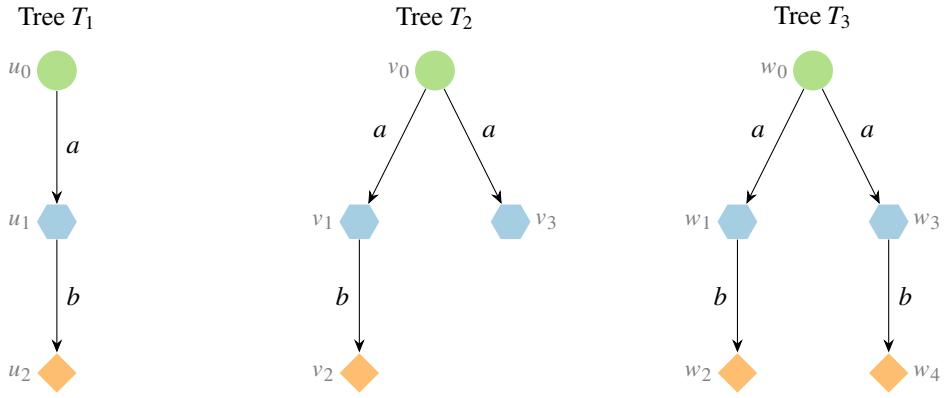


Figure 2.4: Examples of trees.

Tree types

Below we introduce a simple recursive characterization of isomorphism types, bisimilarity types and homomorphism types of trees. We will use this characterization of bisimilarity types throughout the thesis. The characterization of homomorphism types is used in Chapter 11 in a discussion of the time complexity of the developed algorithms. Isomorphism types are considered below only for comparison.

First, let us present slightly informal definitions, which hide some details about their inductive nature. Later we provide a formal definition for bisimilarity types.

For a tree T, we define $\mathsf{IsoType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the **multiset** $\wr \mathsf{IsoType}(T') : T' \in \mathsf{Subtrees}(T, a) \wr$.

Fact 2.2. Two trees T_1, T_2 are isomorphic iff $\mathsf{IsoType}(T_1) = \mathsf{IsoType}(T_2)$.

For a tree T, we define $\mathsf{BisimType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the **set** $\{\mathsf{BisimType}(T') : T' \in \mathsf{Subtrees}(T, a)\}$.

Fact 2.3. Two trees T_1, T_2 are bisimilar iff $\mathsf{BisimType}(T_1) = \mathsf{BisimType}(T_2)$.

In Chapter 11, to provide a similar characterization of homomorphism types of trees, we will work with *rooted homomorphisms* that map the root of one tree to the root of another tree, and the induced notions of *rooted homomorphism types* and the *rooted downward closure*. For a tree T, we define $\mathsf{RootedHomType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the **rooted downward closure** of the set

$$\{\mathsf{RootedHomType}(T') : T' \in \mathsf{Subtrees}(T, a)\}.$$

Fact 2.4. For trees T_1, T_2, $\mathsf{RootedHomType}(T_1) = \mathsf{RootedHomType}(T_2)$ iff T_1 and T_2 are homomorphically equivalent.

Bisimilarity types: a formal definition

We define the *bisimilarity type* of a tree T inductively over the depth of T. For a non-negative integer n, *n-bisimilarity types* are bisimilarity types of all trees of depth at most n.

The *bisimilarity type* of a tree T of depth n is the pair (A, f), where $A = \mathsf{RootLabel}(T)$, and, for each edge label a, $f(a)$ is the set of bisimilarity types of trees in $\mathsf{Subtrees}(T, a)$; formally, f is a function from edge labels to sets of $(n-1)$-bisimilarity types if $n \geq 1$, or to the empty set otherwise.

To avoid working with pairs (A, f) directly, we define additional functions. We denote the bisimilarity type of a tree T as $\mathsf{BisimType}(T)$. For a tree T and its bisimilarity type $\tau = (A, f)$, we extend the function $\mathsf{RootLabel}$ to τ, and define a function $\mathsf{Subtypes}$ as follows:

- $\mathsf{RootLabel}(\tau) = A = \mathsf{RootLabel}(T)$;
- for each edge label a, $\mathsf{Subtypes}(\tau, a) = f(a) = \{\mathsf{BisimType}(T') : T' \in \mathsf{Subtrees}(T, a)\}$.

A *tree bisimilarity type* is a bisimilarity type of some tree. Note that in the above definitions we use the fact that we work with trees of finite depth. Because they are finitely branching, for every tree bisimilarity type τ, the set $\mathsf{Subtypes}(\tau, a)$ is finite for each edge label a, and is non-empty for finitely many edge labels a.

Bisimilarity types: examples

Below we write out bisimilarity types of trees in Figure 2.4 on the preceding page, and of their subtrees. We will define the following tree bisimilarity types:

- $\tau_{\blacklozenge}$, corresponding to subtrees rooted in u_2, v_2, w_2 and w_4;
- $\tau_{\bullet}$, corresponding to the subtree rooted in v_3;
- $\tau_{\bullet\blacklozenge}$, corresponding to subtrees rooted in u_1, v_1, w_1 and w_3;
- $\tau_{1,3}$, corresponding to trees T_1 and T_3;
- τ_2, corresponding to the tree T_2.

For each tree bisimilarity type τ, we list all non-empty values of $\mathsf{Subtypes}(\tau, a)$.

- $\mathsf{RootLabel}(\tau_{\blacklozenge}) = \blacklozenge$
- $\mathsf{RootLabel}(\tau_{\blacksquare}) = \blacksquare$
- $\mathsf{RootLabel}(\tau_{\blacksquare\blacklozenge}) = \blacksquare$ and $\mathsf{Subtypes}(\tau_{\blacksquare\blacklozenge}, b) = \{\tau_{\blacklozenge}\}$
- $\mathsf{RootLabel}(\tau_{1,3}) = \bullet$ and $\mathsf{Subtypes}(\tau_{1,3}, a) = \{\tau_{\blacksquare\blacklozenge}\}$
- $\mathsf{RootLabel}(\tau_2) = \bullet$ and $\mathsf{Subtypes}(\tau_2, a) = \{\tau_{\blacksquare}, \tau_{\blacksquare\blacklozenge}\}$

Bounding the depth of bisimilarity types

For a tree T, its bisimilarity type τ, and a non-negative integer k, intuitively, we want to define $\tau|_k$ as the bisimilarity type of $T|_k$; recall that $T|_k$ is the subgraph of T induced by all nodes of depth at most k. Formally, we define $\tau|_k$ inductively over k, as the k-bisimilarity type such that:

- $\mathsf{RootLabel}(\tau|_k) = \mathsf{RootLabel}(\tau)$;
- for each edge label a, $\mathsf{Subtypes}(\tau|_k, a)$ is the set $\{\tau'|_{k-1} : \tau' \in \mathsf{Subtypes}(\tau, a)\}$ if $k \geq 1$, or the empty set otherwise.

Fact 2.5. For each tree T and each non-negative integer k, $\mathsf{BisimType}(T)|_k = \mathsf{BisimType}(T|_k)$.

Representatives

Below we define the *representative* of a tree bisimilarity type τ, denoted $\mathsf{Representative}(\tau)$. Informally, it is a tree with:

- a root with label $\mathsf{RootLabel}(\tau)$;
- for each edge label a, for each $\tau' \in \mathsf{Subtypes}(\tau, a)$, an edge with label a from the root to a copy of $\mathsf{Representative}(\tau')$.

The construction is simple, but to define $\mathsf{Representative}(\tau)$ formally, we need to deal with the technical annoyance of defining its set of nodes.

For a tree bisimilarity type τ, let $\mathsf{AllSubtypes}(\tau)$ be defined recursively as the union of $\{\tau\}$ and $\mathsf{AllSubtypes}(\tau')$ for all $\tau' \in \mathsf{Subtypes}(\tau, a)$ for all edge labels a; note that the resulting set is finite. Let G_τ be the graph with nodes $\mathsf{AllSubtypes}(\tau)$, with each node τ' having label $\mathsf{RootLabel}(\tau')$, and an edge (τ_1, a, τ_2) iff $\tau_2 \in \mathsf{Subtypes}(\tau_1, a)$. Then, for a non-negative integer n and an n-bisimilarity type τ, we define $\mathsf{Representative}(\tau)$ as the n-step unravelling of G_τ from τ: $\mathsf{Unravel}(n, G_\tau, \tau)$.

Fact 2.6. The bisimilarity type of $\mathsf{Representative}(\tau)$ is τ.

Remark. The graph G_τ defined above is the *bisimulation quotient* of every tree of bisimilarity type τ. For the formal definition of the bisimulation quotient, see for example (Baier and Katoen 2008, Chapter 7).

2.2.4 Modal profiles of graphs

In this subsection we define the notion used to represent modal constraints in graphs: *modal profiles*.

Bounded bisimilarity types of nodes in a graph

Consider a node u in a graph G, and a non-negative integer n. The *n-bisimilarity type* of u in G, denoted $\mathsf{Type}^n_G(u)$, is the bisimilarity type of the n-step unravelling of G from u: $\mathsf{BisimType}(\mathsf{Unravel}(n, G, u))$.

Remark. The above notion coincides with the usual notion of n-bisimilarity: two rooted graphs $(G_1, u_1), (G_2, u_2)$ are n-bisimilar iff $\mathsf{Type}^n_{G_1}(u_1) = \mathsf{Type}^n_{G_2}(u_2)$. For a more standard definition of n-bisimilarity, see for example (Blackburn, Rijke, and Venema 2001, Chapter 2).

By inlining the definition of $\mathsf{BisimType}$ and the recursive characterization of $\mathsf{Unravel}$, we obtain an equivalent, recursive definition of Type^n_G. For $\tau = \mathsf{Type}^n_G(u)$, we get:

- $\mathsf{RootLabel}(\tau) = \mathsf{Label}_G(u)$;
- for each edge label a, $\mathsf{Subtypes}(\tau, a)$ is the set $\{\mathsf{Type}^{n-1}_G(v) : v \in \mathsf{Succ}_G(u, a)\}$ if $n \geq 1$, or the empty set otherwise.

We frequently use this recursive definition of $\mathsf{Type}^n_G(u)$.

Modal profiles

The *n-bisimilarity profile* of a graph G is the set of n-bisimilarity types of all nodes in G. In general, for a non-negative integer n, an *n-bisimilarity profile* is a finite set of n-bisimilarity types. An example of a graph and its 1-bisimilarity profile is shown in Figure 2.5 on the following page.

Remark. The notion of n-bisimilarity profiles coincides with the standard generalization of bisimilarity to transition systems without initial states: two graphs have equal n-bisimilarity profiles iff they are n-bisimilar when seen as labelled transition systems without initial states.

For a convenient representation of n-bisimilarity profiles as inputs to decision problems, we introduce *modal profiles*. A *modal profile* $\mathcal{S}$ consists of a **positive** integer n, called its *depth*, and an n-bisimilarity profile $\mathcal{T}$. A graph G *satisfies* $\mathcal{S}$, denoted $G \models \mathcal{S}$, if the n-bisimilarity profile of G is $\mathcal{T}$. We denote by $\Gamma_{\mathcal{S}}$ the set of all node labels appearing in $\mathcal{S}$, and by $\Sigma_{\mathcal{S}}$ the set of all edge labels appearing in $\mathcal{S}$. We do not consider modal profiles of depth 0 for convenience, to have the following property.

Fact 2.7. For a graph G and a modal profile $\mathcal{S}$, if $G \models \mathcal{S}$, then $\Gamma_G = \Gamma_{\mathcal{S}}$ and $\Sigma_G = \Sigma_{\mathcal{S}}$.

The fundamental decision problem for modal profiles is their *satisfiability*. We parametrize this problem by a class[1] C of graphs; the classes we consider correspond to different extensions of modal logic.

Satisfiability of a modal profile within a class C of graphs

Input: A modal profile $\mathcal{S}$.
Question: Is there a graph $G \in C$ that satisfies $\mathcal{S}$?

We refer to the above problem as the *satisfiability problem* for a class C of graphs. We define $\mathsf{Models}_C(\mathcal{S}) = \{G \in C : G \models \mathcal{S}\}$; the satisfiability problem can be equivalently stated as the question if $\mathsf{Models}_C(\mathcal{S})$ is non-empty. The *constructive satisfiability problem* for C is a computational problem with a modal profile $\mathcal{S}$ as the input, and an arbitrary graph $G \in \mathsf{Models}_C(\mathcal{S})$ as the output, if it exists.

[1] Formally, we could talk about *sets* of graphs instead of *classes*, because we defined graphs using the set $\mathbb{U}$.

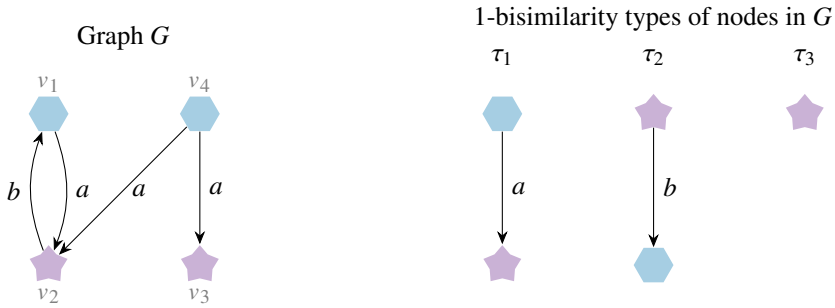


Figure 2.5: A graph G, and 1-bisimilarity types of its nodes: $\mathsf{Type}^1_G(v_1) = \mathsf{Type}^1_G(v_4) = \tau_1$, $\mathsf{Type}^1_G(v_2) = \tau_2$, $\mathsf{Type}^1_G(v_3) = \tau_3$. We draw the bisimilarity type of a tree as its representative, omitting node identifiers. The 1-bisimilarity profile of G is $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$. Note that the 1-bisimilarity type of a node tells us about the set of pairs (a, A) such that there is an outgoing edge with label a to a node with label A.

Computability

Thanks to the computability property shown below, instead of manually constructing modal profiles that satisfy some conditions, we will be able to iterate over all candidates and check if they satisfy the required conditions, which significantly simplifies talking about decidability and computability.

For a positive integer n and finite sets $\Gamma, \Sigma \subseteq \mathbb{U}$, let $\mathsf{Profiles}(n, \Gamma, \Sigma)$ denote the set of all modal profiles $\mathcal{S}$ of depth n such that $\Gamma_{\mathcal{S}} \subseteq \Gamma$ and $\Sigma_{\mathcal{S}} \subseteq \Sigma$.

Fact 2.8. The set $\mathsf{Profiles}(n, \Gamma, \Sigma)$ is finite, and the function $\mathsf{Profiles}$ is computable.

An algorithm computing $\mathsf{Profiles}(n, \Gamma, \Sigma)$ follows directly from the inductive definition of an n-bisimilarity type. Note that the size of $\mathsf{Profiles}(n, \Gamma, \Sigma)$ is exponential in the size of $\mathsf{Profiles}(n-1, \Gamma, \Sigma)$; this means that an upper bound on the size of $\mathsf{Profiles}(n, \Gamma, \Sigma)$ expressed in terms of n, $|\Gamma|$ and $|\Sigma|$ is an exponential tower expression of height n. This fact disqualifies the algorithms developed in the main part of this thesis from any possible applications. However, we emphasize that finiteness and computability of $\mathsf{Profiles}(n, \Gamma, \Sigma)$ is a crucial property of modal profiles that allows us to prove decidability results; for example, adjusting the definition of a modal profile to use *isomorphism* types of n-step unravellings instead of their bisimilarity types would result in infinite sets $\mathsf{Profiles}(n, \Gamma, \Sigma)$.

In Chapter 11 we will discuss a way to avoid such a high time complexity of the developed algorithms. The main idea behind it is to work with *n-node* unravellings instead of n-step unravellings; that is, connected subgraphs of n-step unravellings that contain the root and have at most n nodes. We do not use this idea in the main part of the thesis, because it introduces several technical challenges, which would significantly worsen the clarity of constructions and proofs. We also note that even the set of isomorphism types of n-node unravellings is finite for fixed sets of labels, which is discussed in Chapter 12 in relation to allowing counting in modal constraints (e.g. in the description logic $\mathcal{ALCQ}$).

2.3 Decision problems

In this thesis, we solve two problems defined in the setting of description logics:

- CQ containment modulo an $\mathcal{ALCO}$ TBox (as a warm-up, and to present the general strategy);
- UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox (the main result of this thesis).

The difference in difficulty between these two problems is significant, so we identify and solve some intermediate problems. We translate the problems to the graph setting, to avoid syntax-heavy definitions and to make it easier to state general, reusable lemmas.

All problems in the graph setting are parametrized by a class C of graphs. We solve the problems for several classes of graphs, iteratively introducing techniques used to prove the main result of the thesis. During this process, we want to minimize the need to repeat arguments and proofs for different classes. To this aim, we identify the core computational problem: *the minimal downsets problem*.

As a result of this approach, we describe multiple reductions between problems. For both query containment results, we solve the following problems, for different classes C of graphs:

1. query containment under modal constraints (in the setting of description logics);
2. *homomorphism entailment* under a modal profile, within C;
3. *homomorphism coverage* of a modal profile, within C;
4. *minimal downsets* of a modal profile, within C;
5. satisfiability of a modal profile, within C.

Each problem is reduced to the next one.

Let $\mathbb{G}$ denote the class of all graphs. At the end of this section we define problems (2) and (3) from the above list. The further plan is as follows:

- in Chapter 3 we reduce (1) – CQ containment modulo an $\mathcal{ALCO}$ TBox – to (2) for the class $\mathbb{G}$;
- in Chapter 4 we reduce (2) to (3) for $\mathbb{G}$;
- in Chapter 5 we reduce (3) to (4) for an arbitrary class of graphs;
- in Chapters 6 to 9 we reduce (4) to (5) and solve (5) for different classes of graphs;
- in Chapter 10 we reduce (2) to (3) for classes of "automata-annotated" graphs;
- in Chapter 11 we reduce (1) – UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox – to (2) for classes of "automata-annotated" graphs.

Homomorphism coverage

For a class C of graphs, a modal profile $\mathcal{S}$, and a set $\mathcal{H}$ of graphs, we write $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ if every graph in $\mathsf{Models}_C(\mathcal{S})$ admits a homomorphism from some graph in $\mathcal{H}$. A *counterexample* to $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ is a graph in $\mathsf{Models}_C(\mathcal{S})$ that does not admit such a homomorphism.

Fact 2.9. A counterexample to $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ exists iff $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ does not hold.

HOMOMORPHISM COVERAGE OF A MODAL PROFILE WITHIN A CLASS C OF GRAPHS

Input: A modal profile $\mathcal{S}$ and a set $\mathcal{H}$ of graphs.

Question: Does $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$?

We refer to the above problem as the *homomorphism coverage problem* for a class C of graphs.

For the class $\mathbb{G}$ of all graphs, intuitively, the set $\mathcal{H}$ of graphs can be seen as a set of Boolean conjunctive queries, the satisfaction of a query corresponds to the existence of a graph homomorphism, and the modal profile $\mathcal{S}$ represents modal constraints. This interpretation makes the homomorphism coverage problem very closely related to the finite query entailment problem in the field of knowledge representation and reasoning.

In later chapters we will define classes of graphs that correspond to more expressive queries and modal constraints; for example, we might consider the class of graphs in which edges with some fixed label Reachable encode the reachability relation in the underlying graph. Intuitively, this class of graphs corresponds to conjunctive queries that can additionally express reachability between two elements, and the description logic $\mathcal{ALCO}$ extended with reachability.

Homomorphism entailment

Below we define the *homomorphism entailment problem*, which can be seen as a close analogue of the problem of query containment under modal constraints in the graph setting: as mentioned earlier, in Chapter 3 we show a reduction from the problem of CQ containment modulo an $\mathcal{ALCO}$ TBox to the homomorphism entailment problem for $\mathbb{G}$. The details of this reduction influence the definition of the homomorphism entailment problem in two significant ways.

- In the setting of description logics, queries and TBoxes might refer to *individual names*, which correspond to single elements of interpretations. Because we need to take them into account in the reduction, in the homomorphism entailment problem we work with a set Γ_N of node labels, called *nominals*, and with graphs that have exactly one node with label A for each $A \in \Gamma_N$.
- Since all modal profiles constructed in the reduction have depth 1, we include this condition in the definition of the problem; it simplifies later reductions to the homomorphism coverage problem.

A *shallow modal profile* is a modal profile of depth 1. A *modal profile with nominals* $\tilde{\mathcal{S}}$ is a modal profile $\mathcal{S}$ with a set $\Gamma_N \subseteq \Gamma_{\mathcal{S}}$ of *nominals*. A graph is called Γ_N-*unique* if it contains exactly one node with label A for each $A \in \Gamma_N$; then, these nodes are called *nominal nodes*. A graph G *satisfies* $\tilde{\mathcal{S}}$, denoted $G \models \tilde{\mathcal{S}}$, if G satisfies $\mathcal{S}$ and is Γ_N-unique. For a class C of graphs, we define $\mathsf{Models}_C(\tilde{\mathcal{S}}) = \{G \in C : G \models \tilde{\mathcal{S}}\}$.

For a class C of graphs, a shallow modal profile with nominals $\tilde{\mathcal{S}}$, and sets $\mathcal{G}, \mathcal{H}$ of graphs, we write $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\text{hom}} \mathcal{H}$ if, for every graph $D \in \mathsf{Models}_C(\tilde{\mathcal{S}})$, if D admits a homomorphism from some graph in $\mathcal{G}$, then D admits a homomorphism from some graph in $\mathcal{H}$. A *counterexample* to $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\text{hom}} \mathcal{H}$ is a graph $D \in \mathsf{Models}_C(\tilde{\mathcal{S}})$ that admits a homomorphism from some graph in $\mathcal{G}$, but does not admit a homomorphism from any graph in $\mathcal{H}$.

Fact 2.10. A counterexample to $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\text{hom}} \mathcal{H}$ exists iff $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\text{hom}} \mathcal{H}$ does not hold.

HOMOMORPHISM ENTAILMENT UNDER A MODAL PROFILE WITHIN A CLASS C OF GRAPHS

Input: A shallow modal profile with nominals $\tilde{\mathcal{S}}$ and two sets $\mathcal{G}, \mathcal{H}$ of graphs.

Question: Does $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\text{hom}} \mathcal{H}$?

We refer to the above problem as the *homomorphism entailment problem* for a class C of graphs.

Chapter 3

Query containment under constraints

Recall that $\mathbb{G}$ denotes the class of all graphs. In this chapter we prove the following proposition.

Proposition 3.1. There is a Turing reduction from the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox to the homomorphism entailment problem for $\mathbb{G}$.

The main purpose of this chapter is to convince the reader that the homomorphism entailment problem can be treated as a variant of the problem of query containment under modal constraints. If the reader does not need convincing, we recommend skipping this chapter; even though the translation is relatively straightforward, the syntactic nature of description logics and conjunctive queries complicates the constructions.

For readers familiar with description logics and conjunctive queries, below we present the main ideas of the translation.

- Finite interpretations can be treated as graphs: role names correspond directly to edge labels, node labels are sets of concept names and individual names.
- Conjunctive queries can be assumed to be Boolean, because we can replace answer variables with individual names. In graphs corresponding to Boolean conjunctive queries, the nodes loosely correspond to terms in the query (some terms are identified).
- The $\mathcal{ALCO}$ TBox can be "normalized" by introducing a fresh concept name for each complex sub-concept and rewriting concept inclusions such that each concept description uses at most one constructor. Normalized TBoxes correspond to shallow modal profiles with nominals.

3.1 Definitions

Recall that at the beginning of Chapter 2 we defined an infinite, computable set $\mathbb{U}$, which we use in the definitions to ensure that the corresponding objects can be encoded as inputs of decision problems.

3.1.1 Interpretations

Let ConceptNames, RoleNames, IndividualNames and Variables be disjoint subsets of $\mathbb{U}$ that are infinite and computable; for example, we can set each of them to $\{(i, x) : x \in \mathbb{U}\}$ for different integers i. In terms of first-order interpretations, concept names correspond to unary predicates, role names correspond to binary predicates, and individual names correspond to constants – functional predicates of arity zero.

An *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ consists of a non-empty set Δ, called the *domain*, and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps each concept name to a subset of Δ, each role name to a relation on Δ, and each individual name to an element of Δ. An interpretation is *finite* if Δ is finite.

Remark. In description logics, it is sometimes assumed that different individual names are mapped to different elements of the domain (*unique name assumption*); we do not assume this.

We define the notions for potentially infinite interpretations, as this is standard in the literature, but we actually work exclusively with finite ones. For this reason, and to simplify the translation from interpretations to graphs, let us assume that domains of interpretations are subsets of $\mathbb{U}$. We acknowledge that this assumption is non-standard – in particular, it precludes uncountable interpretations – but it does not affect the problems we consider.

3.1.2 Conjunctive queries

A *term* is a variable or an individual name. A *concept atom* is a formula of the form $A(x)$, where A is a concept name and x is a term. A *role atom* is a formula of the form $r(x, y)$, where r is a role name and x, y are terms. An *equality atom* is a formula of the form $x = y$, where x, y are terms. An *atom* is a concept atom, a role atom, or an equality atom.

Remark. Formally, we treat a *formula* as a finite sequence of elements from $\mathbb{U}$. When defining a formula, we assume that symbols that are not explicitly defined (e.g. the equality sign) are treated as logical symbols, and we implicitly assume their unambiguous encoding as elements of $\mathbb{U}$.

A *conjunctive query* (CQ) consists of:

- a set Φ of atoms;
- a finite sequence $\vec{x}$ of variables, called *answer variables* (or *distinguished*, *free*, or *head variables*).

For a CQ $q = (\vec{x}, \Phi)$, we denote by $\mathsf{Terms}(q)$ the set of all terms that occur in atoms in Φ, and by $\mathsf{Var}(q)$ all variables among them. The length of $\vec{x}$ is called the *arity* of q.

Remark. Conjunctive queries are typically assumed to be *domain-independent* (also called *safe*); this corresponds to a requirement that each answer variable occurs as a term in some concept atom or role atom in Φ. We omit this assumption; in some sense, domain-independence is built into the definition of answers to queries in interpretations, presented below.

For an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ and a CQ $q = (\vec{x}, \Phi)$, a *variable assignment* is a function μ from $\mathsf{Var}(q)$ to Δ. Let us extend μ to individual names, with $\mu(a) = a^{\mathcal{I}}$ for each $a \in \mathsf{IndividualNames}$. The CQ q *evaluates to "true"* in $\mathcal{I}$ under μ if:

- for each concept atom $A(x)$ in Φ, $\mu(x) \in A^{\mathcal{I}}$;

- for each role atom $r(x, y)$ in Φ, $(\mu(x), \mu(y)) \in r^{\mathcal{I}}$;
- for each equality atom $x = y$ in Φ, $\mu(x) = \mu(y)$.

Then, for $\vec{x} = (x_1, \ldots, x_k)$, the sequence $(\mu(x_1), \ldots, \mu(x_k))$ is called an *answer* to q in $\mathcal{I}$. We denote by $q(\mathcal{I})$ the set of all answers to q in $\mathcal{I}$.

Remark. We work under *set semantics*, with sets of answers to queries, as opposed to *bag semantics*, where bags (multisets) of answers are considered, with multiplicities depending on the number of "truthful" variable assignments.

A conjunctive query is *Boolean* if it has arity 0 ($\vec{x}$ is the empty sequence). A Boolean CQ q is *satisfied* in an interpretation $\mathcal{I}$ if q evaluates to "true" in $\mathcal{I}$ under some variable assignment μ; equivalently, if $q(\mathcal{I})$ is the singleton of the empty sequence.

Standard representations

A CQ $q = (\vec{x}, \Phi)$ is often written in the form of a Datalog rule:

$$q(x_1, \ldots, x_k) \leftarrow \varphi_1 \wedge \cdots \wedge \varphi_r,$$

where $\Phi = \{\varphi_1, \ldots, \varphi_r\}$; the order of atoms and their potential repetitions do not matter. It can also be written similarly to a first-order formula:

$$(x_1, \ldots, x_k).\, \exists_{y_1, \ldots, y_\ell}\ \varphi_1 \wedge \cdots \wedge \varphi_r,$$

where $\vec{x} = (x_1, \ldots, x_k)$, $\Phi = \{\varphi_1, \ldots, \varphi_r\}$, and $\vec{y} = (y_1, \ldots, y_\ell)$ is a sequence consisting of all variables that appear as terms in atoms in Φ and are not answer variables.

Conjunctive query containment

A CQ q_1 is *contained* in a CQ q_2, denoted $q_1 \subseteq q_2$, if $q_1(\mathcal{I}) \subseteq q_2(\mathcal{I})$ for all finite interpretations $\mathcal{I}$.

Conjunctive query containment

Input: Conjunctive queries q_1, q_2 of equal arity.
Question: Does $q_1 \subseteq q_2$?

By a famous result by Chandra and Merlin (1977), the problem of CQ containment is NP-complete, and is equivalent to the problem of finding a *homomorphism* between CQs, which is a notion closely related to graph homomorphisms.

3.1.3 Description logics

In description logics, the interpretation function in an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is extended to *concept descriptions*, mapping each of them to a subset of Δ. Each description logic comes with *basic concept descriptions* and a set of *constructors*. Concept descriptions are formulas constructed inductively from basic formulas using constructors, and the corresponding extension of an interpretation function is also defined inductively. Formally, the set of concept descriptions is the smallest set that contains the basic concept descriptions (e.g. concept names, or $\bot$ interpreted as the empty set), and formulas constructed inductively from other concept descriptions (e.g. $C \sqcup D$ for concept descriptions C, D, interpreted as

$C^{\mathcal{I}} \cup D^{\mathcal{I}}$). We provide these definitions in the form of two tables, listing the basic formulas and the constructors, along with their interpretations.

The description logic $\mathcal{ALC}$

In the description logic $\mathcal{ALC}$ (*Attributive Concept Language with Complements*), the set of concept descriptions is given by Table 3.1 and Table 3.2.

Basic concept description C	**Interpretation** $C^{\mathcal{I}}$
$\bot$	$\emptyset$
$\top$	Δ
A	$A^{\mathcal{I}}$

Table 3.1: Basic concept descriptions in $\mathcal{ALC}$ and their interpretations in $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, for every concept name A.

Constructor of a concept description C	**Interpretation** $C^{\mathcal{I}}$
$\neg D$	$\Delta \setminus D^{\mathcal{I}}$
$D \sqcap E$	$D^{\mathcal{I}} \cap E^{\mathcal{I}}$
$D \sqcup E$	$D^{\mathcal{I}} \cup E^{\mathcal{I}}$
$\exists r.D$	$\{x \in \Delta : \exists_{y \in \Delta} (x, y) \in r^{\mathcal{I}} \wedge y \in D^{\mathcal{I}}\}$
$\forall r.D$	$\{x \in \Delta : \forall_{y \in \Delta} (x, y) \in r^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}}\}$

Table 3.2: Constructors in $\mathcal{ALC}$ and the interpretations of the resulting concept descriptions in $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, for all concept descriptions D, E and role names r.

The description logic $\mathcal{ALC}$ can be seen as a syntactic variant of the basic modal logic with multiple modalities (*multi-modal logic K*), as observed by Schild (1991), with Kripke structures corresponding to interpretations, concept names to propositional variables, and role names to modal parameters: there is a direct translation between concept descriptions in $\mathcal{ALC}$ and formulas in multi-modal K; see also (van Harmelen, Lifschitz, and Porter 2007, Chapter 3).

Redundant constructors

To simplify working with concept descriptions, we use the fact that some basic concept descriptions and constructors can be replaced with others, resulting in the same interpretation:

- $\top$ has the same interpretation as $\neg\bot$;
- $D \sqcap E$ has the same interpretation as $\neg((\neg D) \sqcup (\neg E))$;
- $\forall r.D$ has the same interpretation as $\neg(\exists r.(\neg D))$.

Thus, we can rewrite each concept description in $\mathcal{ALC}$ to use only basic concept descriptions $\bot, A$ and constructors $\neg D$, $D \sqcup E$ and $\exists r.D$, while preserving its interpretation. In the rest of this chapter we assume that all concept descriptions are of this form.

Terminological boxes

A *terminological box* (TBox) in a description logic is a finite set of *general concept inclusions*: formulas of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ *satisfies* a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$.

For convenience, we also allow a TBox to contain *general concept equivalences*: formulas of the form $C \equiv D$, which are interpreted as two general concept inclusions, $C \sqsubseteq D$ and $D \sqsubseteq C$.

Nominals

The description logic $\mathcal{ALCO}$ extends $\mathcal{ALC}$ with *nominals*: for each individual name a, the nominal $\{a\}$ is a basic concept description, interpreted in $\mathcal{I}$ as the singleton $\{a^{\mathcal{I}}\}$. The counterpart of $\mathcal{ALCO}$ in the realm of modal logic is the basic *hybrid logic*, from which the name *nominals* comes.

Remark. We noted earlier that we work without the *unique name assumption*, which says that an interpretation maps each individual name to a different element. An $\mathcal{ALCO}$ TBox can express this condition for two individual names a, b by the general concept inclusion $\{a\} \sqcap \{b\} \sqsubseteq \bot$.

3.1.4 Query containment under modal constraints

For CQs q_1, q_2 and an $\mathcal{ALCO}$ TBox $\mathcal{T}$, we say that q_1 is contained in q_2 *modulo* $\mathcal{T}$, denoted $q_1 \subseteq^{\mathcal{T}} q_2$, if $q_1(\mathcal{I}) \subseteq q_2(\mathcal{I})$ for every finite interpretation $\mathcal{I}$ that satisfies $\mathcal{T}$.

Conjunctive query containment modulo an $\mathcal{ALCO}$ TBox
Input: An $\mathcal{ALCO}$ TBox $\mathcal{T}$ and two CQs q_1, q_2 of equal arity.
Question: Does $q_1 \subseteq^{\mathcal{T}} q_2$?

Finite signatures

By a *finite signature* we mean a triple $X = (X_{\mathrm{C}}, X_{\mathrm{R}}, X_{\mathrm{I}})$ of finite sets $X_{\mathrm{C}} \subseteq \mathsf{ConceptNames}$, $X_{\mathrm{R}} \subseteq \mathsf{RoleNames}$, and $X_{\mathrm{I}} \subseteq \mathsf{IndividualNames}$. For a finite signature X, we define conjunctive queries *over* X and $\mathcal{ALCO}$ TBoxes *over* X as ones that use only symbols from X: concept names in X_{C}, role names in X_{R}, and individual names in X_{I}.

For an instance $\mathcal{T}, q_1, q_2$ of the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox, consider the finite signature X consisting of all symbols that appear in the input: the set X_{C} of concept names that appear in at least one of $\mathcal{T}, q_1, q_2$, the set X_{R} of role names that appear in at least one of $\mathcal{T}, q_1, q_2$, and the set X_{I} of all individual names that appear in at least one of $\mathcal{T}, q_1, q_2$.

Then, q_1 and q_2 are CQs over X, and $\mathcal{T}$ is an $\mathcal{ALCO}$ TBox over X. For an interpretation $\mathcal{I}$, the definition of answers to CQs in $\mathcal{I}$ and the definition of $\mathcal{I}$ satisfying $\mathcal{T}$ rely only on interpretations of symbols in X. This is crucial – along with the fact that we consider only finite interpretations – as this will allow us to translate the problem to the graph setting, where everything is finite. We also use this fact in two reductions below, which show that we can assume, without loss of generality, that q_1 and q_2 are Boolean, and that $\mathcal{T}$ has a certain simple structure.

Boolean CQs

Consider an instance of the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox: two CQs q_1, q_2 of equal arity r, and an $\mathcal{ALCO}$ TBox $\mathcal{T}$. The idea is to replace answer variables with individual names.

Let X_{I} be the set of all individual names that occur in at least one of q_1, q_2 and $\mathcal{T}$. Let $\vec{a} = (a_1, \ldots, a_r)$ be an arbitrary sequence of length r of different individual names in $\mathsf{IndividualNames} \setminus X_{\mathrm{I}}$.

For $k \in \{1, 2\}$, consider $q_k = ((x_1, \ldots, x_r), \Phi)$. Intuitively, we want to replace each answer variable x_i by the individual name a_i. Formally, because a variable might occur multiple times as an answer variable, we need to be more careful. Let $f(i)$ denote the smallest index j such that x_j is the same variable as x_i. We define q'_k as the Boolean conjunctive query obtained from q_k by:

- replacing each occurrence of x_i in atoms in Φ with $a_{f(i)}$, for each $i \in \{1, \ldots, r\}$;
- adding to Φ an atom $a_i = a_j$ for each $i, j \in \{1, \ldots, r\}$ such that x_i is the same variable as x_j;
- replacing the sequence of answer variables with the empty sequence.

Fact 3.2. The answer to the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox is the same for $\mathcal{T}, q_1, q_2$ and for $\mathcal{T}, q'_1, q'_2$.

It is easy to compute such queries q'_1, q'_2, given q_1, q_2 and $\mathcal{T}$. Thus, when solving the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox, we can assume that the queries are Boolean, without loss of generality.

Shallow TBoxes

Because concept descriptions are defined inductively, they might have a deeply nested structure. To simplify the translation to the graph setting, we show that one can construct an "equivalent" TBox that uses only concept descriptions with a shallow structure, using at most one constructor.

An $\mathcal{ALCO}$ TBox is *shallow* if it contains only:

- general concept inclusions of the form $A \sqsubseteq B$ for concept names A, B;
- general concept equivalences of the form $A \equiv C$ for a concept name A and a concept description C that uses at most one constructor; that is, C is of the form $\bot$, $\{a\}$, B, $\neg B$, $B_1 \sqcup B_2$ or $\exists r.B$, where B, B_1, B_2 are concept names, a is an individual name, and r is a role name.

For an $\mathcal{ALCO}$ concept description C, let $\mathsf{sub}(C)$ be the set of subformulas of C; that is, the set of concept descriptions defined recursively as follows:

- if C is a basic concept description, $\mathsf{sub}(C) = \{C\}$;
- if C is of the form $\neg D$ or $\exists r.D$, $\mathsf{sub}(C) = \{C\} \cup \mathsf{sub}(D)$;
- if C is of the form $D \sqcup E$, $\mathsf{sub}(C) = \{C\} \cup \mathsf{sub}(D) \cup \mathsf{sub}(E)$.

For an $\mathcal{ALCO}$ TBox $\mathcal{T}$, let $\mathsf{sub}(\mathcal{T})$ denote the union of $\mathsf{sub}(C)$ for all concept descriptions that appear in $\mathcal{T}$.

Consider a finite signature X and an $\mathcal{ALCO}$ TBox $\mathcal{T}$ over X. Below we construct a shallow $\mathcal{ALCO}$ TBox $\mathcal{T}'$ that, intuitively, imposes the same constraints as $\mathcal{T}$ on interpretations of concept names in X_{C}. Formally, we will use the following notion; two interpretations $\mathcal{I}, \mathcal{J}$ *agree on* X if they have equal

domains, and:

- for each $r \in X_{\mathrm{R}}$, $r^{\mathcal{I}} = r^{\mathcal{J}}$;
- for each $A \in X_{\mathrm{C}}$, $A^{\mathcal{I}} = A^{\mathcal{J}}$;
- for each $a \in X_{\mathrm{I}}$, $a^{\mathcal{I}} = a^{\mathcal{J}}$.

For each $C \in \mathsf{sub}(\mathcal{T})$, let A_C be an arbitrary concept name in $\mathsf{ConceptNames} \setminus X_{\mathrm{C}}$, such that $A_C \neq A_D$ whenever $C \neq D$. Let $\mathcal{T}'$ be the shallow $\mathcal{ALCO}$ TBox containing the following general concept inclusions and general concept equivalences:

- $A_C \sqsubseteq A_D$ for each general concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$;
- $A_C \equiv A_D$ for each general concept equivalence $C \equiv D$ in $\mathcal{T}$;
- for each concept description C in $\mathsf{sub}(\mathcal{T})$:
  - $A_C \equiv C$ if C is a basic concept description;
  - $A_C \equiv \neg A_D$ if C is of the form $\neg D$;
  - $A_C \equiv A_D \sqcup A_E$ if C is of the form $D \sqcup E$;
  - $A_C \equiv \exists r.A_D$ if C is of the form $\exists r.D$.

Fact 3.3. For every interpretation $\mathcal{I}$, the following are equivalent:

- $\mathcal{I} \models \mathcal{T}$;
- $\mathcal{I}' \models \mathcal{T}'$ for some interpretation $\mathcal{I}'$ that agrees with $\mathcal{I}$ on X.

It is easy to compute such a shallow $\mathcal{ALCO}$ TBox $\mathcal{T}'$ from $\mathcal{T}$ and X_{C}. Thus, when solving the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox, we can assume that the TBox is shallow, without loss of generality.

3.2 The reduction to homomorphism entailment

In this section we present a translation between the interpretation setting and the graph setting. Elements of the domain correspond to nodes, role names correspond to edge labels, and concept names and individual names are encoded in node labels.

Graph signatures

For a finite signature $X = (X_{\mathrm{C}}, X_{\mathrm{R}}, X_{\mathrm{I}})$, an *$X$-graph* is a non-empty graph in which node labels are subsets of $X_{\mathrm{C}} \cup X_{\mathrm{I}}$ and edge labels are in X_{R}. An X-graph G *respects individual names* if, for each individual name $a \in X_{\mathrm{I}}$, there is exactly one node v in G such that $a \in \mathsf{Label}_G(v)$.

3.2.1 Interpretations to graphs

For a finite signature X and a finite interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, we define $\mathsf{InterpretationToGraph}_X(\mathcal{I})$ as the X-graph G with nodes Δ such that:

- for each role name $r \in X_{\mathrm{R}}$, there is an edge (u, r, v) in G iff $(u, v) \in r^{\mathcal{I}}$;
- for each concept name $A \in X_{\mathrm{C}}$, $A \in \mathsf{Label}_G(v)$ iff $v \in A^{\mathcal{I}}$;
- for each individual name $a \in X_{\mathrm{I}}$, $a \in \mathsf{Label}_G(v)$ iff $v = a^{\mathcal{I}}$.

Note that G respects individual names.

For a finite signature X and an X-graph G respecting individual names, let $\mathsf{GraphToInterpretation}_X(G)$ be an arbitrary fixed interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ such that $\Delta = V_G$ and:

- $r^{\mathcal{I}} = \{(u, v) : (u, r, v) \in E_G\}$ for each role name $r \in X_{\mathrm{R}}$;
- $A^{\mathcal{I}} = \{v \in V_G : A \in \mathsf{Label}_G(v)\}$ for each concept name $A \in X_{\mathrm{C}}$;
- $a^{\mathcal{I}}$ is the unique node v in G such that $a \in \mathsf{Label}_G(v)$, for each individual name $a \in X_{\mathrm{I}}$.

The interpretation of concept names, role names and individual names that do not occur in X can be arbitrary.

Fact 3.4. For a finite signature X and an X-graph G that respects individual names, G is equal to $\mathsf{InterpretationToGraph}_X(\mathsf{GraphToInterpretation}_X(G))$.

Fact 3.5. For a finite signature X and a finite interpretation $\mathcal{I}$, the interpretation $\mathcal{I}$ agrees on X with $\mathsf{GraphToInterpretation}_X(\mathsf{InterpretationToGraph}_X(\mathcal{I}))$.

3.2.2 Conjunctive queries to graphs

A graph G is a *quotient* of a CQ q if the set of nodes in G forms a partition of $\mathsf{Terms}(q)$; that is, the nodes in G are non-empty subsets of $\mathsf{Terms}(q)$ such that each term t is contained in exactly one subset, denoted by $[t]_G$.

For a finite signature X and a non-empty Boolean CQ q, we define $\mathsf{CQToGraphs}_X(q)$ as the set of all X-graphs G that are quotients of q such that:

- for each role atom $r(x, y)$ in q, $([x]_G, r, [y]_G)$ is an edge in G;
- for each concept atom $A(x)$ in q, $A \in \mathsf{Label}_G([x]_G)$;
- for each individual name a in q, $a \in \mathsf{Label}_G([a]_G)$;
- for each equality atom $x = y$ in q, $[x]_G = [y]_G$.

Note that $\mathsf{CQToGraphs}_X(q)$ is the set of all X-graphs that satisfy the conditions above: in particular, node labels might contain other elements of $X_{\mathrm{C}} \cup X_{\mathrm{I}}$. Note also that the function $\mathsf{CQToGraphs}_X$ is computable, as the set of all partitions of $\mathsf{Terms}(q)$ is easily computable, and the graphs use only labels specified by the finite signature X.

For the empty Boolean CQ q, we define $\mathsf{CQToGraphs}_X(q)$ as the singleton of the empty graph.

Fact 3.6. For a finite signature X, a Boolean CQ q over X, a finite interpretation $\mathcal{I}$ and an X-graph G that respects individual names:

- if q is satisfied in $\mathcal{I}$, then $G_q \to \mathsf{InterpretationToGraph}_X(\mathcal{I})$ for some $G_q \in \mathsf{CQToGraphs}_X(q)$;
- if $G_q \to G$ for some $G_q \in \mathsf{CQToGraphs}_X(q)$, then q is satisfied in $\mathsf{GraphToInterpretation}_X(G)$.

3.2.3 TBoxes to modal profiles with nominals

For a finite signature X and a shallow $\mathcal{ALCO}$ TBox $\mathcal{T}$ over X, we define $\mathsf{TBoxToProfiles}_X(\mathcal{T})$ as the set of all shallow modal profiles with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$ such that, for $\mathcal{S} = (1, \Theta)$ and $\hat{A}_\tau = \mathsf{RootLabel}(\tau)$ for each $\tau \in \Theta$:

- Θ is non-empty (domains of interpretations are non-empty by definition, so we need to exclude the empty graph);
- for each $\tau \in \Theta$, $\mathsf{Representative}(\tau)$ is an X-graph;

- for each individual name $a \in X_\mathrm{I}$ there is exactly one $\tau \in \Theta$ with $a \in \hat{A}_\tau$;
- Γ_N is the set of node labels $\hat{A}_\tau$ for all $\tau \in \Theta$ such that $\hat{A}_\tau \cap X_\mathrm{I} \neq \emptyset$;
- for each $\tau \in \Theta$, for each $A_C \sqsubseteq A_D$ in $\mathcal{T}$, if $A_C \in \hat{A}_\tau$ then $A_D \in \hat{A}_\tau$;
- for each $\tau \in \Theta$, for each $A_C \equiv C$ in $\mathcal{T}$:
  - if $C = \bot$, then $A_C \notin \hat{A}_\tau$;
  - if $C \in \mathsf{ConceptNames}$, then $A_C \in \hat{A}_\tau$ iff $C \in \hat{A}_\tau$;
  - if C is of the form $\{a\}$, then $A_C \in \hat{A}_\tau$ iff $a \in \hat{A}_\tau$;
  - if C is of the form $\neg A_D$, then $A_C \in \hat{A}_\tau$ iff $A_D \notin \hat{A}_\tau$;
  - if C is of the form $A_D \sqcup A_E$, then $A_C \in \hat{A}_\tau$ iff $A_D \in \hat{A}_\tau$ or $A_E \in \hat{A}_\tau$;
  - if C is of the form $\exists r.A_D$, then $A_C \in \hat{A}_\tau$ iff there is some $\tau' \in \mathsf{Subtypes}(\tau, r)$ such that $A_D \in \mathsf{RootLabel}(\tau')$.

The function $\mathsf{TBoxToProfiles}$ is computable, because we can compute the set of all shallow modal profiles $\mathcal{S}$ that use only labels determined by the finite signature X (by Fact 2.8 on page 30), consider all $\Gamma_\mathrm{N} \subseteq \Gamma_\mathcal{S}$, and verify the conditions listed above.

Fact 3.7. For a finite signature X, a shallow $\mathcal{ALCO}$ TBox $\mathcal{T}$ over X, a finite interpretation $\mathcal{I}$ and a graph G:

- if $\mathcal{I} \models \mathcal{T}$, then $\mathsf{InterpretationToGraph}_X(\mathcal{I}) \models \tilde{\mathcal{S}}$ for some $\tilde{\mathcal{S}} \in \mathsf{TBoxToProfiles}_X(\mathcal{T})$;
- if $G \models \tilde{\mathcal{S}}$ for some $\tilde{\mathcal{S}} \in \mathsf{TBoxToProfiles}_X(\mathcal{T})$, then G is an X-graph that respects individual names and $\mathsf{GraphToInterpretation}_X(G) \models \mathcal{T}$.

3.2.4 The reduction

Recall that $\mathbb{G}$ denotes the class of all graphs.

Consider an instance $\mathcal{T}, q_1, q_2$ of the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox. As discussed earlier, without loss of generality we can assume that $\mathcal{T}$ is shallow and that q_1, q_2 are Boolean queries. Let X be the finite signature consisting of all symbols in q_1, q_2 and $\mathcal{T}$.

From Fact 3.6 and Fact 3.7 we get the following.

Proposition 3.8. The following conditions are equivalent:

- $q_1 \subseteq^\mathcal{T} q_2$;
- $\mathsf{CQToGraphs}_X(q_1) \models_\mathrm{hom}^{\mathbb{G}, \tilde{\mathcal{S}}} \mathsf{CQToGraphs}_X(q_2)$ for each $\tilde{\mathcal{S}}$ in $\mathsf{TBoxToProfiles}_X(\mathcal{T})$.

3.3 Summary

We have shown a Turing reduction from the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox to the homomorphism entailment problem for the class $\mathbb{G}$ of all graphs. For the main result of the thesis, we will describe an almost identical reduction for unions of conjunctive regular path queries and the description logic $\mathcal{ALCO}_\mathrm{reg}$.

Chapter 4

Homomorphism entailment

Recall that $\mathbb{G}$ denotes the class of all graphs. In this chapter we prove the following proposition, under one assumption, which we will prove later (in Chapter 6), that the constructive satisfiability problem for $\mathbb{G}$ is computable; that is, there is an algorithm that computes some model of a given modal profile $\mathcal{S}$, if it exists. While this assumption is not strictly necessary, it makes some proofs much simpler.

Proposition 4.1. There is a Turing reduction from the homomorphism entailment problem for $\mathbb{G}$ to the homomorphism coverage problem for $\mathbb{G}$.

Recall the notions of counterexamples corresponding to these two problems.

- For an instance $\mathcal{S}, \mathcal{H}$ of the homomorphism coverage problem for a class $\mathcal{C}$ of graphs, a *counterexample* to $\mathcal{H} \twoheadrightarrow \mathsf{Models}_{\mathcal{C}}(\mathcal{S})$ is a graph in $\mathsf{Models}_{\mathcal{C}}(\mathcal{S})$ that does not admit a homomorphism from any graph in $\mathcal{H}$.
- For an instance $\tilde{\mathcal{S}}$, $\mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for a class $\mathcal{C}$ of graphs, a *counterexample* to $\mathcal{G} \models_{\mathrm{hom}}^{\mathcal{C}, \tilde{\mathcal{S}}} \mathcal{H}$ is a graph in $\mathsf{Models}_{\mathcal{C}}(\tilde{\mathcal{S}})$ that admits a homomorphism from some graph in $\mathcal{G}$, but does not admit a homomorphism from any graph in $\mathcal{H}$.

The high-level plan of the reduction is as follows. For an instance $\tilde{\mathcal{S}}, \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem, with $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$, instead of considering all graphs D that satisfy $\tilde{\mathcal{S}}$ and admit a homomorphism h from some graph in $\mathcal{G}$, we "remove" from the graph D all nodes in the image of h and all nominal nodes (nodes with labels in Γ_{N}). For an appropriate notion of "removal" of nodes, we will show that D is a counterexample to the homomorphism entailment problem iff the remaining graph is a counterexample to some instance of the homomorphism coverage problem, in which the modal profile $\mathcal{S}$ and the graphs $\mathcal{H}$ are, intuitively, distributed between the removed nodes and the remaining graph.

We define the notion of *detachments* of graphs, which allows us to formalize the "removal" of nodes; the same notion will be used in an analogous reduction for the main result of this thesis, in Chapter 10.

4.1 Definitions

4.1.1 Detachments

Consider a graph D and a subset V of its nodes. We define two graphs: D_{inner}, which is the subgraph of D induced by V; and D_{outer}, obtained from D by removing all nodes in V and modifying node labels as follows. For each node u, we change its label A to $(A, \mathsf{ENn}, \mathsf{EOut})$, where:

- ENn is the set of pairs (v, a) such that $v \in V$ and (v, a, u) is an edge in D;
- EOut is the set of pairs (a, v) such that $v \in V$ and (u, a, v) is an edge in D.

Intuitively, the sets ENn and EOut represent all edges in D between the node u and nodes in V. The pair $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ is called the *V-detachment* of D; we refer to the graph D_{inner} as the *inner graph*, and to D_{outer} as the *outer graph*. A *detachment* of D is its V-detachment for some $V \subseteq V_D$. We tend to use the letter u to denote a node in the outer graph, v to denote a node in the inner graph, and w to denote a node in D. An example of a graph and its detachment is shown in Figure 4.1.

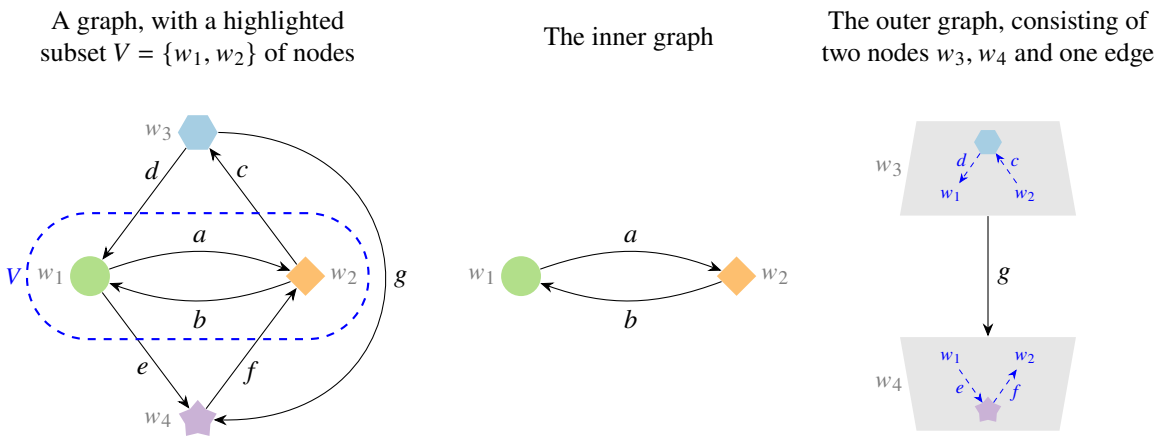


Figure 4.1: An example of a graph and its V-detachment for $V = \{w_1, w_2\}$. In the outer graph, we represent node labels $(A, \mathsf{ENn}, \mathsf{EOut})$ as trapezoids containing the shape of the label A, and incoming and outgoing dashed edges representing the sets ENn and EOut: the label of w_3 is $(\hexagon, \{(w_2, c)\}, \{(d, w_1)\})$, and the label of w_4 is $(\pentagon, \{(w_1, e)\}, \{(f, w_2)\})$.

4.1.2 Attachments

Below we define a reverse operation $\mathsf{AttachGraphs}$, for every pair of graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ with appropriate node labels.

For a graph D_{inner}, an *outer node label* for D_{inner} is a node label of the form $(A, \mathsf{ENn}, \mathsf{EOut})$, where:

- ENn is a finite set of pairs of the form (v, a), where v is a node in D_{inner} and a is an edge label;
- EOut is a finite set of pairs of the form (a, v), where v is a node in D_{inner} and a is an edge label.

For finite sets $\Gamma, \Sigma \subseteq \mathbb{U}$, let $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma, \Sigma)$ be the set of all outer node labels $(A, \mathsf{ENn}, \mathsf{EOut})$ for D_{inner} such that $A \in \Gamma$ and $a \in \Sigma$ for all $(v, a) \in \mathsf{ENn}$ and all $(a, v) \in \mathsf{EOut}$. Note that the function $\mathsf{OuterLabels}$ is computable.

For disjoint graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ such that each node label in D_{outer} is an outer node label for D_{inner},

the *attachment* of $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, denoted $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, is the graph obtained from the union of D_{inner} and D_{outer} by:

- adding an edge (v, a, u) for each node u in D_{outer} with label $(A, \mathsf{EIn}, \mathsf{EOut})$ such that $(v, a) \in \mathsf{EIn}$;
- adding an edge (u, a, v) for each node u in D_{outer} with label $(A, \mathsf{EIn}, \mathsf{EOut})$ such that $(a, v) \in \mathsf{EOut}$;
- replacing the label $(A, \mathsf{EIn}, \mathsf{EOut})$ with A for each node in D_{outer}.

The construction can be easily generalized to non-disjoint graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ by using their disjoint union instead of union, and, instead of u and v, referring to corresponding nodes in the disjoint union. To avoid cluttering the notation, we implicitly assume that the considered graphs are always disjoint.

Fact 4.2. For a graph D and its detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}}) = D$.

Because edges added in the above construction depend only on node labels in D_{outer}, intuitively, AttachGraphs preserves homomorphisms between outer graphs, which is stated formally below.

Fact 4.3. For a graph D, its detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, and a graph H_{outer}, if $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$, then $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}}) \to \mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$.

4.2 Overview

Recall that, for an instance $\tilde{\mathcal{S}}, \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem, with $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$, instead of considering all graphs D that satisfy $\tilde{\mathcal{S}}$, we want to consider their detachments $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, with D_{inner} of bounded size. We want to "distribute" $\mathcal{S}$ and $\mathcal{H}$ over the inner graph and the outer graph; we claim that detachments allow achieving this. We will show that:

- the graph D_{inner} and the shallow modal profile of D_{outer} determine the shallow modal profile of D;
- for a fixed graph D_{inner}, for each graph H we can compute a set $\mathcal{H}_{\mathsf{outer}}$ of graphs such that the following are equivalent:
  - $H \to D$;
  - $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$ for some $H_{\mathsf{outer}} \in \mathcal{H}_{\mathsf{outer}}$.

Thanks to these properties, we are able to reduce the problem to the homomorphism coverage problem by considering all relevant graphs D_{inner} (up to isomorphism) and all potential shallow modal profiles $\mathcal{S}_{\mathsf{outer}}$ of D_{outer}, and solving the homomorphism coverage problem for $\mathcal{S}_{\mathsf{outer}}$ and $\mathcal{H}_{\mathsf{outer}}$.

We begin by proving the first property. Then, we describe the reduction more formally, and prove its correctness, including the second property.

4.3 Distributing the modal profile

Consider graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ such that all node labels in D_{outer} are outer node labels for D_{inner}, and let $\mathcal{S}_{\mathsf{outer}}$ be the shallow modal profile of D_{outer}. We define $\mathsf{AttachProfile}(D_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$ as the shallow modal profile of $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$; by the claim below, this function is correctly defined.

Claim 4.4. For graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ such that all node labels in D_{outer} are outer node labels for D_{inner}, for $D = \mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, the shallow modal profile of D is determined by the graph D_{inner} and the shallow modal profile of D_{outer}.

Proof. We begin by describing edges outgoing from a node in D. By the definition of AttachGraphs, for each node w in D:

- if w is a node in D_{inner}, then there is an outgoing edge:
  - (w, a, v) for each edge (w, a, v) outgoing from w in D_{inner};
  - (w, a, u) for each node u in D_{outer} with label $(A, \mathsf{EIn}, \mathsf{EOut})$ such that $(w, a) \in \mathsf{EIn}$;
- if w is a node in D_{outer} and its label is $(A, \mathsf{EIn}, \mathsf{EOut})$, then there is an outgoing edge:
  - (w, a, v) for each $(a, v) \in \mathsf{EOut}$;
  - (w, a, u) for each edge (w, a, u) outgoing from w in D_{outer}.

Directly from the definition of the 1-bisimilarity type of a node in D, it is straightforward to verify that:

- for each node v in D_{inner}, the 1-bisimilarity type of v in D is determined by its 1-bisimilarity type in D_{inner} and by the set of node labels in D_{outer};
- for each node u in D_{outer}, the 1-bisimilarity type of u in D is determined by its 1-bisimilarity type in D_{outer} and by the graph D_{inner}.

Hence, the shallow modal profile of D is determined by the graph D_{inner} and the shallow modal profile of D_{outer}. □

We proved that the function AttachProfile is correctly defined. Moreover, it is computable: we assumed at the beginning of this chapter that the constructive satisfiability problem for the class $\mathbb{G}$ of all graphs is decidable, so we can consider an arbitrary graph D_{outer} satisfying $\mathcal{S}_{\mathsf{outer}}$, construct $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ and calculate its shallow modal profile.

4.4 The reduction

Consider a negative instance $\tilde{\mathcal{S}}, \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for $\mathbb{G}$, and its counterexample: a graph D that satisfies $\tilde{\mathcal{S}}$, admits a homomorphism h from some $G \in \mathcal{G}$, and admits no homomorphism from any graph in $\mathcal{H}$.

Let $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$. Let V be the image of h extended with all nominal nodes in D. Let $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ be the V-detachment of D. The idea is to iterate over all relevant graphs D_{inner} (up to isomorphism) and over all shallow modal profiles $\mathcal{S}_{\mathsf{outer}}$ such that $\mathsf{AttachProfile}(D_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}}) = \mathcal{S}$, and to define a set of graphs $\mathcal{H}_{\mathsf{outer}}$ such that, if there exists a counterexample D_{outer} to $\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}}(\mathcal{S}_{\mathsf{outer}})$, then the attachment of $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ is a counterexample to $\mathcal{G} \models^{\mathbb{G}, \tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$.

We define three computable functions: InnerGraphs, OuterProfiles and OuterQueries, and prove the following proposition, which shows that the homomorphism entailment problem can be reduced to multiple instances of the homomorphism coverage problem for $\mathbb{G}$.

Proposition 4.5. For an instance $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}}), \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for $\mathbb{G}$, $\mathcal{G} \models^{\mathbb{G}, \tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$ iff, for each $G \in \mathcal{G}$,

for each $D_{\mathsf{inner}} \in \mathsf{InnerGraphs}(G, \Gamma_{\mathrm{N}}, \Sigma_{\mathcal{S}})$,
for each $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$,
$\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}}(\mathcal{S}_{\mathsf{outer}})$, where $\mathcal{H}_{\mathsf{outer}} = \mathsf{OuterQueries}(\mathcal{H}, D_{\mathsf{inner}}, \Sigma_{\mathcal{S}})$.

Computability of sets of graphs

For computability, we introduce the following notion. A graph G is *natural* if $V_G = \{1, \ldots, |V_G|\}$.

Fact 4.6. For finite sets $\Gamma, \Sigma \subseteq \mathbb{U}$ and a non-negative integer n, there are finitely many natural graphs with at most n nodes, node labels in Γ, and edge labels in Σ, and their set is computable.

4.4.1 Inner graphs

For a graph G and finite sets $\Gamma_N, \Sigma \subseteq \mathbb{U}$, let $\mathsf{InnerGraphs}(G, \Gamma_N, \Sigma)$ denote the set of all natural graphs D_{inner} in which:

- there are at most $|V_G| + |\Gamma_N|$ nodes;
- all node labels are in $\Gamma_G \cup \Gamma_N$;
- all edge labels are in Σ;

such that D_{inner} is Γ_N-unique and $G \to D_{\mathsf{inner}}$. Note that the function $\mathsf{InnerGraphs}$ is computable.

Fact 4.7. For a graph G, finite sets $\Gamma_N, \Sigma \subseteq \mathbb{U}$, a Γ_N-unique graph D such that $\Sigma_D \subseteq \Sigma$, and a homomorphism h from G to D, some graph in $\mathsf{InnerGraphs}(G, \Gamma_N, \Sigma)$ is isomorphic to the subgraph of D induced by the image of h extended with all nominal nodes in D.

4.4.2 Outer modal profiles

For a graph D_{inner} and a shallow modal profile with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_N)$, let $\mathsf{OuterProfiles}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$ denote the set of satisfiable shallow modal profiles $\mathcal{S}_{\mathsf{outer}}$ in which:

- all node labels in $\mathcal{S}_{\mathsf{outer}}$ are in $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma_{\mathcal{S}} \setminus \Gamma_N, \Sigma_{\mathcal{S}})$;
- all edge labels in $\mathcal{S}_{\mathsf{outer}}$ are in $\Sigma_{\mathcal{S}}$;

such that $\mathcal{S} = \mathsf{AttachProfile}(D_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$. Note that the function $\mathsf{OuterProfiles}$ is computable, although once again we rely on the assumption made at the beginning of this chapter, that the (constructive) satisfiability problem for the class $\mathbb{G}$ of all graphs is computable.

Fact 4.8. For every graph D and its detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, for every shallow modal profile with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_N)$ such that D_{inner} is Γ_N-unique, the following are equivalent:

- $D \models \tilde{\mathcal{S}}$;
- $D_{\mathsf{outer}} \models \mathcal{S}_{\mathsf{outer}}$ for some $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$.

4.4.3 Outer queries

For graphs D_{inner}, H and a finite set Σ of edge labels, let $\mathsf{OuterQueries}(H, D_{\mathsf{inner}}, \Sigma)$ denote the set of all natural graphs H_{outer} in which:

- there are at most $|V_H|$ nodes;
- all edge labels are in Σ;
- all node labels are in $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma_H, \Sigma)$;

such that $H \to \mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$. Note that the function $\mathsf{OuterQueries}$ is computable.

Claim 4.9. For a graph D, its detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, a graph H, and a finite set $\Sigma \subseteq \mathbb{U}$ such that $\Sigma_D \subseteq \Sigma$, the following are equivalent:

- $H \to D$;
- $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$ for some $H_{\mathsf{outer}} \in \mathsf{OuterQueries}(H, D_{\mathsf{inner}}, \Sigma)$.

Proof. The bottom-up implication follows from Fact 4.3 on page 45, so it remains to prove the top-down implication. Let h be a homomorphism from H to D. We need to construct a graph H_{outer} with at most $|V_H|$ nodes such that $H \to \mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$. Let V be the set of nodes in D_{inner}. Let D_H be the subgraph of D induced by the union of V and the image of h. Let $(D_{\mathsf{inner}}, H_{\mathsf{outer}})$ be the V-detachment of D_H. It is straightforward to verify that H_{outer} is isomorphic to some graph in $\mathsf{OuterQueries}(H, D_{\mathsf{inner}}, \Sigma)$; clearly, $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$, because H_{outer} is a subgraph of D_{outer}. □

We extend $\mathsf{OuterQueries}$ to finite sets $\mathcal{H}$ of graphs, with $\mathsf{OuterQueries}(\mathcal{H}, D_{\mathsf{inner}}, \Sigma)$ defined as the union of $\mathsf{OuterQueries}(H, D_{\mathsf{inner}}, \Sigma)$ for all $H \in \mathcal{H}$.

Corollary 4.10. For a set $\mathcal{H}$ of graphs, a graph D and its detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, and a finite set $\Sigma \subseteq \mathbb{U}$ such that $\Sigma_D \subseteq \Sigma$, the following are equivalent:

- $H \to D$ for some $H \in \mathcal{H}$;
- $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$ for some $H_{\mathsf{outer}} \in \mathsf{OuterQueries}(\mathcal{H}, D_{\mathsf{inner}}, \Sigma)$.

4.4.4 The reduction

From Fact 4.7, Fact 4.8 and Corollary 4.10, we get the proposition stated earlier.

Proposition 4.5. For an instance $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}}), \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for $\mathbb{G}$, $\mathcal{G} \models^{\mathbb{G},\tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$ iff, for each $G \in \mathcal{G}$,
for each $D_{\mathsf{inner}} \in \mathsf{InnerGraphs}(G, \Gamma_{\mathrm{N}}, \Sigma_{\mathcal{S}})$,
for each $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$,
$\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}}(\mathcal{S}_{\mathsf{outer}})$, where $\mathcal{H}_{\mathsf{outer}} = \mathsf{OuterQueries}(\mathcal{H}, D_{\mathsf{inner}}, \Sigma_{\mathcal{S}})$.

Proof. We prove the right-to-left implication by contraposition. Consider a counterexample D to $\mathcal{G} \models^{\mathbb{G},\tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$, and let h be a homomorphism from some graph $G \in \mathcal{G}$ to D. Let V be the image of h extended with all nominal nodes in D, and let $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ be the V-detachment of D.

- By Fact 4.7, there exists a graph $D'_{\mathsf{inner}} \in \mathsf{InnerGraphs}(G, \Gamma_{\mathrm{N}}, \Sigma_{\mathcal{S}})$ that is isomorphic to D_{inner}; for simplicity, let us assume that $D_{\mathsf{inner}} = D'_{\mathsf{inner}}$ – formally, we should rename appropriate nodes in D.
- By Fact 4.8, the shallow modal profile $\mathcal{S}_{\mathsf{outer}}$ of D_{outer} is in $\mathsf{OuterProfiles}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$.
- By Corollary 4.10, since D admits no homomorphism from any graph in $\mathcal{H}$, D_{outer} admits no homomorphism from any $H_{\mathsf{outer}} \in \mathcal{H}_{\mathsf{outer}}$, so D_{outer} is a counterexample to $\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}}(\mathcal{S}_{\mathsf{outer}})$.

We prove the left-to-right implication by contraposition as well. Let G, D_{inner} and $\mathcal{S}_{\mathsf{outer}}$ be such that there exists a counterexample D_{outer} to $\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}}(\mathcal{S}_{\mathsf{outer}})$. Let $D = \mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$. We claim that D is a counterexample to $\mathcal{G} \models^{\mathbb{G},\tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$.

- By the definition of $\mathsf{InnerGraphs}$, $G \to D_{\mathsf{inner}}$, so $G \to D$; also, D_{inner} is Γ_{N}-unique.
- By Fact 4.8 and because $D_{\mathsf{outer}} \models \mathcal{S}_{\mathsf{outer}}$ and D_{inner} is Γ_{N}-unique, $D \models \tilde{\mathcal{S}}$.
- By Corollary 4.10, since D_{outer} admits no homomorphism from any graph in $\mathcal{H}$, D admits no homomorphism from any graph in $\mathcal{H}$. □

4.5 Summary

We have shown that the homomorphism entailment problem for the class $\mathbb{G}$ of all graphs can be reduced to the homomorphism coverage problem for $\mathbb{G}$ – assuming that the constructive satisfiability problem for $\mathbb{G}$ is computable. The reduction is based on *detachments* of graphs. An analogous reduction for the main result of this thesis will also use detachments, in a very similar way. However, proving that detachments behave well with respect to homomorphisms and modal profiles will be significantly more challenging in the presence of regular expressions.

Chapter 5

The toolbox

In this chapter we define the core computational problem that we work with in Chapters 6 to 9: the *minimal downsets problem*. After the definition, we present a reduction from the homomorphism coverage problem, and introduce several constructions and lemmas that simplify working with the minimal downsets problem.

5.1 The minimal downsets problem

5.1.1 Definitions

Downsets

For a graph G and a non-negative integer n, we define $\mathsf{Downset}_n(G)$ as the set of all graphs with at most n edges that map homomorphically to G. We will refer to such sets as *downsets*.

Remark. We bound the size of graphs by the number of *edges*, because it aligns well with some proofs; one could just as well work with graphs with at most n nodes.

As an example, consider simple cycles C_1, C_2, C_3, presented in Figure 5.1 on the next page. The downset $\mathsf{Downset}_2(C_1)$ is the set of all graphs with at most two edges that map homomorphically to C_1; this is simply the set of all graphs with at most two edges (with a single node label ● and one edge label a). The downset $\mathsf{Downset}_2(C_2)$ is the subset of all such graphs without loops (edges from a node to itself). The downset $\mathsf{Downset}_2(C_3)$ is the subset of all such graphs without cycles. However, for higher numbers of edges – e.g. for $\mathsf{Downset}_3(C_3)$ – it seems difficult to find simple descriptions of downsets, other than their definitions; consider for example the graph presented in Figure 5.2 that does not belong to $\mathsf{Downset}_3(C_3)$.

Fact 5.1. For graphs G, H and a non-negative integer n, $\mathsf{Downset}_n(G) \subseteq \mathsf{Downset}_n(H)$ iff every subgraph of G with at most n edges maps homomorphically to H (equivalently: to a subgraph of H with at most n edges).

Fact 5.2. For graphs G, H, if $G \to H$ then $\mathsf{Downset}_n(G) \subseteq \mathsf{Downset}_n(H)$ for all non-negative integers n.

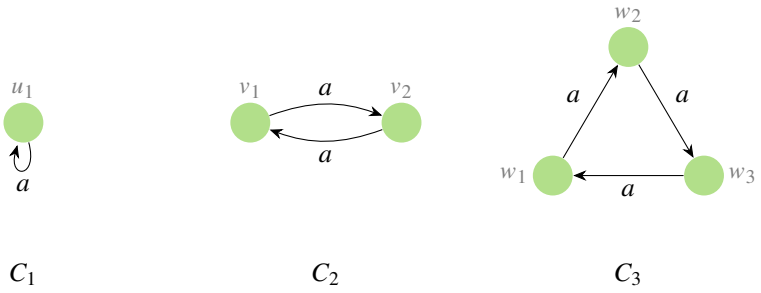


Figure 5.1: Simple cycles C_1, C_2, C_3, with a single node label ● and one edge label a.

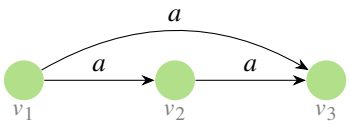


Figure 5.2: An example of a graph that does not belong to $\mathsf{Downset}_3(C_3)$.

Here are two alternative ways of thinking about the set $\mathsf{Downset}_n(G)$:

- the set of all Boolean conjunctive queries that have at most n binary atoms and are satisfied in G;
- the downward closure of the set of homomorphism types of subgraphs of G with at most n edges[1].

Importantly, the latter set can be naturally represented as the set of *maximal* homomorphism types of subgraphs of G with at most n edges, which has size bounded in terms of n, $|\Gamma_G|$ and $|\Sigma_G|$. This leads to the following fact.

Fact 5.3. For a non-negative integer n and finite sets $\Gamma, \Sigma \subseteq \mathbb{U}$, there are finitely many different downsets $\mathsf{Downset}_n(G)$ for graphs G using only node labels in Γ and edge labels in Σ.

Downsets of all models of a modal profile

For a class C of graphs, a modal profile $\mathcal{S}$, and a non-negative integer n, we define $\mathsf{AllDownsets}^n_C(\mathcal{S})$ as the set $\{\mathsf{Downset}_n(G) : G \in \mathsf{Models}_C(\mathcal{S})\}$. Consider the problem of computing $\mathsf{AllDownsets}^n_C(\mathcal{S})$ for given n and $\mathcal{S}$, for a fixed class C of graphs. We argue that the output can be represented in a finite way, and that the homomorphism coverage problem for C can be easily reduced to this problem.

By Fact 5.3, $\mathsf{AllDownsets}^n_C(\mathcal{S})$ is a finite set of downsets. Each downset might be an infinite set, but it can be represented in a finite way – in particular, a downset $\mathcal{D}$ can be represented by a graph G and a non-negative integer n such that $\mathcal{D} = \mathsf{Downset}_n(G)$.

In the homomorphism coverage problem for a class C of graphs, we are given a modal profile $\mathcal{S}$ and a set $\mathcal{H}$ of graphs, and we ask whether every graph in $\mathsf{Models}_C(\mathcal{S})$ admits a homomorphism from some graph in $\mathcal{H}$. This is equivalent to checking, for $n = \max\{|E_H| : H \in \mathcal{H}\}$, if each downset in

[1]Formally, such a downward closure contains also homomorphism types of graphs with more than n edges; however, the notion of downsets is quite robust: for the operations on downsets that we consider (the most important being the containment of one downset in another), we could equivalently use such a definition, or restrict the downward closure to homomorphism types of graphs with at most n edges.

$\mathsf{AllDownsets}^n_C(\mathcal{S})$ contains some graph from $\mathcal{H}$. Since we represent each downset $\mathcal{D}$ by a pair G, n such that $\mathcal{D} = \mathsf{Downset}_n(G)$, the membership of a graph H in $\mathcal{D}$ is equivalent to the existence of a homomorphism from H to G (if H has at most n edges).

However, computing $\mathsf{AllDownsets}^n_C(\mathcal{S})$ seems to be difficult, so we consider a simpler problem, which is also sufficient to solve the homomorphism coverage problem.

Minimal downsets

Let $\mathsf{MinDownsets}^n_C(\mathcal{S})$ denote the set of minimal (inclusion-wise) downsets in $\mathsf{AllDownsets}^n_C(\mathcal{S})$. Computing this set is enough to solve the homomorphism coverage problem, as all downsets in $\mathsf{AllDownsets}^n_C(\mathcal{S})$ contain some graph from $\mathcal{H}$ iff all downsets in $\mathsf{MinDownsets}^n_C(\mathcal{S})$ contain some graph from $\mathcal{H}$. We note the importance of Fact 5.3, which implies that the set $\mathsf{AllDownsets}^n_C(\mathcal{S})$ is finite; in general, the homomorphism order on graphs is not well-founded, which means that working with minimal elements might be unintuitive. We work with minimal elements taken from a finite set, so this does not concern us.

The reason why computing $\mathsf{MinDownsets}^n_C(\mathcal{S})$ is easier than computing $\mathsf{AllDownsets}^n_C(\mathcal{S})$ can be illustrated on the example of C being the class $\mathbb{G}$ of all graphs, considered in the next chapter: as we will see, in this case each minimal downset consists only of trees, and graphs that map homomorphically to trees; this can be argued as the main reason why the considered problems are decidable. For other classes of graphs considered in this thesis, a crucial part of the decidability proofs is to show that, intuitively, it is enough to focus on graphs that can be represented by trees – with their precise definition depending on the considered class of graphs.

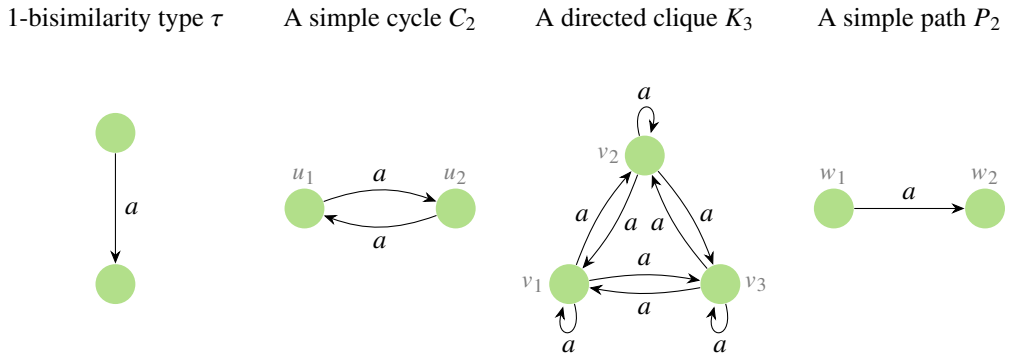


Figure 5.3: An example 1-bisimilarity type τ and graphs C_2, K_3 and P_2.

For an example of $\mathsf{MinDownsets}^n_C(\mathcal{S})$, consider Figure 5.3, representing a 1-bisimilarity type τ and three graphs: a simple cycle C_2, a directed clique K_3, and a simple path P_2. Consider the modal profile $\mathcal{S} = (1, \mathcal{T})$ with $\mathcal{T} = \{\tau\}$; graphs C_2 and K_3 satisfy $\mathcal{S}$, while P_2 does not satisfy $\mathcal{S}$: the 1-bisimilarity type of w_2 is not τ, since w_2 has no outgoing edges. All simple cycles C_n (with a single node label ● and a single edge label a) satisfy $\mathcal{S}$. Note that, for non-negative integers $k < n$, $\mathsf{Downset}_k(C_n)$ consists of a simple path of length k and all graphs (with at most k edges) that map homomorphically to it. It is easy to see that these are, in fact, the smallest downsets of models of $\mathcal{S}$: $\mathsf{MinDownsets}^k_{\mathbb{G}}(\mathcal{S}) = \{\mathsf{Downset}_k(C_{k+1})\}$ for each non-negative integer k.

Equating depths of modal profiles with sizes of graphs in downsets

Throughout the thesis, we work exclusively with sets $\mathsf{MinDownsets}^n_C(\mathcal{S})$ for n equal to the depth of $\mathcal{S}$. For this reason, we omit the parameter n in the notation, and write $\mathsf{MinDownsets}_C(\mathcal{S})$; similarly, we write $\mathsf{AllDownsets}_C(\mathcal{S})$ for $\mathsf{AllDownsets}^n_C(\mathcal{S})$ when n is equal to the depth of $\mathcal{S}$.

This might seem like an arbitrary choice at the moment; the reasons for it are mostly technical.

- The sets $\mathsf{MinDownsets}^n_C(\mathcal{S})$ for classes of graphs considered in this thesis behave well when the depth of $\mathcal{S}$ is greater than or equal to n; for example, for the class of all graphs, $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S})$ contains at most one downset, as we will see in the next chapter.
- The homomorphism coverage problem can be reduced to computing $\mathsf{MinDownsets}_C(\mathcal{S})$; by describing this reduction once, we avoid repeating similar reasoning in the main technical chapters of the thesis.
- We eliminate the parameter n.

Below we formally define the minimal downsets problem. Then, we present a reduction from the homomorphism coverage problem to the minimal downsets problem. At the end of this section, we discuss in slightly more detail the relation of the minimal downsets to *universal* and *k-universal* models that appear in the literature.

The core computational problem

Minimal downsets of a modal profile within a class C of graphs

Input: A modal profile $\mathcal{S}$.
Output: The set $\mathsf{MinDownsets}_C(\mathcal{S})$.

We refer to the above problem as the *minimal downsets problem* for C. This is the problem we work with in the main technical chapters of the thesis: in Chapter 6 we solve the minimal downsets problem for the class $\mathbb{G}$ of all graphs, and in Chapters 7 to 9 we consider other classes of graphs, corresponding to other query classes and more expressive modal constraints, leading up to conjunctive regular path queries and modal constraints expressed in the description logic $\mathcal{ALCO}_{\text{reg}}$.

5.1.2 Reduction from homomorphism coverage

Proposition 5.4. For every class C of graphs, there is a Turing reduction from the homomorphism coverage problem for C to the minimal downsets problem for C.

We argued that an instance $\mathcal{S}, \mathcal{H}$ of the homomorphism coverage problem for a class C of graphs can be solved by computing $\mathsf{MinDownsets}^k_C(\mathcal{S})$ for an appropriately large $k \geq \max\{|E_H| : H \in \mathcal{H}\}$. To reduce the homomorphism coverage problem to the minimal downsets problem, we show that, if the depth of $\mathcal{S}$ is smaller than k, the set $\mathsf{MinDownsets}^k_C(\mathcal{S})$ can be computed based on $\mathsf{MinDownsets}_C(\mathcal{S}')$ for all *k-deepenings* $\mathcal{S}'$ of $\mathcal{S}$.

Recall that the n-bisimilarity type of a node u in a graph G is the bisimilarity type of the n-step unravelling of G from u. The n-bisimilarity profile of a graph G is the set of all n-bisimilarity types of nodes in G. A graph G satisfies a modal profile $\mathcal{S} = (n, \mathcal{T})$ if the n-bisimilarity profile of G is $\mathcal{T}$.

A consequence of Fact 2.5 on page 28 is that, if two graphs have equal k-bisimilarity profiles, then they also have equal n-bisimilarity profiles for each $n < k$. Thus, for a modal profile $\mathcal{S} = (k, \mathcal{T})$ and a positive integer $n < k$, we can define $\mathcal{S}|_n$ as the modal profile of depth n such that, for all graphs G, if $G \models \mathcal{S}$ then $G \models \mathcal{S}|_n$. Formally, $\mathcal{S}|_n$ is the modal profile $(n, \mathcal{T}')$ for $\mathcal{T}' = \{\tau|_n : \tau \in \mathcal{T}\}$.

For a modal profile $\mathcal{S} = (n, \mathcal{T})$ and an integer $k > n$, a *k-deepening* of $\mathcal{S}$ is any modal profile $\mathcal{S}'$ of depth k such that $\mathcal{S}'|_n = \mathcal{S}$, and $\mathcal{S}'$ uses the same labels as $\mathcal{S}$: $\Gamma_{\mathcal{S}} = \Gamma_{\mathcal{S}'}$ and $\Sigma_{\mathcal{S}} = \Sigma_{\mathcal{S}'}$. Recall that, if a graph G satisfies a modal profile $\mathcal{S}$, then G uses the same labels as $\mathcal{S}$: $\Gamma_G = \Gamma_{\mathcal{S}}$ and $\Sigma_G = \Sigma_{\mathcal{S}}$ (Fact 2.7 on page 29).

Fact 5.5. For every graph G, $G \models \mathcal{S}$ iff $G \models \mathcal{S}'$ for some k-deepening $\mathcal{S}'$ of $\mathcal{S}$.

The set of all k-deepenings of $\mathcal{S}$ is computable, because the set of all modal profiles of a given depth that use only labels from fixed finite sets is computable (Fact 2.8 on page 30).

The reduction

Consider a modal profile $\mathcal{S} = (n, \mathcal{T})$ and a positive integer $k > n$. By Fact 5.5, $\mathsf{Models}_C(\mathcal{S})$ is the union of $\mathsf{Models}_C(\mathcal{S}')$ for all k-deepenings $\mathcal{S}'$ of $\mathcal{S}$. Consequently, $\mathsf{AllDownsets}^k_C(\mathcal{S})$ is the union of $\mathsf{AllDownsets}_C(\mathcal{S}')$ for all k-deepenings $\mathcal{S}'$ of $\mathcal{S}$. Thus, $\mathsf{MinDownsets}^k_C(\mathcal{S})$ is the set of minimal downsets in the union of $\mathsf{MinDownsets}_C(\mathcal{S}')$ for all k-deepenings $\mathcal{S}'$ of $\mathcal{S}$.

The reduction can be described as follows. Consider an instance $\mathcal{S}, \mathcal{H}$ of the homomorphism coverage problem for a class C of graphs. Let n be the depth of $\mathcal{S}$, and let $k = \max\{|E_H| : H \in \mathcal{H}\}$. If $n \geq k$, we solve the minimal downsets problem for C and $\mathcal{S}$; the answer is positive iff each downset in $\mathsf{MinDownsets}_C(\mathcal{S})$ contains some graph from $\mathcal{H}$. If $n < k$, the answer is positive iff, for each k-deepening $\mathcal{S}'$ of $\mathcal{S}$, each downset in $\mathsf{MinDownsets}_C(\mathcal{S}')$ contains some graph from $\mathcal{H}$.

5.1.3 Relation to universal models

In the introduction, when discussing the existing techniques for the problems of query containment under constraints and query entailment, we mentioned working with "minimal" models, where the minimality is defined based on the homomorphism order. Particularly when infinite models are considered, it is common to work with a *universal model*: a model that maps homomorphically to all other models. Such a universal model often has the shape of an infinite tree, or, in more complicated cases, a forest-like structure.

When only finite models are considered, a weaker notion is needed, sometimes called a *k-universal model*: a model in which every substructure of size at most k maps homomorphically to every other model. The notion of minimal downsets is closely related to k-universal models: for a modal profile $\mathcal{S}$ and a positive integer k, there exists a k-universal model among $\mathsf{Models}_C(\mathcal{S})$ iff $\mathsf{MinDownsets}^k_C(\mathcal{S})$ is a singleton set $\{\mathcal{D}\}$. Then, a model G such that $\mathsf{Downset}_k(G) = \mathcal{D}$ is a k-universal model.

We defined the minimal downsets problem using $\mathsf{MinDownsets}_C(\mathcal{S})$, rather than $\mathsf{MinDownsets}^k_C(\mathcal{S})$ for a given k, to simplify the proofs in the main technical chapters of the thesis. The reason why using this definition significantly simplifies the proofs is related to the existence of k-universal models: as mentioned earlier, for the class $\mathbb{G}$ of all graphs, it turns out that the condition that the depth of $\mathcal{S}$ is at

least as large as k guarantees that $\mathsf{MinDownsets}^k_{\mathbb{G}}(\mathcal{S})$ contains at most one downset, which means that there exists a k-universal model (if any model exists).

5.2 One-Step Bisimulation Lemma

Below we state and prove a lemma that will be used multiple times throughout the thesis to show that two graphs have equal n-bisimilarity profiles.

Lemma 5.6 (*One-Step Bisimulation Lemma*)**.** Given a modal profile $\mathcal{S} = (n, \mathcal{T})$, a graph G, and a function $\mathsf{SupposedType} : V_G \to \mathcal{T}$, the following conditions are equivalent.

- For each node u in G, $\mathsf{Type}^n_G(u) = \mathsf{SupposedType}(u)$.
- For each node u in G, with $\tau = \mathsf{SupposedType}(u)$:
  - $\mathsf{Label}_G(u) = \mathsf{RootLabel}(\tau)$, and
  - for each edge label a, $\mathsf{Subtypes}(\tau, a) = \{\mathsf{SupposedType}(v)|_{n-1} : v \in \mathsf{Succ}_G(u, a)\}$.

Proof. The top-down implication follows directly from the definition of the n-bisimilarity type of a node in a graph. We prove the bottom-up implication inductively for $k = 0, 1, \ldots, n$; we prove that, for each node u in G, $\mathsf{Type}^k_G(u) = \mathsf{SupposedType}(u)|_k$. Let u be a node in G, and let $\tau = \mathsf{SupposedType}(u)$.

We have $\mathsf{RootLabel}(\mathsf{Type}^k_G(u)) = \mathsf{Label}_G(u) = \mathsf{RootLabel}(\tau|_k)$, so it remains to analyze the function $\mathsf{Subtypes}$.

- For $k = 0$, $\mathsf{Subtypes}(\mathsf{Type}^k_G(u), a)$ is empty for every edge label a, the same as $\mathsf{Subtypes}(\tau|_k, a)$.
- For $k \geq 1$, for each edge label a:

  $$\mathsf{Subtypes}(\mathsf{Type}^k_G(u), a) = \{\mathsf{Type}^{k-1}_G(v) : v \in \mathsf{Succ}_G(u, a)\}.$$

  By the inductive assumption, $\mathsf{Type}^{k-1}_G(v) = \mathsf{SupposedType}(v)|_{k-1}$:

  $$\mathsf{Subtypes}(\mathsf{Type}^k_G(u), a) = \{\mathsf{SupposedType}(v)|_{k-1} : v \in \mathsf{Succ}_G(u, a)\}.$$

  We need to show that $\mathsf{Subtypes}(\mathsf{Type}^k_G(u), a) = \mathsf{Subtypes}(\tau|_k, a)$. For $k = n$, this is exactly the second condition in the statement of the lemma. For $k < n$ it follows, as $\mathsf{Subtypes}(\tau|_k, a)$ can be expressed in terms of $\mathsf{Subtypes}(\tau, a)$ as follows:

  $$\mathsf{Subtypes}(\tau|_k, a) = \{\theta|_{k-1} : \theta \in \mathsf{Subtypes}(\tau, a)\}.$$

□

As an example, consider graphs G, H presented in Figure 5.4, and 1-bisimilarity types τ_1, τ_2, τ_3 of nodes in G. Let $\mathcal{S}$ be the modal profile $(1, \mathcal{T})$ of G of depth 1: $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$. We would like to use the One-Step Bisimulation Lemma to prove that H has the same 1-bisimilarity profile as G. Note that H can be seen as the graph obtained from G by identifying nodes v_1 and v_4, and renaming the nodes; this is the typical situation we will encounter in this thesis: we present a construction of a graph H from a graph G, and we want to prove that H has the same n-bisimilarity profile as G, for some n.

We define $\mathsf{SupposedType} : V_H \to \mathcal{T}$ as follows: $\mathsf{SupposedType}(w_1) = \tau_1$, $\mathsf{SupposedType}(w_2) = \tau_2$, and $\mathsf{SupposedType}(w_3) = \tau_3$. By the One-Step Bisimulation Lemma, to prove that $H \models \mathcal{S}$, it is enough to check that, for each node w in H and for $\tau = \mathsf{SupposedType}(w)$:

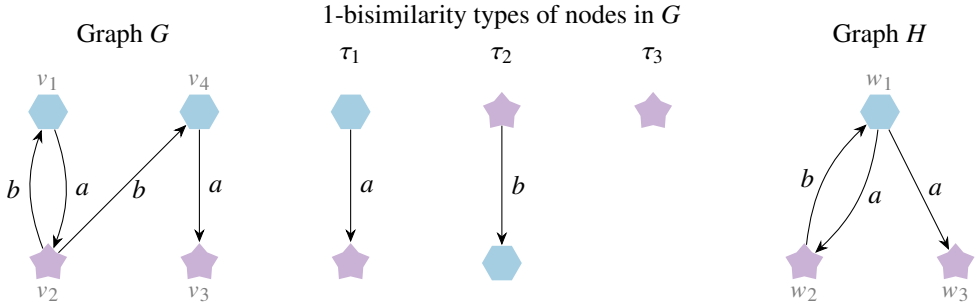


Figure 5.4: A graph G, 1-bisimilarity types of its nodes, and a graph H with the same 1-bisimilarity profile as G.

- $\mathsf{Label}_H(w) = \mathsf{RootLabel}(\tau)$;
- for each edge label a, $\mathsf{Subtypes}(\tau, a) = \{\mathsf{SupposedType}(v)|_0 : v \in \mathsf{Succ}_H(w, a)\}$.

For example, for the node w_1, the second condition takes the following form:

$$\mathsf{Subtypes}(\tau_1, a) = \{\mathsf{SupposedType}(w_2)|_0, \mathsf{SupposedType}(w_3)|_0\} = \{\tau_2|_0, \tau_3|_0\}.$$

This equality can be verified visually, since $\mathsf{RootLabel}(\tau_2) = \mathsf{RootLabel}(\tau_3) =$ ★, which means that $\tau_2|_0$ and $\tau_3|_0$ are both equal to the unique subtype of τ_1. To prove it formally, we use the graph G, and the 1-bisimilarity types of its nodes: $\mathsf{Type}^1_G(v_1) = \mathsf{Type}^1_G(v_4) = \tau_1$, $\mathsf{Type}^1_G(v_2) = \tau_2$, $\mathsf{Type}^1_G(v_3) = \tau_3$.

From the recursive characterization of n-bisimilarity types of nodes in a graph, or again from the One-Step Bisimulation Lemma, we have:

- from the node v_1, $\mathsf{Subtypes}(\tau_1, a) = \{\mathsf{Type}^0_G(v_2)\} = \{\tau_2|_0\}$;
- from the node v_4, $\mathsf{Subtypes}(\tau_1, a) = \{\mathsf{Type}^0_G(v_3)\} = \{\tau_3|_0\}$.

From the above equalities, $\tau_2|_0 = \tau_3|_0$; thus, $\mathsf{Subtypes}(\tau_1, a) = \{\tau_2|_0, \tau_3|_0\}$, which is exactly what we needed to prove. Alternatively, we can take the union of the (equal) sets above, also getting $\mathsf{Subtypes}(\tau_1, a) = \{\tau_2|_0, \tau_3|_0\}$.

In most constructions used in this thesis, the reasoning will be split into two parts; on this example, to prove that $\mathsf{Type}^1_H(w_1) = \tau_1$, we would show two set inclusions, for each edge label a:

- $\mathsf{Subtypes}(\tau_1, a) \supseteq \{\mathsf{SupposedType}(v)|_0 : v \in \mathsf{Succ}_H(w_1, a)\}$;
- $\mathsf{Subtypes}(\tau_1, a) \subseteq \{\mathsf{SupposedType}(v)|_0 : v \in \mathsf{Succ}_H(w_1, a)\}$.

These set inclusions can be intuitively interpreted as follows:

- for each outgoing edge from w_1 in H, there is a corresponding edge outgoing from some node of type τ_1 in G: an edge with the same label, to a node with the same (supposed) 0-bisimilarity type;
- there is some node v in G of type τ_1 such that, for each edge outgoing from v in G, there is a corresponding edge outgoing from w_1 in H.

The above interpretation is quite similar to a common game characterization of bisimilarity, based on Ehrenfeucht-Fraïssé games. We do not use the game characterization for two reasons: while talking about games is often more intuitive, the arguments are typically informal; and, more importantly, proving that two graphs have equal n-bisimilarity profiles using the game characterization would likely involve an inductive reasoning over n, while using the One-Step Bisimulation Lemma avoids it. This is especially important in Chapters 7 and 8, where this would result in a nested inductive reasoning.

5.3 Ravelling constructions

In this section we define a family of constructions on graphs that preserve modal profiles (of all depths) and guarantee that the resulting graph maps homomorphically to the original graph. These constructions are the backbone of a majority of combinatorial results in this thesis; we call them *ravelling constructions*. For example, they will be used to construct a graph that is, intuitively, locally tree-like: its connected subgraphs of bounded size map homomorphically to trees.

Consider a graph G. A *G-ravelling function* is a function f from walks in G to an arbitrary finite set, such that f is *last-node sensitive*: if $\mathsf{LastNode}(\pi) \neq \mathsf{LastNode}(\pi')$, then $f(\pi) \neq f(\pi')$. For a G-ravelling function f, we define $\mathsf{Ravel}_f(G)$ as the graph with nodes $\{f(\pi) : \pi \text{ is a walk in } G\}$ and edges $\{(f(\pi), a, f(\pi')) : \pi' \in \mathsf{Extensions}_G(\pi, a)\}$. The label of a node $f(\pi)$ is the label of $\mathsf{LastNode}(\pi)$ in G; since f is last-node sensitive, this label is unambiguous.

Let $G' = \mathsf{Ravel}_f(G)$. There is a natural homomorphism h from G' to G, mapping $f(\pi)$ to $\mathsf{LastNode}(\pi)$; we call it the *ravelling homomorphism*. Note that h is surjective, because for every node u in G there exists a walk in G that ends in u: the trivial walk $[u]$. For each node u' in G', its *ravelling witness* is any walk π in G such that $f(\pi) = u'$. For each edge e' in G', its *ravelling witness* is any triple (π_1, e, π_2) such that π_1, π_2 are walks in G and e is an edge in G, $\pi_2 = \pi_1 \cdot e$, and $e' = (f(\pi_1), a, f(\pi_2))$, where a is the label of e.

For convenience, we extend ravelling constructions to functions f that are not last-node sensitive. For a graph G and a function f from walks in G to an arbitrary finite set, let $\mathsf{SafeRavel}_f(G)$ be defined as $\mathsf{Ravel}_g(G)$ for g mapping each walk π in G to $(f(\pi), \mathsf{LastNode}(\pi))$.

An example of a ravelling construction is presented in Figure 5.5.

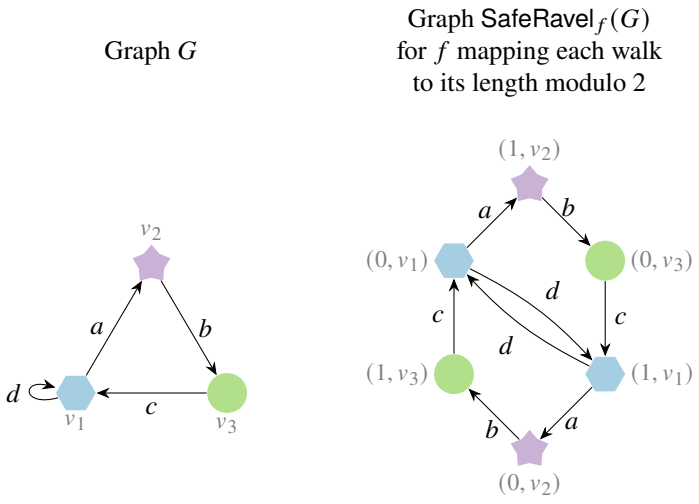


Figure 5.5: A graph G and a graph obtained from G using a ravelling construction. For example, the walk $\pi = [v_1, (v_1, a, v_2), v_2, (v_2, b, v_3), v_3]$ in G has length 2, so $f(\pi) = 0$, and the corresponding node in $\mathsf{SafeRavel}_f(G)$ is $(0, v_3)$, with the label of v_3 in G.

Lemma 5.7 (*Ravelling Profile Preservation Lemma*)**.** For every graph G and every G-ravelling function f, $\mathsf{Ravel}_f(G)$ has the same n-bisimilarity profile as G for every positive integer n.

Proof. Let $H = \mathsf{Ravel}_f(G)$ and let h be the ravelling homomorphism from H to G. Let n be a positive integer. We claim that, for each node u in H, $\mathsf{Type}^n_H(u) = \mathsf{Type}^n_G(h(u))$; since h is surjective, the lemma follows. We use the One-Step Bisimulation Lemma, for $\mathsf{SupposedType}$ mapping a node u to $\mathsf{Type}^n_G(h(u))$. Consider a node u in H. The label of u in H is the label of $h(u)$ in G. We need to prove that, for each edge label a,

$$\mathsf{Subtypes}(\mathsf{Type}^n_G(h(u)), a) = \left\{\mathsf{Type}^{n-1}_G(h(v)) : v \in \mathsf{Succ}_H(u, a)\right\}.$$

We first prove that $\mathsf{Subtypes}(\mathsf{Type}^n_G(h(u)), a) \subseteq \left\{\mathsf{Type}^{n-1}_G(h(v)) : v \in \mathsf{Succ}_H(u, a)\right\}$. Consider some ravelling witness π_1 of u; recall that $f(\pi_1) = u$, and $h(u) = \mathsf{LastNode}(\pi_1)$. Let $u' = h(u)$; we need to prove that for each node $v' \in \mathsf{Succ}_G(u', a)$ there is some node $v \in \mathsf{Succ}_H(u, a)$ such that $\mathsf{Type}^{n-1}_G(h(v)) = \mathsf{Type}^{n-1}_G(v')$. Let $e' = (u', a, v')$, $\pi_2 = \pi_1 \cdot e'$, and $v = f(\pi_2)$. By the construction of H, there is an edge $(u, a, v) = (f(\pi_1), a, f(\pi_2))$ in H. Since $h(v) = v'$, we have $\mathsf{Type}^{n-1}_G(h(v)) = \mathsf{Type}^{n-1}_G(v')$.

Now we prove that $\left\{\mathsf{Type}^{n-1}_G(h(v)) : v \in \mathsf{Succ}_H(u, a)\right\} \subseteq \mathsf{Subtypes}(\mathsf{Type}^n_G(h(u)), a)$. For each edge (u, a, v) in H, consider its ravelling witness (π_1, e', π_2); recall that, for $e' = (u', a, v')$, we have $h(u) = u'$ and $h(v) = v'$. Since e' is an edge in G, we have $\mathsf{Type}^{n-1}_G(v') \in \mathsf{Subtypes}(\mathsf{Type}^n_G(u'), a)$, which proves the inclusion. □

5.4 Trace-based graph transformations

As the last construction in this chapter, we introduce *trace-based graph transformations*. Such a transformation t maps a graph G to a graph $t(G)$ with the same set of nodes, but potentially different edges and node labels. We will use such transformations several times in this thesis as a way of annotating a graph with some information – for example, the reachability between nodes – in the form of additional edges; in this section we prove some general lemmas regarding homomorphisms, downsets, and modal profiles of the resulting graphs. However, trace-based graph transformations will also be used in a different manner in a later reduction between two variants of the minimal downsets problem, in Chapter 9; for this reason, in this section we prove another technical lemma showing a Turing reduction between the minimal downsets problem for two classes of graphs, if, intuitively, one class can be obtained from the other by applying a trace-based graph transformation.

5.4.1 Definitions

The *trace* of a walk π is the sequence of labels of nodes and edges in π: if $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, a_i, v_i)$ and each A_i being the label of v_i, the trace of π is the sequence $L = (A_0, a_1, A_1, \ldots, a_m, A_m)$. The *walk-length* of L, denoted $|L|$, is the length of π: $|L| = |\pi| = m$. In general, a *trace* is a sequence of elements of $\mathbb{U}$ that has an odd length.

For a graph G and a positive integer k, the graph $\mathsf{Trace}_k(G)$ is the graph with nodes V_G, with the same labels as in G, and an edge (u, L, v) iff there is a walk in G from u to v of length at most k with trace L.

A *trace mapping* g consists of two functions: g_Γ mapping node labels to node labels, and g_Σ mapping traces to finite sets of edge labels. For a graph G in which edge labels are traces, we define $\mathsf{Map}_g(G)$ as the graph obtained from G by replacing each node label A with $g_\Gamma(A)$, and replacing each edge (u, L, v) with edges (u, a, v) for each $a \in g_\Sigma(L)$. We define $\mathsf{TraceMap}_k^g(G)$ as $\mathsf{Map}_g(\mathsf{Trace}_k(G))$. We call the function $\mathsf{TraceMap}_k^g$ a trace-based graph transformation, and we say that it depends on walks of length at most k. See Figure 5.6 for an example.

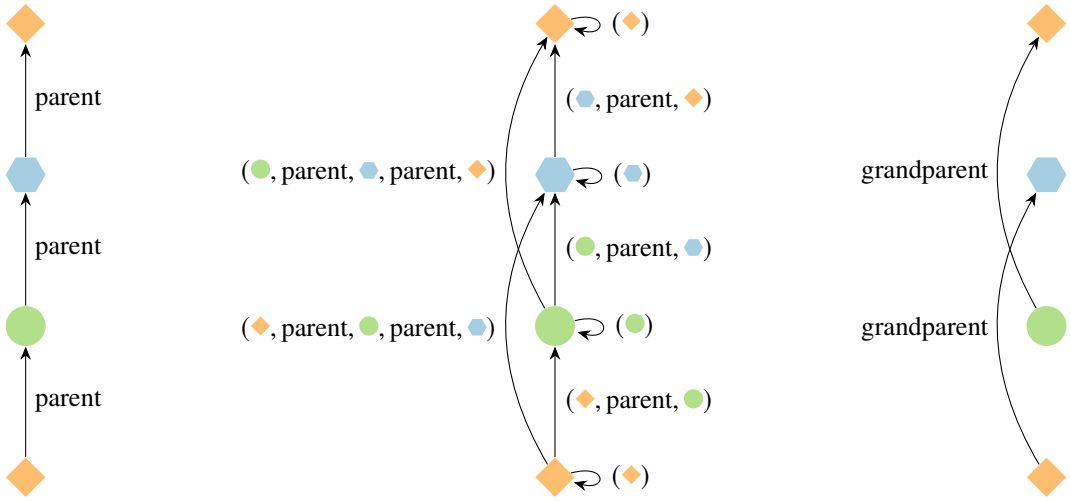


Figure 5.6: Graphs G, $\mathsf{Trace}_2(G)$ and $\mathsf{TraceMap}_2^g(G)$, for g_Γ being the identity, and g_Σ mapping each trace of walk-length 2 to the singleton {grandparent}, and all other traces to the empty set.

5.4.2 Properties

As we will see later, trace-based graph transformations are well-suited to talk about regular path queries. Moreover, they behave well in the context of homomorphisms, downsets, and modal profiles: for every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$ and graphs G, H:

- t preserves homomorphisms: if $G \to H$, then $t(G) \to t(H)$;
- if $\mathsf{Downset}_{nk}(G) \subseteq \mathsf{Downset}_{nk}(H)$, then $\mathsf{Downset}_n(t(G)) \subseteq \mathsf{Downset}_n(t(H))$;
- the modal profile of G of depth nk determines the modal profile of $t(G)$ of depth n.

Consequently, as we will see, for classes C_1, C_2 of graphs such that $C_1 = \{t(G) : G \in C_2\}$, the minimal downsets problem for C_1 can be reduced to the same problem for C_2 (under some minor computability conditions); this also holds for the constructive satisfiability problem.

Below we state the above properties formally; we move technical proofs to the end of this section.

The first property follows easily from the definitions of homomorphisms and traces: each homomorphism between graphs preserves node labels, and it can be lifted to walks, preserving their lengths and traces.

Fact 5.8. For every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$ and graphs G, H, every homomorphism from G to H is also a homomorphism from $t(G)$ to $t(H)$.

The second property is crucial for the reduction between the minimal downsets problem for different classes, as it shows that trace-based graph transformations preserve non-minimality of downsets. We

prove it at the end of this section.

Claim 5.9. For every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$, all graphs G, H, and all non-negative integers n, if $\mathsf{Downset}_{nk}(G) \subseteq \mathsf{Downset}_{nk}(H)$, then $\mathsf{Downset}_n(t(G)) \subseteq \mathsf{Downset}_n(t(H))$.

Corollary 5.10. For all trace-based graph transformations $t = \mathsf{TraceMap}_k^g$, graphs G, H, and non-negative integers n, if $\mathsf{Downset}_{nk}(G) = \mathsf{Downset}_{nk}(H)$, then $\mathsf{Downset}_n(t(G)) = \mathsf{Downset}_n(t(H))$.

The third property is used in the reductions to "translate" modal profiles of graphs in one class to the other class. It is also used in a different context: for a graph G, every graph H obtained from G using ravelling constructions (defined in the previous section) has the same modal profile as G (for each depth). By the lemma below, the graphs $t(G)$ and $t(H)$ also have equal modal profiles for all depths.

We say that a trace-based graph transformation $\mathsf{TraceMap}_k^g$ is *computable* if g_Γ and g_Σ are computable.

Lemma 5.11. For every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$ there is a function F such that, for every graph G and positive integer n, the modal profile of $t(G)$ of depth n is $F(\mathcal{S})$, where $\mathcal{S}$ is the modal profile of G of depth nk. Moreover, if t is computable, then F is computable.

Defining the function F is simple: for $\mathcal{S} = (nk, \mathcal{T})$, $F(\mathcal{S}) = (n, \mathcal{T}')$, where $\mathcal{T}'$ is obtained, intuitively, by applying the transformation t to the representative of every $\tau \in \mathcal{T}$. However, proving that F satisfies the required conditions is technically demanding; we prove it at the end of this section.

Corollary 5.12. For a trace-based graph transformation t, a graph G, a G-ravelling function f, and $H = \mathsf{Ravel}_f(G)$, the n-bisimilarity profiles of $t(G)$ and $t(H)$ are equal for each non-negative integer n.

Reductions

While the above properties are used multiple times throughout the thesis, the reductions described below are used only once, near the end.

Lemma 5.11 states that there is a computable function F mapping a modal profile of G to a modal profile of $t(G)$ (for modal profiles of appropriate depths). For the reductions below, we need to be able to compute the inverse image of a modal profile $\mathcal{S}$ under F; that is, the set of all modal profiles $\mathcal{S}'$ such that $F(\mathcal{S}') = \mathcal{S}$. In our usage, this will be straightforward: for a given modal profile $\mathcal{S}$, we will be able to define finite sets of relevant node labels and edge labels, and simply consider all modal profiles $\mathcal{S}'$ of appropriate depth, checking if $F(\mathcal{S}') = \mathcal{S}$.

A trace-based graph transformation is *strongly computable* if it is computable and the function from a modal profile to its inverse image under F is computable.

Recall that the constructive satisfiability problem for a class C of graphs is a computational problem of computing some graph $G \in \mathsf{Models}_C(\mathcal{S})$ for a given modal profile $\mathcal{S}$, if such a graph exists.

Lemma 5.13. For every strongly computable trace-based graph transformation $t = \mathsf{TraceMap}_k^g$, for classes C_1, C_2 of graphs such that $C_1 = \{t(G) : G \in C_2\}$, there is a Turing reduction from the constructive satisfiability problem for C_1 to the constructive satisfiability problem for C_2.

Proof. Consider a modal profile $\mathcal{S}$ of depth n. Let F be the function from Lemma 5.11, mapping the modal profile of G of depth nk to the modal profile of $t(G)$ of depth n. Then, $\mathsf{Models}_{C_1}(\mathcal{S})$ is the union of $\{t(G) : G \in \mathsf{Models}_{C_2}(\mathcal{S}')\}$ for all modal profiles $\mathcal{S}'$ of depth nk such that $F(\mathcal{S}') = \mathcal{S}$. Thus, to construct some graph in $\mathsf{Models}_{C_1}(\mathcal{S})$, if it exists, it is enough to iterate over all such modal profiles $\mathcal{S}'$ – the set of such modal profiles is computable by the strong computability assumption – solve the constructive satisfiability problem for $\mathcal{S}'$, and if it yields a graph G, output $t(G)$. □

Lemma 5.14. For every strongly computable trace-based graph transformation $t = \mathsf{TraceMap}^g_k$, for classes C_1, C_2 of graphs such that $C_1 = \{t(G) : G \in C_2\}$, there is a Turing reduction from the minimal downsets problem for C_1 to the minimal downsets problem for C_2.

Proof. Consider an instance of the minimal downsets problem for C_1: a modal profile $\mathcal{S}$ of depth n. Let F be the function from Lemma 5.11, mapping the modal profile of G of depth nk to the modal profile of $t(G)$ of depth n.

Recall that we represent a downset $\mathcal{D}$ as a pair (G, m) such that $\mathcal{D} = \mathsf{Downset}_m(G)$. The algorithm is quite simple: for each modal profile $\mathcal{S}'$ of depth nk such that $F(\mathcal{S}') = \mathcal{S}$, compute the set of minimal downsets for C_2 and $\mathcal{S}'$; for each downset, represented by a pair (G, nk), calculate $\mathsf{Downset}_n(t(G))$, and return the minimal downsets among all resulting downsets.

Each downset in the resulting set is of the form $\mathsf{Downset}_n(H)$ for some $H \in C_1$; this is obvious if the graph G considered above is in C_2 – otherwise, it follows from Corollary 5.10. It remains to prove that all minimal downsets are present in the resulting set. For this, we prove that, for each graph $H \in C_1$, there is some downset $\mathcal{D}$ in the resulting set such that $\mathcal{D} \subseteq \mathsf{Downset}_n(H)$. By the definition of C_1, $H = t(G)$ for some $G \in C_2$. For the modal profile $\mathcal{S}'$ of depth nk of G, by Lemma 5.11, $F(\mathcal{S}') = \mathcal{S}$, so we considered the modal profile $\mathcal{S}'$ in the procedure. There is some minimal downset $\mathcal{D}$ for C_2 and $\mathcal{S}'$ such that $\mathcal{D} \subseteq \mathsf{Downset}_{nk}(G)$. If $\mathcal{D}$ is represented by (G', nk), then $\mathsf{Downset}_n(t(G')) \subseteq \mathsf{Downset}_n(H)$, by Claim 5.9 (as $H = t(G)$ and $\mathsf{Downset}_{nk}(G') = \mathcal{D} \subseteq \mathsf{Downset}_{nk}(G)$), which is what we needed. □

Example: reachability

Fix an edge label Reachable. Consider a graph G that does not use the edge label Reachable, and the graph H obtained from G by adding an edge $(u, \mathsf{Reachable}, v)$ iff v is reachable from u. For a fixed graph G, this construction can be defined as a trace-based graph transformation, for a trace mapping $g = (g_\Gamma, g_\Sigma)$ such that g_Γ is the identity, and g_Σ maps each trace L to:

- the singleton $\{\mathsf{Reachable}\}$ if the walk-length of L is not 1;
- the set $\{\mathsf{Reachable}, a_1\}$ for $L = (A_0, a_1, A_1)$.

Then, $\mathsf{TraceMap}^g_k(G) = H$, for every $k \geq |V_G|$. This means that we can use the properties of trace-based graph transformations listed above. However, because k depends on the size of G, the reductions described above are not particularly useful; in this thesis we are interested in classes C_1, C_2 of graphs that are infinite and contain arbitrarily large graphs, which means that we cannot define a single trace-based graph transformation that would correspond to the described construction for all graphs in the class.

5.4.3 Remaining proofs

Proof of Claim 5.9

Claim 5.9. For every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$, all graphs G, H, and all non-negative integers n, if $\mathsf{Downset}_{nk}(G) \subseteq \mathsf{Downset}_{nk}(H)$, then $\mathsf{Downset}_n(t(G)) \subseteq \mathsf{Downset}_n(t(H))$.

Let $G_t = t(G), H_t = t(H)$. Recall that, to show that $\mathsf{Downset}_n(G_t) \subseteq \mathsf{Downset}_n(H_t)$, it suffices to prove that every subgraph G'_t of G_t with at most n edges maps homomorphically to H_t (Fact 5.1 on page 51). We will show that for each such subgraph G'_t we can find a subgraph G' of G with at most nk edges such that G'_t is a subgraph of $t(G')$. The rest of the proof is straightforward: we use the assumption $\mathsf{Downset}_{nk}(G) \subseteq \mathsf{Downset}_{nk}(H)$ (which gives $G' \to H$), and the fact that t preserves homomorphisms (which gives $t(G') \to t(H)$, so $G'_t \to H_t$, which is what we needed to prove).

Consider a subgraph G'_t of G_t with at most n edges. By the definition of trace-based graph transformations, for each edge $e = (u, a, v)$ in G_t there is a walk π_e from u to v in G with trace L such that $a \in g_\Sigma(L)$, and the length of π_e is at most k. Let G' be the subgraph of G consisting of all nodes V_G, and all edges traversed by walks π_e for all edges e in G'_t. By the construction, G'_t is a subgraph of $t(G')$, and G' has at most nk edges.

Proof of Lemma 5.11

Lemma 5.11. For every trace-based graph transformation $t = \mathsf{TraceMap}_k^g$ there is a function F such that, for every graph G and positive integer n, the modal profile of $t(G)$ of depth n is $F(\mathcal{S})$, where $\mathcal{S}$ is the modal profile of G of depth nk. Moreover, if t is computable, then F is computable.

For a tree bisimilarity type τ and $T = \mathsf{Representative}(\tau)$, for a non-negative integer n, we write $t(\tau)|_n$ for $\mathsf{Type}^n_{t(T)}(\mathsf{Root}(T))$: the n-bisimilarity type of the root of T in the graph obtained from T by applying the transformation t. Then, for $\mathcal{S} = (nk, \mathcal{T})$, we define $F(\mathcal{S}) = (n, \mathcal{T}')$, where $\mathcal{T}' = \{t(\tau)|_n : \tau \in \mathcal{T}\}$. The function F is easily computable. It remains to prove that, for a node u in G such that $\mathsf{Type}^{nk}_G(u) = \tau$, we have $\mathsf{Type}^n_{t(G)}(u) = t(\tau)|_n$.

We prove it in two steps. Recall that $\mathsf{Trace}_k(G)$ is the graph with nodes V_G, with the same labels as in G, and an edge (u, L, v) iff there is a walk in G from u to v of length at most k with trace L. The graph $\mathsf{Map}_g(H)$ is obtained from H by replacing each node label A with $g_\Gamma(A)$, and replacing each edge (u, L, v) with edges (u, a, v) for each $a \in g_\Sigma(L)$.

We will show that there are functions $\mathsf{TraceType}_k$ and $\mathsf{MapType}_g$ such that, for all graphs G, $H = \mathsf{Trace}_k(G)$ and $D = \mathsf{Map}_g(H)$, for all nodes u in G and all non-negative integers n:

- $\mathsf{TraceType}_k(\mathsf{Type}^{nk}_G(u), n) = \mathsf{Type}^n_H(u)$;
- $\mathsf{MapType}_g(\mathsf{Type}^n_H(u)) = \mathsf{Type}^n_D(u)$.

Recall that we want to prove that, for a node u in a graph G and $\tau = \mathsf{Type}^{nk}_G(u)$, $\mathsf{Type}^n_{t(G)}(u) = t(\tau)|_n$. This follows from the existence of the above functions: for $T = \mathsf{Representative}(\tau)$, and r being the root of T, $\mathsf{Type}^{nk}_T(r) = \tau = \mathsf{Type}^{nk}_G(u)$, so $\mathsf{Type}^n_{t(T)}(r) = \mathsf{MapType}_g(\mathsf{TraceType}_k(\tau, n)) = \mathsf{Type}^n_{t(G)}(u)$.

Defining $\mathsf{TraceType}_k$

We define a function $\mathsf{TraceType}_k(\tau, n)$ such that, for a node u in a graph G and non-negative integers n, m such that $m \geq nk$, $\mathsf{TraceType}_k(\mathsf{Type}^m_G(u), n) = \mathsf{Type}^n_H(u)$. We will define the function $\mathsf{TraceType}_k(\tau, n)$ inductively over n, using auxiliary functions $\mathsf{TraceSucc}_G$ and $\mathsf{TraceSubtypes}$.

For a node u in G and a trace L, let $\mathsf{TraceSucc}_G(u, L)$ denote the set of nodes v for which there is a walk from u to v in G with trace L. By the definition of $H = \mathsf{Trace}_k(G)$, we get the following.

Fact 5.15. For each trace L of walk-length at most k, $\mathsf{Succ}_H(u, L) = \mathsf{TraceSucc}_G(u, L)$.

Fact 5.16. There is a function $\mathsf{TraceSubtypes}$ such that, for a node u in a graph G, a trace L, and an integer $n \geq |L|$, $\mathsf{TraceSubtypes}(\mathsf{Type}^n_G(u), L) = \{\mathsf{Type}^{n-|L|}_G(v) : v \in \mathsf{TraceSucc}_G(u, L)\}$.

We define $\mathsf{TraceType}_k(\tau, n)$ as θ such that $\mathsf{RootLabel}(\theta) = \mathsf{RootLabel}(\tau)$, and $\mathsf{Subtypes}(\theta, a)$ is defined as follows. If $n = 0$, $\mathsf{Subtypes}(\theta, a) = \emptyset$ for all edge labels a. If $n \geq 1$, for each trace L of walk-length at most k, $\mathsf{Subtypes}(\theta, L) = \{\mathsf{TraceType}_k(\tau', n-1) : \tau' \in \mathsf{TraceSubtypes}(\tau, L)\}$.

Claim 5.17. For a node u in a graph G and non-negative integers n, m such that $m \geq nk$, $\mathsf{TraceType}_k(\mathsf{Type}^m_G(u), n) = \mathsf{Type}^n_H(u)$.

Proof. Recall that $H = \mathsf{Trace}_k(G)$. Let $\tau = \mathsf{Type}^m_G(u)$, and let $\theta = \mathsf{TraceType}_k(\tau, n)$. We want to show that $\mathsf{Type}^n_H(u) = \theta$. We prove the claim by induction over n. The root labels match: $\mathsf{RootLabel}(\theta) = \mathsf{RootLabel}(\tau) = \mathsf{Label}_G(u) = \mathsf{Label}_H(u)$. For $n = 0$, $\mathsf{Subtypes}(\theta, a) = \emptyset = \mathsf{Subtypes}(\mathsf{Type}^0_H(u), a)$ for every edge label a. For $n \geq 1$, for every trace L of walk-length at most k:

$$\mathsf{Subtypes}(\theta, L) = \{\mathsf{TraceType}_k(\tau', n-1) : \tau' \in \mathsf{TraceSubtypes}(\tau, L)\}.$$

By the definition of $\mathsf{TraceSubtypes}$:

$$\mathsf{Subtypes}(\theta, L) = \{\mathsf{TraceType}_k(\mathsf{Type}^{m-|L|}_G(v), n-1) : v \in \mathsf{TraceSucc}_G(u, L)\}.$$

Since $m \geq nk$ and $|L| \leq k$, we have $m - |L| \geq (n-1)k$, so, by the inductive assumption:

$$\mathsf{Subtypes}(\theta, L) = \{\mathsf{Type}^{n-1}_H(v) : v \in \mathsf{TraceSucc}_G(u, L)\}.$$

By Fact 5.15:

$$\mathsf{Subtypes}(\theta, L) = \{\mathsf{Type}^{n-1}_H(v) : v \in \mathsf{Succ}_H(u, L)\} = \mathsf{Subtypes}(\mathsf{Type}^n_H(u), L).$$

□

Defining $\mathsf{MapType}_g$

Recall that $D = \mathsf{Map}_g(H)$. We want to define a function $\mathsf{MapType}_g$ such that, for all nodes u in G and all non-negative integers n, $\mathsf{MapType}_g(\mathsf{Type}^n_H(u)) = \mathsf{Type}^n_D(u)$.

We define $\mathsf{MapType}_g$ recursively; $\mathsf{MapType}_g(\tau) = \theta$, where $\mathsf{RootLabel}(\theta) = g_\Gamma(\mathsf{RootLabel}(\tau))$, and, for each edge label a, $\mathsf{Subtypes}(\theta, a) = \{\mathsf{MapType}_g(\tau') : \tau' \in \mathsf{Subtypes}(\tau, L), a \in g_\Sigma(L)\}$.

Fact 5.18. For all nodes u in G and all non-negative integers n, $\mathsf{MapType}_g(\mathsf{Type}^n_H(u)) = \mathsf{Type}^n_D(u)$.

Chapter 6

The base case: the class of all graphs

In this chapter we solve the minimal downsets problem for the class $\mathbb{G}$ of all graphs, by proving the following two propositions.

Proposition 6.1. There is a Turing reduction from the minimal downsets problem for $\mathbb{G}$ to the satisfiability problem for $\mathbb{G}$.

Proposition 6.2. The satisfiability problem for $\mathbb{G}$ is decidable.

Recall that in the minimal downsets problem for $\mathbb{G}$ we need to compute, for a given modal profile $\mathcal{S}$, the set $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S})$: the minimal downsets in the set $\mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$, which in turn is defined as $\{\mathsf{Downset}_n(G) : G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})\}$, where n is the depth of $\mathcal{S}$.

This chapter is split into three sections:

- a reduction from the minimal downsets problem to the satisfiability problem, along with a high-level description of the proof of its correctness;
- the crux of the proof of the correctness of the reduction;
- a solution of the satisfiability problem.

The next two chapters will follow the same structure. The main purpose of this chapter is to present basic techniques and the structure of the proofs in a simple setting.

6.1 Reduction to satisfiability

Below we reduce the minimal downsets problem for $\mathbb{G}$ to the satisfiability problem for $\mathbb{G}$.

The structure of the reduction

Consider a modal profile $\mathcal{S}$. We present an algorithm computing some set of downsets, denoted $\mathsf{AllComputedDownsets}(\mathcal{S})$, and we define $\mathsf{MinComputedDownsets}(\mathcal{S})$ as the set of minimal downsets in $\mathsf{AllComputedDownsets}(\mathcal{S})$. We claim that $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S}) = \mathsf{MinComputedDownsets}(\mathcal{S})$; to prove it, we will show that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$ there is some $\mathcal{D}' \in \mathsf{AllComputedDownsets}(\mathcal{S})$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}(\mathcal{S})$ there is some $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

The first claim will easily follow from properties proven when defining $\mathsf{AllComputedDownsets}(\mathcal{S})$. The crux of the proof of the correctness of the reduction is the second claim, which is proved in a separate section.

The algorithm

Consider a modal profile $\mathcal{S}$ of depth n. If $\mathcal{S}$ is not satisfiable, then $\mathsf{Models}_{\mathbb{G}}(\mathcal{S})$ is the empty set, so $\mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$ and $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S})$ are empty as well. Thus, let us assume that $\mathcal{S}$ is satisfiable.

The central notion we will introduce is the *global representative* of a modal profile; we define $\mathsf{AllComputedDownsets}(\mathcal{S})$ as the singleton set $\{\mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S}))\}$. Thus, in this case, $\mathsf{MinComputedDownsets}(\mathcal{S}) = \mathsf{AllComputedDownsets}(\mathcal{S})$, but this will not necessarily be the case in the next two chapters.

Importantly, we will define $\mathsf{GlobalRepresentative}(\mathcal{S})$ as a graph that does not necessarily satisfy $\mathcal{S}$: it represents the smallest downset of models of $\mathcal{S}$, but might not be a model of $\mathcal{S}$ itself.

6.1.1 Global unravellings and global representatives

Recall that we defined the *representative* of a tree bisimilarity type τ as a canonical (minimal) tree of this bisimilarity type. Thus, by definition, for a node u in a graph G and a non-negative integer n, the n-step unravelling of G from u is bisimilar to the representative of $\mathsf{Type}^n_G(u)$; note that this implies that they are homomorphically equivalent as well (Fact 2.1 on page 26). Below we define a natural notion of *global unravellings* of a graph, and an analogous notion of *global representatives* of a modal profile.

For a non-negative integer n, the *global n-step unravelling* of a graph G, denoted $\mathsf{GlobalUnravel}(n, G)$, can be seen as the union of all n-step unravellings of G: it is the graph with nodes being all walks of length at most n in G, and an edge (π_1, a, π_2) iff $\pi_2 \in \mathsf{Extensions}_G(\pi_1, a)$. The label of a node π is the label of $\mathsf{LastNode}(\pi)$ in G. An example of a graph and its global unravelling is presented in Figure 6.1.

Recall that, for two graphs G, H, if G maps homomorphically to H, we denote it by $G \to H$, and if G, H are homomorphically equivalent, we denote it by $G \leftrightarrow H$.

Fact 6.3. For every graph G and every non-negative integer n, $\mathsf{GlobalUnravel}(n, G) \to G$.

The *global representative* of a modal profile $\mathcal{S} = (n, \mathcal{T})$, denoted $\mathsf{GlobalRepresentative}(\mathcal{S})$, is the disjoint union[1] of the trees $\mathsf{Representative}(\tau)$ for all $\tau \in \mathcal{T}$. An example of a graph, its modal profile, and the global representative of this modal profile is presented in Figure 6.2 on the next page.

Fact 6.4. For every modal profile $\mathcal{S}$ of depth n and every graph G satisfying $\mathcal{S}$:

$$\mathsf{GlobalRepresentative}(\mathcal{S}) \leftrightarrow \mathsf{GlobalUnravel}(n, G).$$

[1]Formally, we could use *union* instead of *disjoint union* and end up with an isomorphic graph, but then we would need to prove that representatives of different tree bisimilarity types have disjoint sets of nodes.

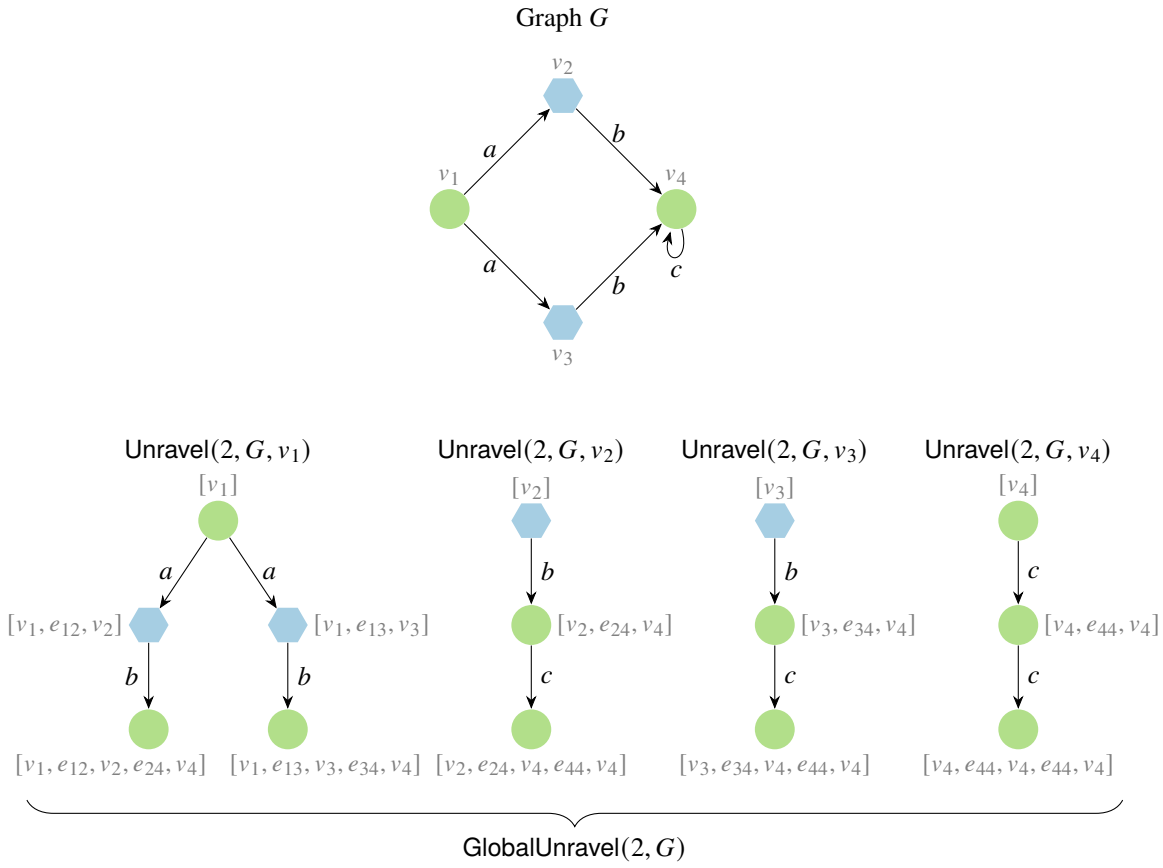


Figure 6.1: A graph and its global 2-step unravelling. We denote by e_{ij} the unique edge from v_i to v_j.

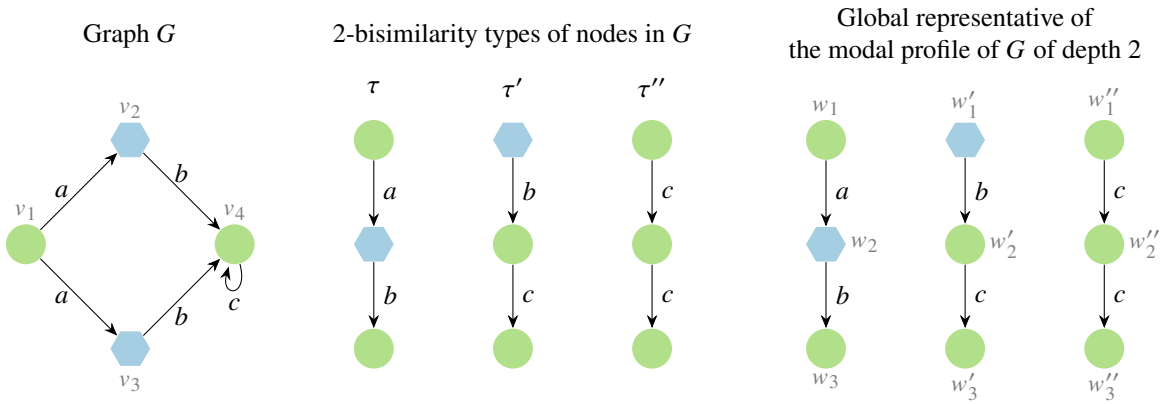


Figure 6.2: A graph, the 2-bisimilarity types of its nodes, and the global representative of its modal profile of depth 2. Visually, the latter two look the same, modulo node identifiers. We do not write out the node identifiers (w_1, w_2, etc.) in terms of τ, τ' and τ'', as they are quite convoluted; intuitively, one can think of w_1 as τ, w_2 as $\tau'|_1$, and w_3 as $\tau''|_0$; however, this is not entirely correct, because this way we would end up with both w'_2 and w''_2 being equal to $\tau''|_1$, but they are separate nodes – to make this separation clear, we used *disjoint* union in the definition of the global representative, but it also follows from the formal definition of the representative of a tree bisimilarity type.

In the next two chapters, when working with other classes of graphs – for example, with some edges corresponding to reachability in the underlying graph – we will define notions of global unravellings and global representatives extended with additional edges, with properties analogous to Fact 6.3 and Fact 6.4.

6.1.2 Correctness of the reduction: the first steps

We claim that $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S}) = \mathsf{MinComputedDownsets}(\mathcal{S})$ for each satisfiable modal profile $\mathcal{S}$. Recall that $\mathsf{AllComputedDownsets}(\mathcal{S}) = \{\mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S}))\}$, where n is the depth of $\mathcal{S}$. To prove the claim, we will show that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$ there is some $\mathcal{D}' \in \mathsf{AllComputedDownsets}(\mathcal{S})$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}(\mathcal{S})$ there is some $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}}(\mathcal{S})$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

The first claim follows from Facts 6.3 and 6.4: for every graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$, by Fact 6.4, $\mathsf{GlobalRepresentative}(\mathcal{S}) \leftrightarrow \mathsf{GlobalUnravel}(n, G)$, and by Fact 6.3, $\mathsf{GlobalUnravel}(n, G) \to G$; thus, $\mathsf{GlobalRepresentative}(\mathcal{S}) \to G$, so $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S})) \subseteq \mathsf{Downset}_n(G)$ (by Fact 5.2 on page 51). The core of the proof of the correctness of the reduction is the second claim, stated below with expanded definitions, and proved in the next section.

Proposition 6.5. For every satisfiable modal profile $\mathcal{S} = (n, \mathcal{T})$ there is a graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$ such that every subgraph of G with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}(\mathcal{S})$.

Assuming that the satisfiability problem for $\mathbb{G}$ is decidable, the computability of the minimal downsets problem for $\mathbb{G}$ follows, as constructing the global representative of a given modal profile $\mathcal{S}$ is easily computable.

6.2 Correctness of the reduction: the crux

In this section we prove Proposition 6.5, to finish the proof of the correctness of the reduction. Let $\mathcal{S} = (n, \mathcal{T})$ be a satisfiable modal profile.

6.2.1 Constructing a witness

Since $\mathcal{S}$ is satisfiable, there exists some graph G_0 satisfying $\mathcal{S}$. We construct the desired graph in two steps, using ravelling constructions.

Let $G_1 = \mathsf{SafeRavel}_f(G_0)$ for f mapping each walk to its length modulo $n + 1$. An example of graphs G and $\mathsf{SafeRavel}_f(G)$ is presented in Figure 6.3 on the next page, for $n = 2$.

Claim 6.6. Each cycle in G_1 has length at least $n + 1$.

Proof. By the definition of $\mathsf{SafeRavel}_f$, G_1 is the graph with nodes $(f(\pi), \mathsf{LastNode}(\pi))$ for each walk π in G_0, and an edge from $(f(\pi), \mathsf{LastNode}(\pi))$ to $(f(\pi'), \mathsf{LastNode}(\pi'))$ with label a for each $\pi' \in \mathsf{Extensions}_{G_0}(\pi, a)$. Thus, the nodes in G_1 are of the form (ℓ, v), where $\ell \in \{0, 1, \ldots, n\}$ and v is a node in G_0, and for each edge $((\ell, v), a, (\ell', v'))$ in G_1 we have $\ell' = (\ell + 1) \bmod (n + 1)$. □

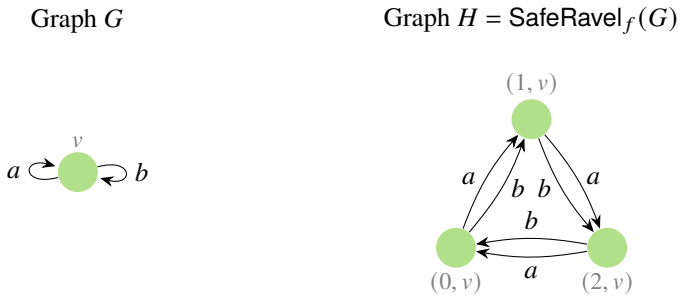


Figure 6.3: Graphs G and $H = \mathsf{SafeRavel}_f(G)$ for f mapping each walk to its length modulo 3. Note that, while H does not contain (directed) cycles of length strictly smaller than 3, there are subgraphs of H with at most 2 edges that do not map homomorphically to any global representative of a modal profile; for example, in the subgraph of H induced by $\{(0, v), (1, v)\}$ there are two edges with different labels incoming to $(1, v)$, which cannot happen in a tree.

Let Suffix_n be the function mapping each walk π in G_1 to its suffix of length n if $|\pi| > n$, or to the whole walk π otherwise. Let $G_2 = \mathsf{Ravel}_f(G_1)$ for $f = \mathsf{Suffix}_n$. Examples of graphs G and $\mathsf{SafeRavel}_f(G)$ for $f = \mathsf{Suffix}_2$ are presented in Figures 6.4 to 6.6.

By the Ravelling Profile Preservation Lemma (Lemma 5.7 on page 58), $G_1 \models \mathcal{S}$ and $G_2 \models \mathcal{S}$. To show this on examples, the lemma states that, in each figure among Figures 6.3 to 6.6, graphs G and H have equal modal profiles of each depth.

We claim that every subgraph of G_2 with at most n edges maps homomorphically to the graph $\mathsf{GlobalRepresentative}(\mathcal{S})$, which is exactly what we need to prove Proposition 6.5. Because nodes in G_2 are walks in G_1, and nodes in $\mathsf{GlobalUnravel}(n, G_1)$ are also walks in G_1, it is convenient to define a homomorphism between them, and use the fact that $\mathsf{GlobalRepresentative}(\mathcal{S}) \leftrightarrow \mathsf{GlobalUnravel}(n, G_1)$ (because $G_1 \models \mathcal{S}$, and by Fact 6.4 on page 66). Thus, it remains to show the following proposition.

Proposition 6.7. For each subgraph D of G_2 with at most n edges, $D \to \mathsf{GlobalUnravel}(n, G_1)$.

We prove this proposition in two steps: we first describe the construction of the required homomorphism, and then prove that it is, indeed, a homomorphism from D to $\mathsf{GlobalUnravel}(n, G_1)$. In this chapter, the arguments are quite short, but in each of the next two chapters, the proof that the constructed function is indeed a homomorphism is one of the most technically involved parts.

Constructing the homomorphism $h : D \to \mathsf{GlobalUnravel}(n, G_1)$

Recall that $\mathsf{GlobalUnravel}(n, G_1)$ is the graph in which the nodes are all walks of length at most n in G_1 and there is an edge (π, a, π') iff $\pi' \in \mathsf{Extensions}_{G_1}(\pi, a)$. In G_2, each node is also a walk in G_1 of length at most n, as G_2 is obtained from G_1 by the ravelling construction for the function Suffix_n. Let $h_{G_2 \to G_1}$ be the ravelling homomorphism from G_2 to G_1, mapping each node π to $\mathsf{LastNode}(\pi)$.

An edge in G_1 is *important* if it is in the graph $h_{G_2 \to G_1}(D)$, using the graph-image notation; that is, if some edge in D is mapped by $h_{G_2 \to G_1}$ to this edge. A walk in G_1 is *important* if it traverses only important edges.

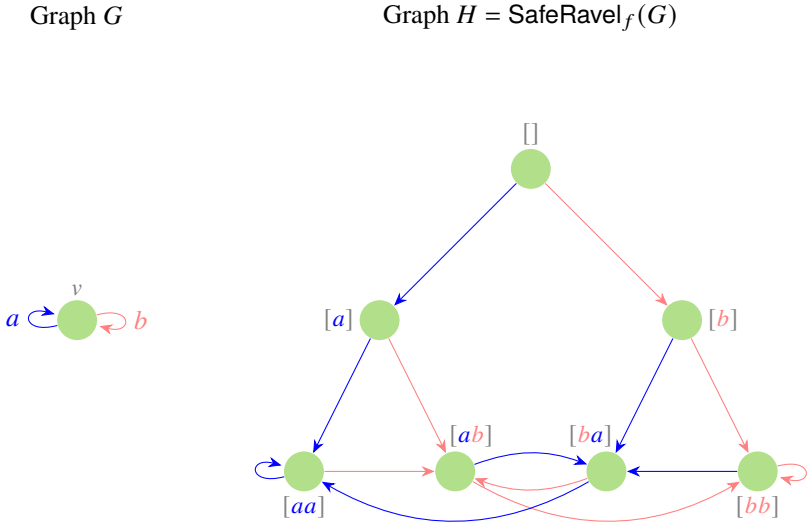


Figure 6.4: Graphs G and H = SafeRavel$_f(G)$ for f mapping each walk to its suffix of length at most 2. In the graph H, we use colors to denote different edge labels, and write node identifiers in a shortened way, to avoid cluttering the picture: for example, the identifier $[ab]$ denotes the sequence $[v, e_a, v, e_b, v]$, where e_a is the unique edge with label a in G, and e_b is the unique edge with label b. Note that the graph H can be obtained from the 2-step unravelling of G from v by adding some edges between its leaves. For each node in H, all incoming edges have the same label; even more, all walks of length 2 ending in a given node have equal traces. However, the graph H contains (directed) cycles of length strictly smaller than 3, including loops (edges from a node to itself).

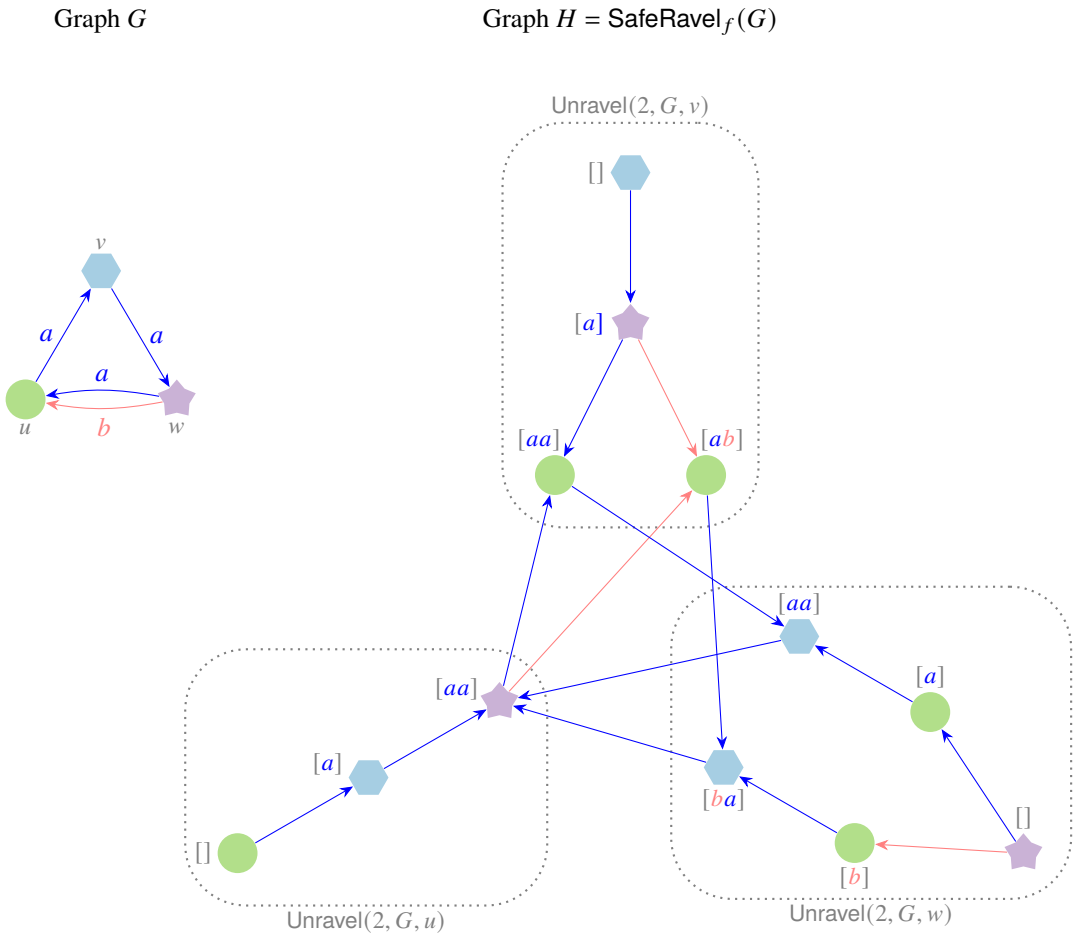


Figure 6.5: Graphs G and H = SafeRavel$_f(G)$ for f mapping each walk to its suffix of length at most 2. We use the same graphical notation for the graph H as in Figure 6.4. Note that H can be obtained from the global 2-step unravelling of G by adding some edges between leaves.

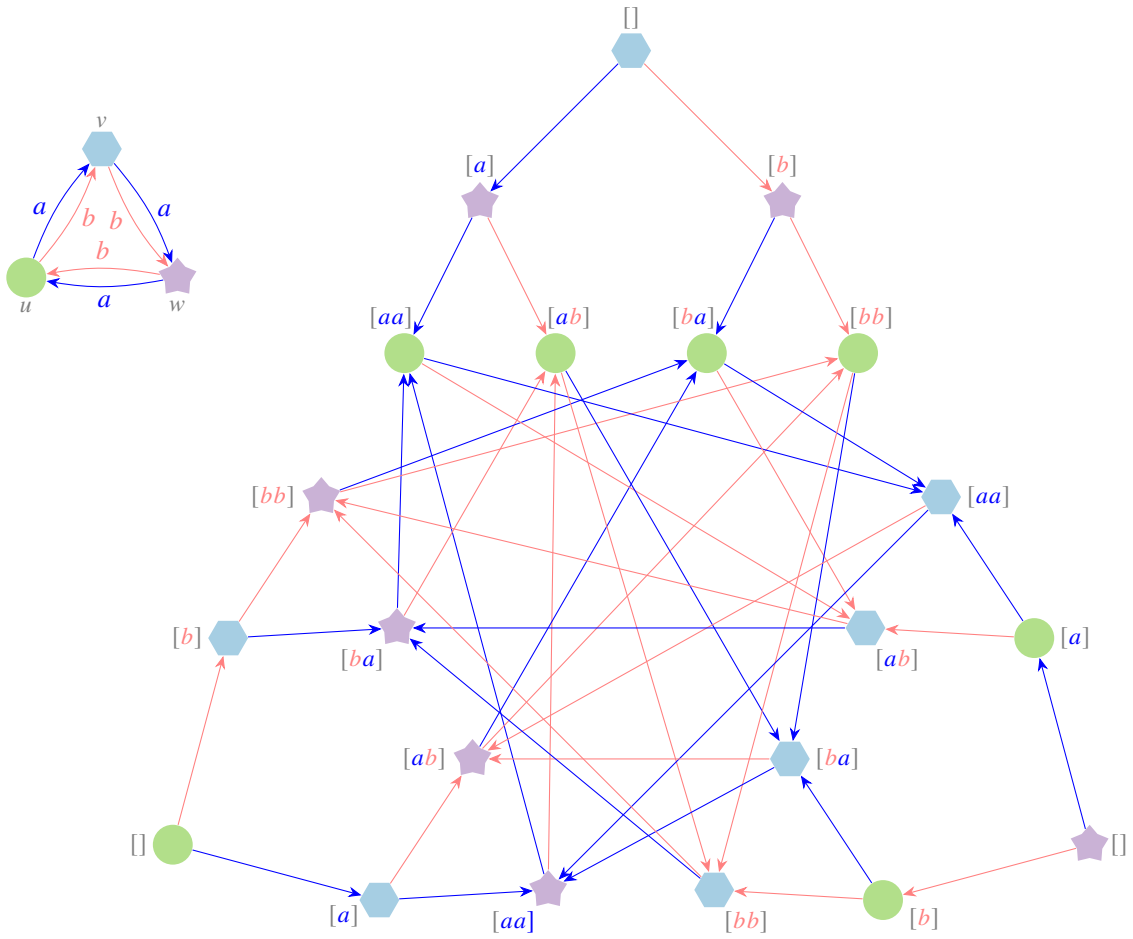


Figure 6.6: Graphs G and H = SafeRavel$_f(G)$ for f mapping each walk to its suffix of length at most 2. Note that G is very similar to the graph obtained by using the ravelling construction in Figure 6.3, but with multiple node labels, to improve the clarity of the picture. Even for such a small graph G, the structure of H becomes quite convoluted.

Let h map each walk in G_1 to its longest important suffix. We claim that h restricted to V_D is a homomorphism from D to $\mathsf{GlobalUnravel}(n, G_1)$.

Proving that h is a homomorphism

We need to show that:

- for each node π in D, $h(\pi)$ is a walk in G_1 of length at most n;
- the label of each node π in D is the same as the label of $h(\pi)$ in $\mathsf{GlobalUnravel}(n, G_1)$;
- for each edge (π_1, a, π_2) in D, $h(\pi_2) \in \mathsf{Extensions}_{G_1}(h(\pi_1), a)$.

The first two points are straightforward: the first one, because π itself is a walk in G_1 of length at most n, and $h(\pi)$ is a suffix of π; the second one, because both functions on walks (Suffix_n and h) preserve the last node of a walk, and in both constructions (the ravelling construction and the global unravelling) the labels are inherited from the last nodes.

To prove the third point, we will use the following claim; its analogues will appear in future chapters. Intuitively, it can be read as: h is a "coarsening" of Suffix_n.

Claim 6.8. For each walk π in G_1, $h(\pi) = h(\mathsf{Suffix}_n(\pi))$.

Proof. At most n different edges in G_1 are important, as D has at most n edges, and so does $h_{G_2 \to G_1}(D)$. The function h maps each walk to its longest important suffix, so it suffices to prove that $h(\pi)$ has length at most n. This holds, because otherwise $h(\pi)$ would need to traverse some (important) edge more than once, but each cycle in G_1 has length at least $n + 1$ (Claim 6.6 on page 68), so each cycle contains some unimportant edge. □

The rest of the proof boils down to unravelling the definitions. Consider an edge (π_1, a, π_2) in D, and let (σ_1, e, σ_2) be its ravelling witness; recall that this means that:

- $\mathsf{Suffix}_n(\sigma_1) = \pi_1$, $\mathsf{Suffix}_n(\sigma_2) = \pi_2$;
- $\sigma_2 = \sigma_1 \cdot e$;
- the label of e is a.

The edge e is an important edge in G_1, as the edge (π_1, a, π_2) in D is mapped by $h_{G_2 \to G_1}$ to e. Thus, $h(\sigma_2) = h(\sigma_1) \cdot e$; by the claim above, $h(\pi_2) = h(\pi_1) \cdot e$, so $h(\pi_2) \in \mathsf{Extensions}_{G_1}(h(\pi_1), a)$.

This ends the proof of Proposition 6.7: for each subgraph D of G_2 with at most n edges, we showed a homomorphism from D to $\mathsf{GlobalUnravel}(n, G_1)$. This also ends the proof of Proposition 6.5, and of the correctness of the reduction.

6.2.2 Summary

We described the following reduction from the minimal downsets problem to the satisfiability problem for the class of all graphs. Consider a modal profile $\mathcal{S}$, and let $\mathcal{D} = \mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S}))$. The reduction checks if $\mathcal{S}$ is satisfiable; if it is, the result is the singleton set $\{\mathcal{D}\}$; otherwise, the result is the empty set. To prove the correctness of the reduction, we needed to show that:

- for each graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$, $\mathcal{D} \subseteq \mathsf{Downset}_n(G)$;
- there is some graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$ such that $\mathsf{Downset}_n(G) \subseteq \mathcal{D}$.

The first statement follows from the properties of global representatives and global unravellings. To prove the second statement, we considered an arbitrary graph $G_0 \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$, used ravelling constructions to construct graphs G_1, G_2, and claimed that G_2 satisfies the required conditions. The fact that $G_2 \models \mathcal{S}$ followed from the properties of ravelling constructions – the Ravelling Profile Preservation Lemma (Lemma 5.7 on page 58). The most challenging part was showing that $\mathsf{Downset}_n(G_2) \subseteq \mathcal{D}$; that is, that every subgraph of G_2 with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}(\mathcal{S})$.

As mentioned in the introduction, the graph we define using ravelling constructions (G_2) has a very similar structure to database instances obtained by the *finite chase* procedure by Rosati (2006). It is also common to prove similar results using other combinatorial constructions, such as the *coloured blocking principle* (Gogacz and Marcinkowski 2013; Gogacz, Ibañez-García, and Murlak 2018). Instead of using existing techniques, we introduced ravelling constructions, because they generalize well to the problems considered in later chapters; they also seem to be simpler than all commonly used constructions.

To finish the proof of the decidability of the minimal downsets problem for $\mathbb{G}$, we need to show that the satisfiability problem for $\mathbb{G}$ is decidable. This will also finish the proof that the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox is decidable.

6.3 Satisfiability

To show that the satisfiability problem for $\mathbb{G}$ is decidable, we prove the following proposition.

Proposition 6.9. For every satisfiable modal profile $\mathcal{S} = (n, \mathcal{T})$, there is a graph in $\mathsf{Models}_{\mathbb{G}}(\mathcal{S})$ with $|\mathcal{T}|$ nodes.

The decidability follows, as it is enough to iterate over all (isomorphism types of) graphs with $|\mathcal{T}|$ nodes that use only labels appearing in $\mathcal{S}$, calculate their n-bisimilarity profiles, and compare them with $\mathcal{T}$.

To prove Proposition 6.9 we want to show that, for a graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$, the graph obtained from G by identifying nodes with equal n-bisimilarity types also satisfies $\mathcal{S}$. This is closely related to the notion of *filtrations* in modal logic; see, for example, (Blackburn, Rijke, and Venema 2001, Chapter 4).

At the end of this section we present a direct construction of a canonical candidate for a model of $\mathcal{S}$.

6.3.1 Existence of a small model

Consider a satisfiable modal profile $\mathcal{S} = (n, \mathcal{T})$, and a graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$.

Let $\mathcal{E} = \{(\mathsf{Type}^n_G(u), a, \mathsf{Type}^n_G(v)) : (u, a, v) \in E_G\}$.

Let H be the graph with nodes $\mathcal{T}$ and edges $\mathcal{E}$. The label of a node τ is $\mathsf{RootLabel}(\tau)$.

Claim 6.10. For each node τ in H, $\mathsf{Type}^n_H(\tau) = \tau$.

Proof. By the One-Step Bisimulation Lemma applied to the function SupposedType being the identity, we only need to show that, for each node τ in H and each edge label a:

$$\mathsf{Subtypes}(\tau, a) = \left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_H(\tau, a)\right\}.$$

We show two set inclusions.

First we prove that $\left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_H(\tau, a)\right\} \subseteq \mathsf{Subtypes}(\tau, a)$. For each edge (u, a, v) in G, by the definition of the n-bisimilarity type of u in G:

$$\mathsf{Type}_G^{n-1}(v) \in \mathsf{Subtypes}(\mathsf{Type}_G^n(u), a).$$

Thus, for each $(\tau, a, \theta) \in \mathcal{E}$, $\theta|_{n-1} \in \mathsf{Subtypes}(\tau, a)$. Since $\mathcal{E}$ is the set of edges in H, this proves the set inclusion.

Now we show the other inclusion, $\mathsf{Subtypes}(\tau, a) \subseteq \left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_H(\tau, a)\right\}$. Consider a node u in G such that $\mathsf{Type}_G^n(u) = \tau$; by the definition of the n-bisimilarity type of u in G:

$$\mathsf{Subtypes}(\tau, a) = \{\mathsf{Type}_G^{n-1}(v) : v \in \mathsf{Succ}_G(u, a)\}.$$

Thus, we need to show that, for each node $v \in \mathsf{Succ}_G(u, a)$ there is some node $\theta \in \mathsf{Succ}_H(\tau, a)$ such that $\theta|_{n-1} = \mathsf{Type}_G^{n-1}(v)$. By the construction of H, this holds for $\theta = \mathsf{Type}_G^n(v)$, as $(\tau, a, \theta) \in \mathcal{E}$, which is witnessed by the edge (u, a, v). □

This finishes the proof of Proposition 6.9, and of the decidability of the satisfiability problem for the class $\mathbb{G}$ of all graphs.

6.3.2 A canonical candidate for a model

Below we present an alternative construction of a small model of a modal profile. It provides additional intuitions about the properties of modal profiles, and we will use a similar type of construction in the context of optimizing the time complexity of the obtained algorithms.

Consider a modal profile $\mathcal{S} = (n, \mathcal{T})$. Let $G_\mathcal{S}$ be the graph with nodes $\mathcal{T}$ and an edge (τ, a, θ) iff $\theta|_{n-1} \in \mathsf{Subtypes}(\tau, a)$. The label of a node τ is $\mathsf{RootLabel}(\tau)$.

Note that $G_\mathcal{S}$ has the same set of nodes as the graph H constructed above, but might have more edges; intuitively, $G_\mathcal{S}$ can be seen as the maximal model of $\mathcal{S}$ with the set of nodes $\mathcal{T}$.

Claim 6.11. If $\mathcal{S}$ is satisfiable, then $G_\mathcal{S}$ satisfies $\mathcal{S}$.

Proof. The proof is almost identical to the one above. We use the One-Step Bisimulation Lemma for SupposedType being the identity. For each node τ in $G_\mathcal{S}$, $\mathsf{Label}_{G_\mathcal{S}}(\tau) = \mathsf{RootLabel}(\tau)$, so we only need to show that, for each edge label a, $\mathsf{Subtypes}(\tau, a) = \left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_{G_\mathcal{S}}(\tau, a)\right\}$.

First we prove that $\left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_{G_\mathcal{S}}(\tau, a)\right\} \subseteq \mathsf{Subtypes}(\tau, a)$. By the construction of $G_\mathcal{S}$, for each edge (τ, a, θ) we have $\theta|_{n-1} \in \mathsf{Subtypes}(\tau, a)$.

Next we prove that $\mathsf{Subtypes}(\tau, a) \subseteq \left\{\theta|_{n-1} : \theta \in \mathsf{Succ}_{G_\mathcal{S}}(\tau, a)\right\}$. Since $\mathcal{S}$ is satisfiable, there exists some graph G that satisfies $\mathcal{S}$. Consider some node u in G such that $\mathsf{Type}_G^n(u) = \tau$. For

each $v \in \mathsf{Succ}_G(u,a)$, $\mathsf{Type}_G^{n-1}(v) \in \mathsf{Subtypes}(\mathsf{Type}_G^n(u), a)$. Thus, in G_S there is an edge $(\mathsf{Type}_G^n(u), a, \mathsf{Type}_G^n(v))$. □

6.4 Summary

In this chapter we solved the minimal downsets problem for the class $\mathbb{G}$ of all graphs, by reducing it to the satisfiability problem. This finishes the proof of the decidability of the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox. The proof consists of a chain of reductions, between the following problems:

1. the problem of conjunctive query containment modulo an $\mathcal{ALCO}$ TBox;
2. the homomorphism entailment problem for $\mathbb{G}$;
3. the homomorphism coverage problem for $\mathbb{G}$;
4. the minimal downsets problem for $\mathbb{G}$;
5. the satisfiability problem for $\mathbb{G}$.

The same series of reductions will be used to prove the main result of this thesis, for different classes of graphs. Let us briefly summarize the reductions, and highlight the three main combinatorial constructions – some of which are quite simple in the presented setting, but become more complex in later chapters.

The reduction from (1) to (2), presented in Chapter 3, is mostly a straightforward translation from the setting of description logics to the graph setting.

The reduction from (2) to (3), presented in Chapter 4, uses the notion of *detachments* of graphs, and relies on the proof that, intuitively, a modal profile and a homomorphism from a given graph can be "distributed" over an inner graph of bounded size, and the remaining outer graph. This is the first main combinatorial construction, which does not change much in the proof of the main theorem of this thesis, although the proofs become significantly more complex.

The reduction from (3) to (4), presented in Chapter 5 for an arbitrary class of graphs, used the notion of *k-deepenings* of a modal profile: all modal profiles of depth k that are "compatible" with the given modal profile.

The reduction from (4) to (5), presented in this chapter, relied on defining *global representatives* of modal profiles. If a modal profile is satisfiable, then the downset of its global representative coincides with the smallest downset of models of $\mathcal{S}$; the second main combinatorial construction uses ravelling constructions to obtain, from an arbitrary model of $\mathcal{S}$, a model with the smallest downset.

To solve the satisfiability problem for $\mathbb{G}$, we proved that every satisfiable modal profile $\mathcal{S}$ has a small model, which can be obtained from an arbitrary model by identifying all nodes with equal n-bisimilarity types, where n is the depth of $\mathcal{S}$. Constructing a small model from an arbitrary model is the third main combinatorial construction.

The core techniques used to prove the main theorem of this thesis are related to the minimal downsets problem – reducing it to the satisfiability problem, proving that the reduction is correct, and solving the satisfiability problem – while the rest of the reductions remain mostly unchanged. For this reason, in the intermediate results in Chapter 7 and Chapter 8 we focus on the minimal downsets problem.

Chapter 7

Reachability-annotated graphs

Let us fix three edge labels: Short, Long, and $\bot$. For a graph G, the *reachability annotation* of G, denoted G^+, is the graph obtained from G by replacing the label a of each edge with (Short, a), and adding an edge from u to v labelled $(\mathsf{Long}, \bot)$ iff v is reachable from u in G. An example of a graph and its reachability annotation is shown in Figure 7.1.

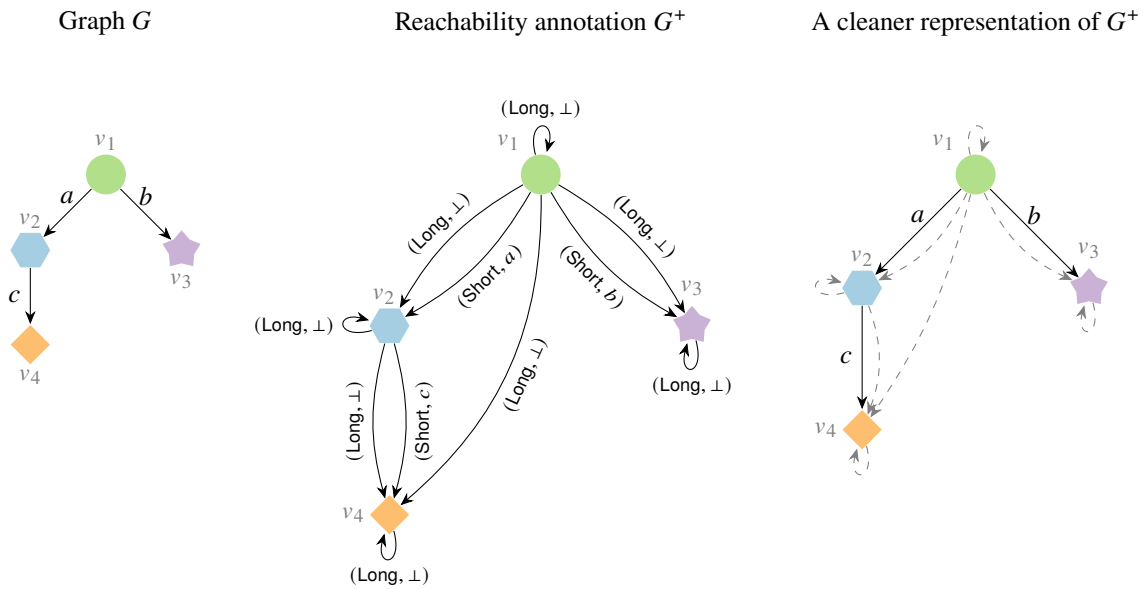


Figure 7.1: An example of graphs G and G^+, along with a representation of G^+ in which labels $(\mathsf{Long}, \bot)$ are omitted, the corresponding edges are dashed and gray, and edge labels of the form (Short, a) are replaced with a.

Let $\mathbb{G}_+ = \{G^+ : G \in \mathbb{G}\}$ be the class of all reachability annotations of graphs. In this chapter we solve the minimal downsets problem for $\mathbb{G}_+$.

To familiarize the reader with reachability annotations, we begin by solving the satisfiability problem for $\mathbb{G}_+$. Next, we will show why the approach presented in the previous chapter fails: that is, it is impossible to define an analogue of the global representative of a modal profile in the class $\mathbb{G}_+$, as there

exists a modal profile $\mathcal{S}$ such that $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S})$ contains more than one element. To solve this issue, we will consider yet another class of graphs, $\mathbb{G}_*$, annotated additionally with some information about strongly connected components. Then, we will prove the following propositions.

Proposition 7.1. There is a Turing reduction from the minimal downsets problem for $\mathbb{G}_+$ to the satisfiability problem for $\mathbb{G}_*$.

Proposition 7.2. The satisfiability problem for $\mathbb{G}_*$ is decidable.

The techniques introduced in this chapter are some of the core technical contributions of this thesis. However, the results are not used directly to prove the main result of the thesis – the purpose of this chapter, similarly to the previous one, is to introduce and discuss techniques and constructions in a simpler setting.

Because we work with several classes of graphs, we use subscripts to make it clear that an object is related to one of these classes; for example, we might write G_+ to refer to some graph in $\mathbb{G}_+$, and a_+ to refer to an edge label of the form (Short, a) or $(\mathsf{Long}, \bot)$.

7.1 Warm-up: satisfiability for reachability annotations

In this section we solve the satisfiability problem for $\mathbb{G}_+$, by proving the following proposition.

Proposition 7.3. If a modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$ is satisfiable within $\mathbb{G}_+$, then there is a graph in $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ with $|\mathcal{T}_+|$ nodes.

The decidability of the satisfiability problem for $\mathbb{G}_+$ follows, as it is enough to iterate over all (isomorphism types of) graphs G with $|\mathcal{T}_+|$ nodes and appropriate labels (node labels appearing in $\mathcal{S}_+$ and edge labels a such that (Short, a) appears in $\mathcal{S}_+$), compute G^+, calculate the n-bisimilarity profile of G^+ and compare it with $\mathcal{T}_+$.

7.1.1 Constructing a small model

The idea of the construction is essentially the same as in the previous chapter – identifying all nodes with equal types – but this time we need to distinguish between the underlying graph G and its reachability annotation G^+. We identify nodes in G, but we are interested in n-bisimilarity types of nodes in G^+: we obtain a small model by considering an arbitrary graph G such that $G^+ \models \mathcal{S}_+$, identifying all nodes u, v such that $\mathsf{Type}^n_{G^+}(u) = \mathsf{Type}^n_{G^+}(v)$, and taking the reachability annotation of the resulting graph. The most important difference in the proof is that it relies on the transitivity of reachability: for an edge (u, a, v) in a graph G, all nodes reachable from v are also reachable from u, which, in terms of types, means that:

$$\mathsf{Subtypes}(\mathsf{Type}^n_{G^+}(v), (\mathsf{Long}, \bot)) \subseteq \mathsf{Subtypes}(\mathsf{Type}^n_{G^+}(u), (\mathsf{Long}, \bot)).$$

Consider a modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$ that is satisfiable within $\mathbb{G}_+$, and let G be a graph such that $G^+ \models \mathcal{S}_+$. Let $\mathcal{E}_+ = \{(\mathsf{Type}^n_{G^+}(u), a, \mathsf{Type}^n_{G^+}(v)) : (u, a, v) \in E_G\}$; note that we iterate over edges in

G, not in G^+. Let H be the graph with nodes $\mathcal{T}_+$ and edges $\mathcal{E}_+$; the label of a node τ_+ is $\mathsf{RootLabel}(\tau_+)$.

Claim 7.4. For each node τ_+ in H, $\mathsf{Type}^n_{H^+}(\tau_+) = \tau_+$.

Proof. We use the One-Step Bisimulation Lemma for the graph H^+ and the function $\mathsf{SupposedType}$ being the identity. Since the label of τ_+ is $\mathsf{RootLabel}(\tau_+)$ in H and in H^+, we only need to show that, for each node τ_+ in H^+ and each edge label a_+:

$$\mathsf{Subtypes}(\tau_+, a_+) = \left\{\theta_+|_{n-1} : \theta_+ \in \mathsf{Succ}_{H^+}(\tau_+, a_+)\right\}.$$

We prove two set inclusions; we consider the case of $a_+ = (\mathsf{Long}, \bot)$ separately for each inclusion.

First we show that $\mathsf{Subtypes}(\tau_+, a_+) \subseteq \left\{\theta_+|_{n-1} : \theta_+ \in \mathsf{Succ}_{H^+}(\tau_+, a_+)\right\}$. Consider a node u in G such that $\mathsf{Type}^n_{G^+}(u) = \tau_+$. Intuitively, we need to show that for each edge outgoing from u in G^+ there is a corresponding edge outgoing from τ_+ in H^+. Consider an edge (u, a_+, v) in G^+ for some node v, and let $\theta_+ = \mathsf{Type}^n_{G^+}(v)$. We consider two cases.

- If $a_+ = (\mathsf{Short}, a)$ for some a, then $(u, a, v) \in E_G$, so $(\tau_+, a, \theta_+) \in \mathcal{E}_+ = E_H$, so $(\tau_+, a_+, \theta_+) \in E_{H^+}$.
- If $a_+ = (\mathsf{Long}, \bot)$, there is a walk in G from u to v; replacing each node w in this walk with $\mathsf{Type}^n_{G^+}(w)$ results in a walk from τ_+ to θ_+ in H, by the same argument as above, so $(\tau_+, (\mathsf{Long}, \bot), \theta_+)$ is an edge in H^+.

To prove the other set inclusion, we will use the following fact. For each edge (τ_+, a, θ_+) in H there are some corresponding edges (u, a, v) in G and $(u, (\mathsf{Short}, a), v)$ in G^+ such that $\mathsf{Type}^n_{G^+}(u) = \tau_+$ and $\mathsf{Type}^n_{G^+}(v) = \theta_+$, so we have:

- $\theta_+|_{n-1} \in \mathsf{Subtypes}(\tau_+, (\mathsf{Short}, a))$;
- $\mathsf{Subtypes}(\theta_+, (\mathsf{Long}, \bot)) \subseteq \mathsf{Subtypes}(\tau_+, (\mathsf{Long}, \bot))$.

The second condition holds because every node in G reachable from v is also reachable from u.

Now, for the other set inclusion, for each $\theta_+ \in \mathsf{Succ}_{H^+}(\tau_+, a_+)$ we show that $\theta_+|_{n-1} \in \mathsf{Subtypes}(\tau_+, a_+)$.

- If $a_+ = (\mathsf{Short}, a)$, this is exactly the first condition above.
- If $a_+ = (\mathsf{Long}, \bot)$, there is a walk in H from τ_+ to θ_+. Applying the inclusion from the second condition above to each edge traversed by this walk, we get

  $$\mathsf{Subtypes}(\theta_+, (\mathsf{Long}, \bot)) \subseteq \mathsf{Subtypes}(\tau_+, (\mathsf{Long}, \bot)).$$

  Since each node in G is reachable from itself, $\theta_+|_{n-1} \in \mathsf{Subtypes}(\theta_+, (\mathsf{Long}, \bot))$.

□

This finishes the proof of Proposition 7.3, and of the decidability of the satisfiability problem for $\mathbb{G}_+$.

7.1.2 A canonical candidate for a model

For a modal profile $\mathcal{S}_+$, let $G_{\mathcal{S}_+}$ be the graph with nodes $\mathcal{T}_+$ and an edge (τ_+, a, θ_+) iff

- $\theta_+|_{n-1} \in \mathsf{Subtypes}(\tau_+, (\mathsf{Short}, a))$, and
- $\mathsf{Subtypes}(\theta_+, (\mathsf{Long}, \bot)) \subseteq \mathsf{Subtypes}(\tau_+, (\mathsf{Long}, \bot))$.

The label of a node τ_+ is $\mathsf{RootLabel}(\tau_+)$.

Claim 7.5. If $\mathcal{S}_+$ is satisfiable within $\mathbb{G}_+$, then $(G_{\mathcal{S}_+})^+ \models \mathcal{S}_+$.

We leave this claim without a proof, as the reasoning is almost identical to the one above, and this is not a part of the main result.

7.2 No global representatives

In the previous chapter we proved that, for each satisfiable modal profile $\mathcal{S}$ of depth n, $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S})$ is a singleton $\{\mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S}))\}$. In this section we argue that there exists a modal profile $\mathcal{S}_+$ for which $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ contains more than one element, so there is no analogue of the global representative for the class $\mathbb{G}_+$. Unfortunately, modal profiles $\mathcal{S}_+$ witnessing it (that we are aware of) are quite large; we will show such a witness, but without a full analysis of its correctness, as it would be very tedious. Instead, we begin by discussing simpler examples that illustrate the main idea.

Note that the purpose of this section is to justify the need to introduce more elaborate techniques, but this discussion is not necessary for the main result of this chapter.

No monotonicity: enforcing a cycle

First, we show the lack of a kind of monotonicity: intuitively, a "larger" modal profile might have "smaller" minimal downsets. To keep the examples simple, we formalize this statement in a relatively weak form. There are modal profiles $\mathcal{S}_+$ and $\mathcal{S}'_+$ of equal depth n such that:

- $\mathsf{GlobalRepresentative}(\mathcal{S}_+) \to \mathsf{GlobalRepresentative}(\mathcal{S}'_+)$ ($\mathcal{S}'_+$ is "larger" than $\mathcal{S}_+$);
- there is some $\mathcal{D}' \in \mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}'_+)$ such that, for all $\mathcal{D} \in \mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$, $\mathcal{D} \not\subseteq \mathcal{D}'$ ($\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}'_+)$ is not "larger" than $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$).

We note that this kind of monotonicity holds for the class $\mathbb{G}$ considered in the previous chapter.

Consider the modal profile $\mathcal{S}_+$ of depth 2 of the graph G^+ presented in Figure 7.2, and the modal profile $\mathcal{S}'_+$ of depth 2 of the graph H^+ presented in Figure 7.3. It is easy to see that $\mathcal{S}'_+$ is "larger" than $\mathcal{S}_+$. We claim that every graph in $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ has a cycle containing nodes with labels ⬣ and ★; this is because every node with label ⬣ has a successor with label ★, and every node with label ★ has a successor with label ⬣, and we work exclusively with finite graphs. At the same time, the graph H^+ does not have such a cycle, so $\mathcal{S}'_+$ does not enforce it. Because such a cycle can be represented by a graph with two nodes and two long edges between them, this means that $\mathsf{Downset}_2(H^+)$ does not contain any downset in $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$.

Enforcing one of two cycles

Consider graphs G and H presented in Figure 7.4, and the shallow modal profile $\mathcal{S}_+$ of G^+ and H^+. We claim that every graph in $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ has either a cycle containing nodes with labels ● and ★ (as u_1 and u_3 in G), or with labels ● and ⬣ (as v_1 and v_5 in H). This is due to the following reasoning. Consider a graph D such that $D^+ \models \mathcal{S}_+$; when referring to "types" of nodes, we always mean their 1-bisimilarity type in D^+, but "successors" concern the graph D (or only short edges in D^+).

- There is a node u of type τ_1.
- All successors of u have labels ★ or ⬣, and there is a node with label ◆ reachable from u.

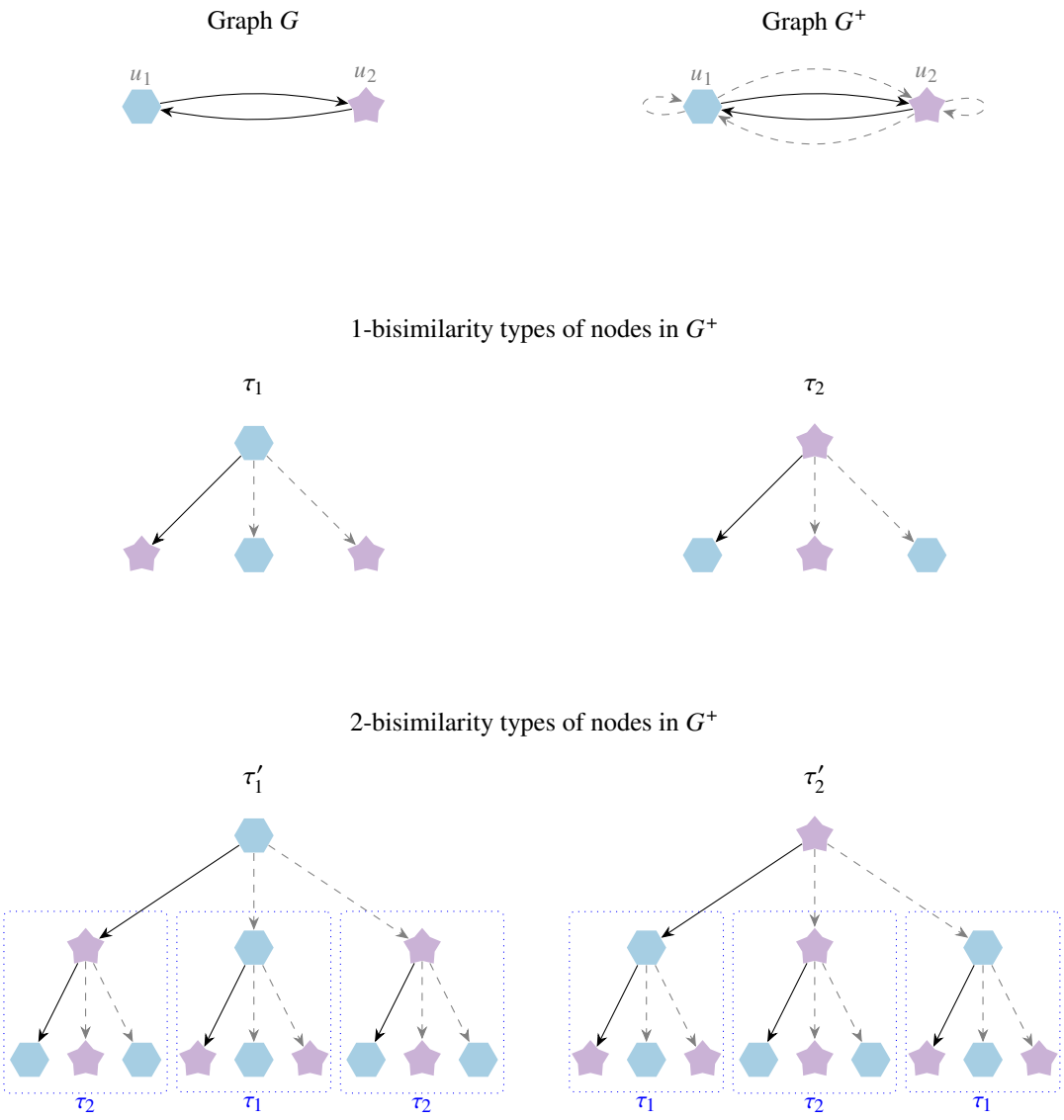


Figure 7.2: Graphs G and G^+, and modal profiles of G^+ of depth 1 and 2. All edges in G have the same label a; we omit it in the picture.

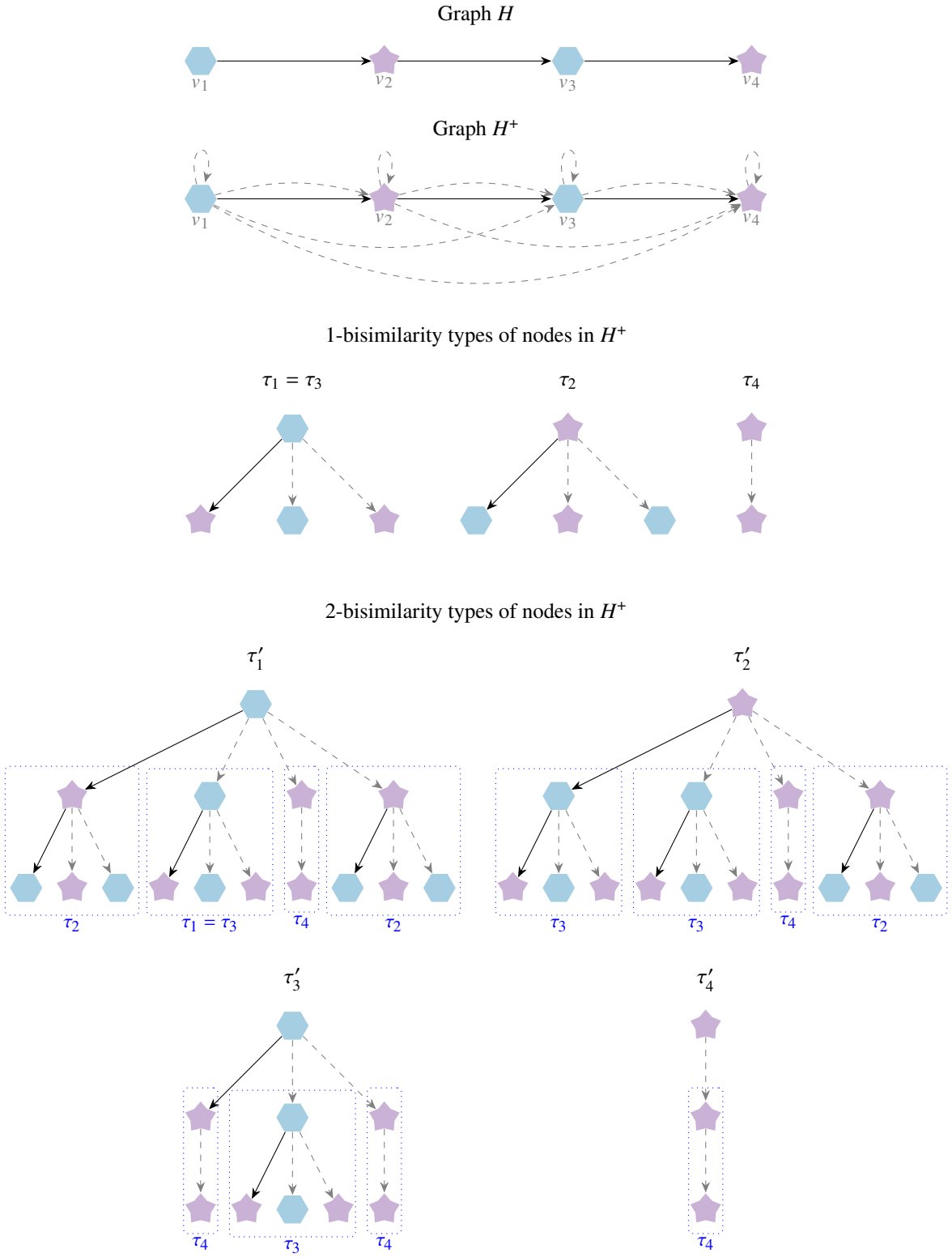


Figure 7.3: Graphs H and H^+, and modal profiles of H^+ of depth 1 and 2. All edges in H have the same label a; we omit it in the picture.

- There is a successor v of u of type τ_3 or τ_5, as otherwise all successors have types τ_2 or τ_4 and there is no node with label ◆ reachable from u.
- In both cases, v has a successor with label ●; its type is τ_1.

Similarly to the previous example, because we work with finite graphs, this means that there is a cycle containing nodes with labels ● and ★ or with labels ● and ⬢. However, graphs G^+ and H^+ show that there are models of $\mathcal{S}_+$ without both such cycles at the same time. This is very close to our goal of $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ containing more than one element; however, $\mathcal{S}_+$ has depth 1, so downsets in $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ consist of graphs with at most one edge, while graphs representing such cycles have two edges. The example can be extended to a modal profile of depth 2, as presented in Figure 7.5, but we omit the proof of its correctness.

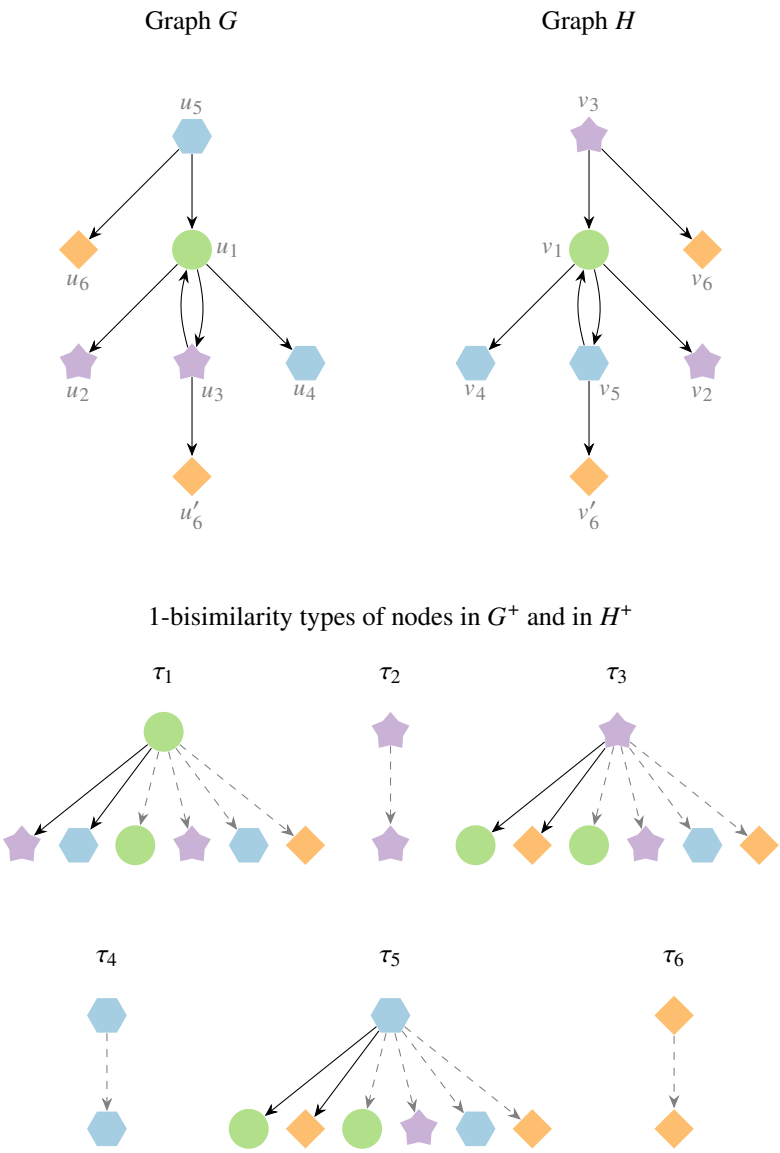


Figure 7.4: Graphs G and H, and the shallow modal profile of their reachability annotations. The graph H is obtained from G by swapping all node labels ⬢ and ★; the nodes are also renamed to correspond to bisimilarity types.

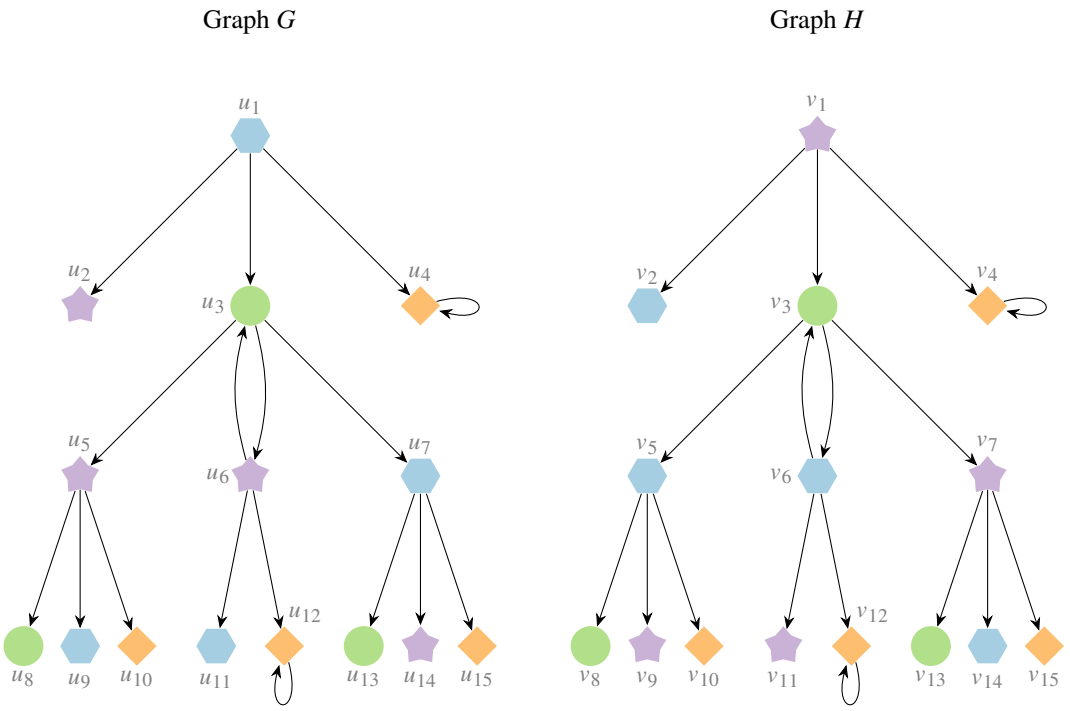


Figure 7.5: An example of graphs G and H; the graph H is obtained from G by swapping all node labels ⬢ and ★. We claim that G^+ and H^+ satisfy the same modal profile $\mathcal{S}_+$ of depth 2, and that in each graph in $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ there is either a cycle containing nodes with labels ● and ★ (as u_3 and u_6 in G), or with labels ● and ⬢ (as v_3 and v_6 in H). Consequently, $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ contains at least two elements.

7.3 Basic definitions

7.3.1 Reachability annotations

As defined earlier, the reachability annotation of a graph G, denoted G^+, is the graph obtained from G by replacing the label a of each edge with (Short, a), and adding an edge from u to v labelled $(\mathsf{Long}, \bot)$ iff v is reachable from u in G. Recall that $\mathbb{G}_+ = \{G^+ : G \in \mathbb{G}\}$ is the class of all reachability annotations of graphs.

Short-long graphs

A *long edge* is an edge labelled $(\mathsf{Long}, \bot)$, and a *short edge* is an edge labelled (Short, a) for some a. A *short-long graph* is a graph in which each edge is either short or long. A *short-long modal profile* is a modal profile in which each edge label is either $(\mathsf{Long}, \bot)$ or (Short, a) for some a.

Trace-based graph transformations

For a fixed graph G, its reachability annotation can be defined using a trace-based graph transformation. We refer the reader to the definition of trace-based graph transformations in Section 5.4 on page 59.

For a graph G, let $k = |V_G|$, let g_Γ be the identity function on node labels, and let g_Σ be as follows:

- for a trace $L = (A, a, B)$ of walk-length exactly 1, $g_\Sigma(L) = \{(\mathsf{Short}, a), (\mathsf{Long}, \bot)\}$;
- for a trace of walk-length different than 1, $g_\Sigma(L) = \{(\mathsf{Long}, \bot)\}$.

Then, the trace-based graph transformation $\tau = \mathsf{TraceMap}^g_k$ transforms G into G^+. The bound k on the length of relevant traces follows from the fact that, if a node is reachable from another node in G, there is a walk between them that visits each node in G at most once.

The above transformation does not depend on the structure of the graph G, only the value of k depends on the number of its nodes. Thus, for a positive integer k it is possible to define a single trace-based graph transformation τ that, given a graph G with at most k nodes, produces the graph G^+. By applying Fact 5.8 on page 60 and Corollary 5.12 on page 61 to such a transformation, we get the following.

Fact 7.6. For graphs G, H, every homomorphism from G to H is a homomorphism from G^+ to H^+.

Fact 7.7. For a graph G, a G-ravelling function f, and $H = \mathsf{Ravel}_f(G)$, the n-bisimilarity profiles of G^+ and H^+ are equal for each non-negative integer n.

7.3.2 SCC-annotations

Strongly connected components

Consider a graph G. Two nodes u, v are *strongly connected* if v is reachable from u and u is reachable from v. A *strongly connected component (SCC)* in G is a maximal subgraph of G in which each pair of nodes is strongly connected.

Each node in a graph is contained in exactly one SCC, and each edge is contained in at most one SCC. For a node u in a graph G, let $\mathsf{SCC}_G(u)$ denote the SCC in G that contains u.

SCC-annotations

The *SCC-annotation* of a graph G, denoted $G^\times$, is the graph obtained from G by replacing the label a of each edge e with $(a, \mathsf{InsideSCC})$ if e is contained in some SCC in G, or with $(a, \mathsf{OutsideSCC})$ otherwise. An example of a graph and its SCC-annotation is shown in Figure 7.6.

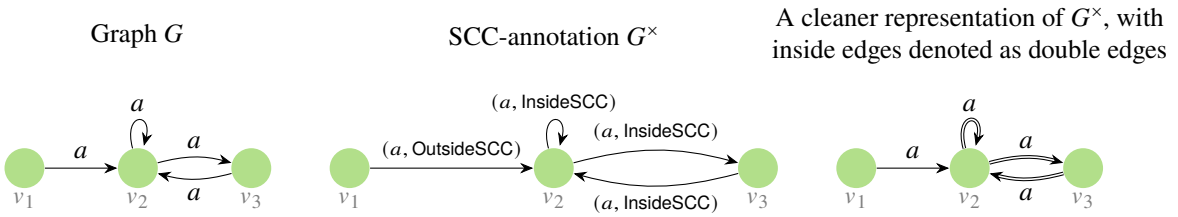


Figure 7.6: An example of graphs G and $G^\times$.

Let $\mathbb{G}_\times = \{G^\times : G \in \mathbb{G}\}$ be the class of all SCC-annotations of graphs.

Fact 7.8. For nodes u, v in a graph G and a non-negative integer n, if $\mathsf{Type}^n_{G^\times}(u) = \mathsf{Type}^n_{G^\times}(v)$, then $\mathsf{Type}^n_G(u) = \mathsf{Type}^n_G(v)$.

No trace-based graph transformation

Note that the SCC-annotation of a graph cannot be defined as a trace-based graph transformation. Consider the graph G in Figure 7.6: the set of traces of all walks from v_1 to v_2 in G is equal to the set of traces of all walks from v_3 to v_2 in G, which, after applying a trace-based graph transformation, would result in the same labels of edges from v_1 to v_2 and from v_3 to v_2 in the resulting graph. Meanwhile, in $G^\times$, the edge from v_1 to v_2 has label $(a, \mathsf{OutsideSCC})$, while the edge from v_3 to v_2 has label $(a, \mathsf{InsideSCC})$.

Inside-outside graphs

An *inside edge* is an edge labelled $(a, \mathsf{InsideSCC})$ for some a, and an *outside edge* is an edge labelled $(a, \mathsf{OutsideSCC})$ for some a. An *inside-outside graph* has only inside and outside edges. An *inside-outside modal profile* uses only edge labels of the form $(a, \mathsf{InsideSCC})$ and $(a, \mathsf{OutsideSCC})$.

Deannotations

The *deannotation* of an edge label of the form $(a, \mathsf{InsideSCC})$ or $(a, \mathsf{OutsideSCC})$ is a. The deannotation of an inside-outside graph is obtained by replacing all edge labels with their deannotations. The deannotation of an inside-outside modal profile is obtained by replacing all edge labels with their deannotations.

Inside-outside annotations of modal profiles

A modal profile $\mathcal{S}_\times$ is an *inside-outside annotation* of a modal profile $\mathcal{S}$ if $\mathcal{S}$ is the deannotation of $\mathcal{S}_\times$.

Fact 7.9. For every modal profile $\mathcal{S}$, the set of all inside-outside annotations of $\mathcal{S}$ is finite and computable.

Fact 7.10. For every modal profile $\mathcal{S}$ and every graph G, $G \models \mathcal{S}$ iff $G^\times \models \mathcal{S}_\times$ for some inside-outside annotation $\mathcal{S}_\times$ of $\mathcal{S}$.

Weak homomorphisms

Consider graphs G, H presented in Figure 7.7. There is a homomorphism from G to H, but there is no homomorphism from $G^\times$ to $H^\times$, because the edge in $G^\times$ has label $(a, \mathsf{OutsideSCC})$, and both edges in $H^\times$ have label $(a, \mathsf{InsideSCC})$. We define a notion of a *weak homomorphism* between inside-outside graphs such that, for all graphs G, H, there is a weak homomorphism from $G^\times$ to $H^\times$ iff $G \to H$; there are multiple ways to define such a notion, and the one we choose will be useful in the context of global representatives of inside-outside modal profiles.

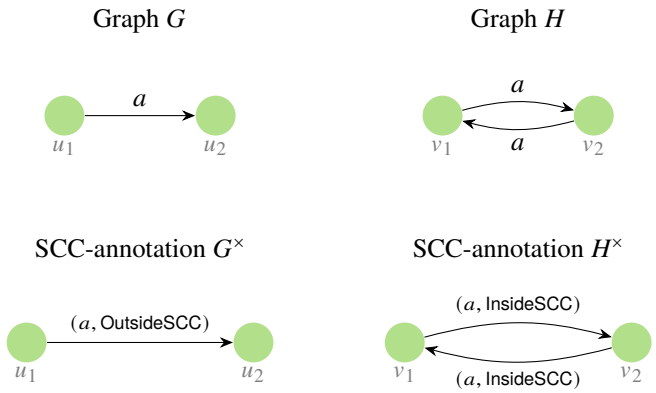


Figure 7.7: An example of graphs G, H and their SCC-annotations $G^\times, H^\times$ such that G maps homomorphically to H, but $G^\times$ does not map homomorphically to $H^\times$.

For two inside-outside graphs $G_\times$ and $H_\times$, a *weak homomorphism* from $G_\times$ to $H_\times$ is, intuitively, a homomorphism from $G_\times$ to $H_\times$ that is additionally allowed to map outside edges to inside edges, but not the other way around. Formally, it is a function h from $V_{G_\times}$ to $V_{H_\times}$ such that:

- for each node u in $G_\times$, $\mathsf{Label}_{G_\times}(u) = \mathsf{Label}_{H_\times}(h(u))$;
- for each inside edge $(u, (a, \mathsf{InsideSCC}), v) \in E_{G_\times}$, $(h(u), (a, \mathsf{InsideSCC}), h(v)) \in E_{H_\times}$;
- for each outside edge $(u, (a, \mathsf{OutsideSCC}), v) \in E_{G_\times}$, either:
  - $(h(u), (a, \mathsf{OutsideSCC}), h(v)) \in E_{H_\times}$, or
  - $(h(u), (a, \mathsf{InsideSCC}), h(v)) \in E_{H_\times}$.

Claim 7.11. For graphs G, H, a function $h : V_G \to V_H$ is a homomorphism from G to H iff it is a weak homomorphism from $G^\times$ to $H^\times$.

Proof. The right-to-left implication holds because the graphs G, H are deannotations of $G^\times, H^\times$, and a weak homomorphism differs from a homomorphism only in the conditions concerning the second components of edge labels. To prove the left-to-right implication, we need to show that h maps each edge (u, a, v) inside an SCC in G to an edge inside an SCC in H. This follows from the fact that h can

be lifted to walks, mapping a walk from v to u in G to a walk from $h(v)$ to $h(u)$ in H, which shows that the edge $(h(u), a, h(v))$ is inside an SCC in H. □

Remark. We could define the SCC-annotation of a graph G differently, as the graph obtained from G by replacing each edge (u, a, v) with $(u, (a, \mathsf{InsideOrOutsideSCC}), v)$, and adding an edge $(u, (a, \mathsf{InsideSCC}), v)$ if (u, a, v) is inside some SCC in G. Then we would not need the notion of a weak homomorphism, as it would correspond to the usual notion of a homomorphism. We chose the current definition for three reasons:

- there is a bijection between edges in G and in $G^\times$, so it is easier to think about unravellings of $G^\times$;
- the meaning of the annotations seems more natural;
- modal profiles of $G^\times$ contain more information about the structure of G, so some results we prove are stronger.

However, the author believes that the other definition would harmonize better with other techniques we are using.

7.3.3 SCC-reachability annotations

The *SCC-reachability annotation* of a graph G, denoted G^*, is defined as $(G^+)^\times$. An example of a graph and its SCC-reachability annotation is shown in Figure 7.8.

Let $\mathbb{G}_* = \{G^* : G \in \mathbb{G}\}$ be the class of all SCC-reachability annotations of graphs.

Inside-outside short-long graphs

An *inside-outside short-long graph* is an inside-outside graph whose deannotation is a short-long graph. An *inside-outside short-long modal profile* is an inside-outside modal profile whose deannotation is a short-long modal profile.

From Fact 7.6 on page 85 and Claim 7.11, we get the following.

Fact 7.12. For graphs G, H, there is a weak homomorphism from G^* to H^* iff $G \to H$.

In an inside-outside short-long graph G_*, we use the term *long edge* to refer to an edge with a label $((\mathsf{Long}, \bot), z)$ for some $z \in \{\mathsf{InsideSCC}, \mathsf{OutsideSCC}\}$, and the term *short edge* to refer to an edge with a label $((\mathsf{Short}, a), z)$ for some $z \in \{\mathsf{InsideSCC}, \mathsf{OutsideSCC}\}$ and some a.

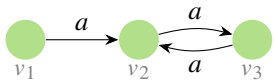
SCC Invariance Under Annotations

The fact below states that SCCs of a graph G coincide with SCCs of its annotations; we refer to it as *SCC Invariance Under Annotations*.

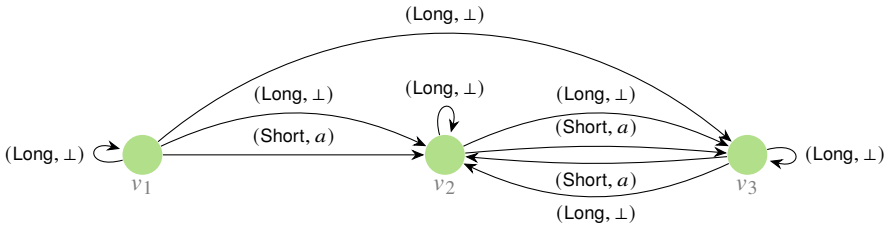
Fact 7.13. For nodes u, v in a graph G, the following are equivalent:

- $\mathsf{SCC}_G(u) = \mathsf{SCC}_G(v)$;
- $\mathsf{SCC}_{G^+}(u) = \mathsf{SCC}_{G^+}(v)$;
- $\mathsf{SCC}_{G^\times}(u) = \mathsf{SCC}_{G^\times}(v)$;
- $\mathsf{SCC}_{G^*}(u) = \mathsf{SCC}_{G^*}(v)$.

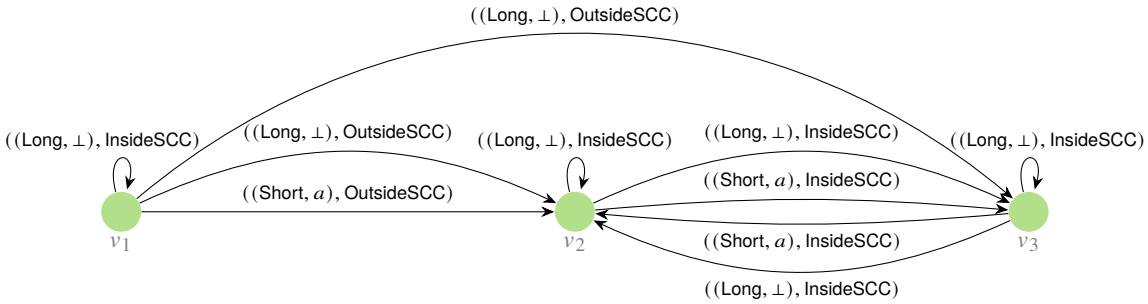
Graph G



Reachability annotation G^+



SCC-reachability annotation G^*



A cleaner representation of G^*

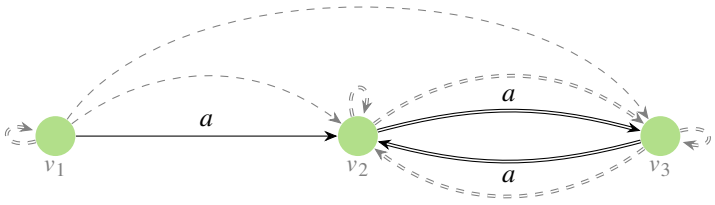


Figure 7.8: An example of graphs G, G^+ and G^*, along with a cleaner visual representation of G^*, using double edges to denote inside edges, and gray dashed edges to denote long edges.

7.4 Reduction to satisfiability

7.4.1 Overview

The previous algorithm

In the previous chapter, we described the following algorithm for computing $\mathsf{MinDownsets}_{\mathbb{G}}(\mathcal{S})$ for a given modal profile $\mathcal{S}$ of depth n. If $\mathcal{S}$ is not satisfiable, the result is the empty set. Otherwise, the result is $\{\mathsf{Downset}_n(\mathsf{GlobalRepresentative}(\mathcal{S}))\}$; to prove it, we showed that $\mathsf{GlobalRepresentative}(\mathcal{S}) \to G$ for each $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$, and used ravelling constructions to define a graph $G \in \mathsf{Models}_{\mathbb{G}}(\mathcal{S})$ such that each subgraph of G with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}(\mathcal{S})$.

We showed that it is impossible to define an analogue of the global representative for reachability annotations, and we claim that the additional information about strongly connected components in SCC-reachability annotations circumvents this issue.

Idea: reduction to the minimal downsets problem for SCC-reachability annotations

A natural idea would be to solve the minimal downsets problem for $\mathbb{G}_*$. Then, the minimal downsets problem for $\mathbb{G}_+$ can be easily solved, by considering all inside-outside annotations $\mathcal{S}_*$ of the given modal profile $\mathcal{S}_+$, computing $\mathsf{MinDownsets}_{\mathbb{G}_*}(\mathcal{S}_*)$ for each of them, considering the deannotations of the resulting downsets, and taking the minimal downsets among them.

There is a natural definition of the *SCC-reachability-annotated global representative* of an inside-outside short-long modal profile $\mathcal{S}_*$, denoted $\mathsf{GlobalRepresentative}_*(\mathcal{S}_*)$, with the property that $\mathsf{GlobalRepresentative}_*(\mathcal{S}_*) \to G_*$ for each graph $G_* \in \mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$; we do not define it formally, as we will use a slightly simpler construction, but a picture representing intuitions behind it is shown in Figure 7.9 on the facing page. We conjecture that, if $\mathcal{S}_*$ is satisfiable within $\mathbb{G}_*$, then $\mathsf{MinDownsets}_{\mathbb{G}_*}(\mathcal{S}_*) = \{\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_*(\mathcal{S}_*))\}$, where n is the depth of $\mathcal{S}_*$.

Proving the above conjecture would require constructing a graph $G_* \in \mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$ such that each subgraph of G_* with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}_*(\mathcal{S}_*)$. However, such a construction and the proof of its correctness seem to be quite complicated; for this reason, we describe a slightly different algorithm for solving the minimal downsets problem for $\mathbb{G}_+$, resulting in significantly simpler constructions and proofs. It is useful to still think of the algorithm as the reduction described above, but with a swapped order of two steps: computing the downset of an SCC-reachability-annotated global representative, and taking its deannotation.

Our approach

We want to compute $\mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ for a modal profile $\mathcal{S}_+$ of depth n. For each inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ that is satisfiable within $\mathbb{G}_*$, we can compute the following graphs and downsets:

- $G_* = \mathsf{GlobalRepresentative}_*(\mathcal{S}_*)$;
- $G_+ = \mathsf{Deannotate}(G_*)$;
- $\mathcal{D} = \mathsf{Downset}_n(G_+)$.

The result is the set of minimal downsets among $\mathcal{D}$ computed above.

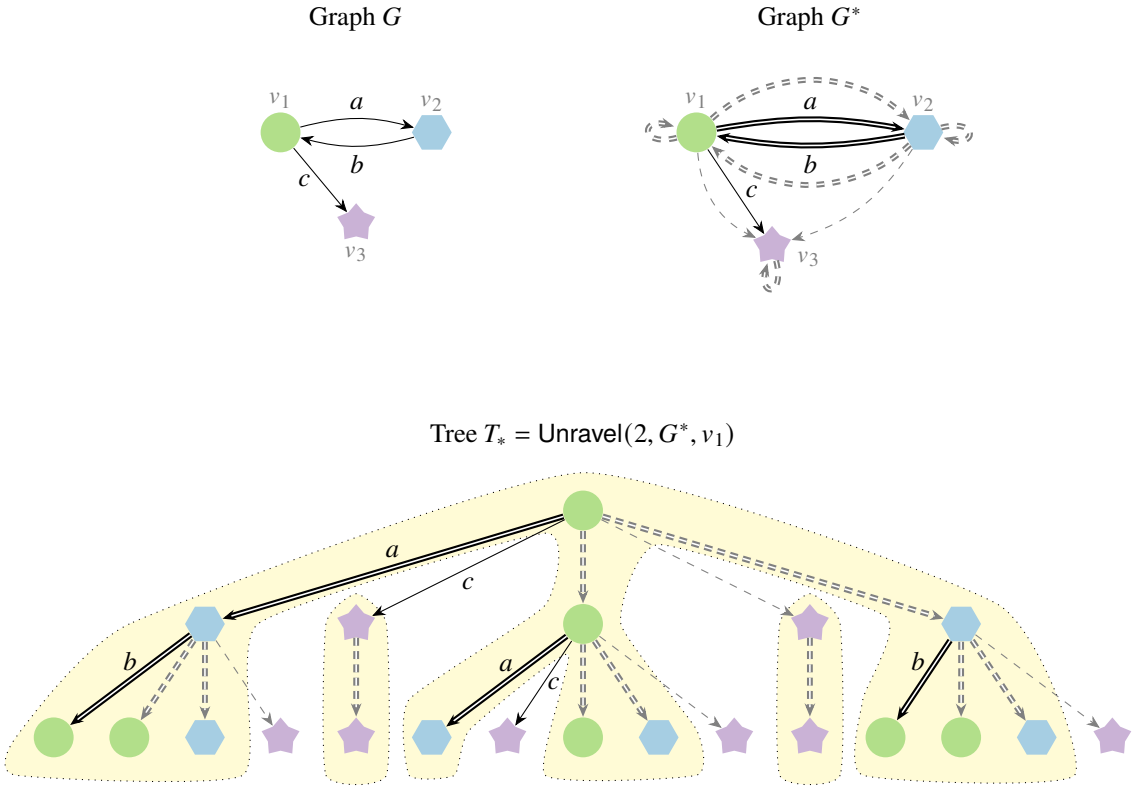


Figure 7.9: An example of graphs G, G^* and $T_* = \mathsf{Unravel}(2, G^*, v_1)$. Note that T_* maps homomorphically to G^*. In T_*, we highlighted three sets of nodes; inside each set, all nodes are connected by inside edges. This means that all these nodes originate from the same SCC in G; thus, we can add long inside edges between them, and retain a homomorphism to G^*. This is the main idea behind the construction of $\mathsf{GlobalRepresentative}_*(\mathcal{S}_*)$: consider $\mathsf{GlobalRepresentative}(\mathcal{S}_*)$, identify each maximal set of nodes connected only by inside edges, and add long inside edges between these nodes; to finish the construction, we would also need to add long *outside* edges appropriately.

As a technical simplification, instead of working with functions $\mathsf{GlobalRepresentative}_*$ and $\mathsf{Deannotate}$, we define directly a function that can be seen as their composition, $\mathsf{GlobalRepresentative}_{*\to+}$. Then, the algorithm is as follows. For a given modal profile $\mathcal{S}_+$, we define $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ as the set of $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*))$ for each inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ that is satisfiable within $\mathbb{G}_*$. The result is the set $\mathsf{MinComputedDownsets}_+(\mathcal{S}_+)$ of minimal downsets in $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$.

To prove the correctness of the algorithm, we will show that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

We note that the main simplification gained by our approach, compared to a reduction to the minimal downsets problem for $\mathbb{G}_*$, lies specifically in the statement of the second claim above; intuitively, the difference can be described as $\mathcal{D}'$ ranging over downsets of all graphs in $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$, instead of downsets of graphs in $\mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$ for some fixed inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ – up to (de)annotations, $\mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$ can be seen as a subset of $\mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$.

7.4.2 Additional definitions

Before we define the adjusted notions of unravellings and representatives, we provide a few simple definitions related to walks and trees.

Removing suffixes of walks

Consider a walk $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$ and its suffix $\sigma = [v_i, e_{i+1}, v_{i+1}, \ldots, e_m, v_m]$. The walk obtained from π by *removing* the suffix σ is the walk $[v_0, e_1, v_1, \ldots, e_i, v_i]$. Note that the node v_i is not removed; in particular, removing the trivial suffix from π results in π, and removing the suffix π from π results in a trivial walk $[v_0]$. Similarly, the walk obtained from π by removing a prefix $[v_0, e_1, v_1, \ldots, e_i, v_i]$ is the walk $[v_i, e_{i+1}, v_{i+1}, \ldots, e_m, v_m]$.

Forests, lowest common ancestors

A *forest* is a graph in which each (weakly) connected component is a tree.

Consider a tree T. A *descendant* of a node u in T is any node reachable from u. If v is a descendant of u, then u is an *ancestor*[1] of v. The *lowest common ancestor* of nodes u, v in T, denoted as $\mathsf{LCA}_T(u, v)$, is the common ancestor of both u and v of maximal depth; recall that the depth of a node in a tree is the length of the walk from the root to this node.

We use the following property of ancestors and homomorphisms. Consider a homomorphism h from a tree T_1 to a tree T_2. For two nodes u, v in T_1, if u is an ancestor of v in T_1, then $h(u)$ is an ancestor of $h(v)$ in T_2, because the walk from u to v in T_1 is mapped by h to a walk from $h(u)$ to $h(v)$. Consider two nodes u, v in T_1 and their lowest common ancestor w. Then, $h(w)$ is a common ancestor of $h(u)$ and $h(v)$ in T_2, but not necessarily the *lowest* common ancestor. Thus, the lowest common ancestor of $h(u)$ and $h(v)$ in T_2 occurs in the walks from $h(w)$ to $h(u)$ and from $h(w)$ to $h(v)$ in T_2.

[1] We note that we used the term *ancestor* differently in the introduction; here, every node is its own ancestor.

7.4.3 Unravellings

Consider a node v_0 in a graph G and a non-negative integer n. Below we define the *reachability-annotated n-step unravelling* of G^+ from v_0, denoted $\mathsf{Unravel}_+(n, G^+, v_0)$, by providing two equivalent definitions: one based on $\mathsf{Unravel}(n, G^+, v_0)$, and a direct one. Recall that $\mathsf{Unravel}(n, G, v_0)$ is the tree with nodes being all walks of length at most n in G that begin in v_0, with the label of a node π being the label of $\mathsf{LastNode}(\pi)$ in G, and an edge (π_1, a, π_2) iff $\pi_2 \in \mathsf{Extensions}_G(\pi_1, a)$. Both definitions of $\mathsf{Unravel}_+(n, G^+, v_0)$ use the notion of *pseudoextensions*; this is one of the most important concepts in this chapter.

Pseudoextensions

Consider two walks π_1, π_2 in a graph G. Let π_0 be the walk obtained from π_1 by removing the longest suffix contained in one SCC in G. Then, π_2 is a *pseudoextension* of π_1 if π_2 is an extension of π_0. Some examples and non-examples of pseudoextensions are shown in Figure 7.10.

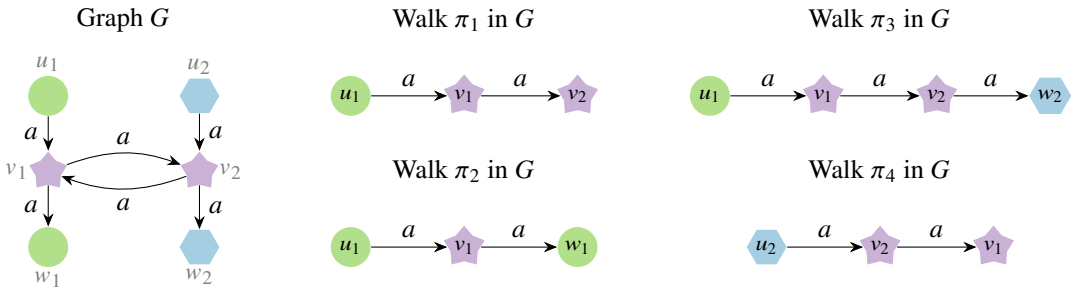


Figure 7.10: An example of walks in a graph. The walk π_2 is a pseudoextension of π_1, because the prefix of π_1 obtained by removing the longest suffix contained in one SCC in G is $\pi_0 = [u_1, (u_1, a, v_1), v_1]$; π_2 is an extension of π_0. Similarly, π_3 is a pseudoextension of π_1. There are no other pseudoextensions in this example, apart from every walk being a pseudoextension of itself.

Note that, in unravellings of graphs, the lowest common ancestor of two nodes π_1, π_2 is the longest common prefix of π_1 and π_2. For this reason, it is useful to provide an equivalent definition of a pseudoextension in terms of the longest common prefix. Consider two walks π_1, π_2 in a graph G, beginning in the same node, and let π_0 be the longest common prefix of π_1 and π_2. Then, π_2 is a pseudoextension of π_1 if the suffix of π_1 obtained by removing π_0 is contained in one SCC in G.

Fact 7.14. In every graph, if a walk π_2 is an extension of a walk π_1, then π_2 is a pseudoextension of π_1.

Fact 7.15 (Pseudoextension transitivity)**.** For walks π_1, π_2, π_3 in a graph, if π_2 is a pseudoextension of π_1 and π_3 is a pseudoextension of π_2, then π_3 is a pseudoextension of π_1.

Unravellings

We define $\mathsf{Unravel}_+(n, G^+, v_0)$ as the short-long graph obtained from $\mathsf{Unravel}(n, G^+, v_0)$ by adding a long edge from π_+ to π'_+ if π'_+ is a pseudoextension of π_+ in G^+. An example of graphs G and $\mathsf{Unravel}_+(1, G^+, u)$ is shown in Figure 7.11.

Equivalently, due to Fact 7.14, $\mathsf{Unravel}_+(n, G^+, v_0)$ is the short-long graph with:

- nodes being all walks of length at most n in G^+ that begin in v_0;
- a short edge $(\pi_+, (\mathsf{Short}, a), \pi'_+)$ iff $\pi'_+ \in \mathsf{Extensions}_{G^+}(\pi_+, (\mathsf{Short}, a))$;
- a long edge from π_+ to π'_+ iff π'_+ is a pseudoextension of π_+ in G^+.

The label of a node π_+ is the label of $\mathsf{LastNode}(\pi_+)$ in G.

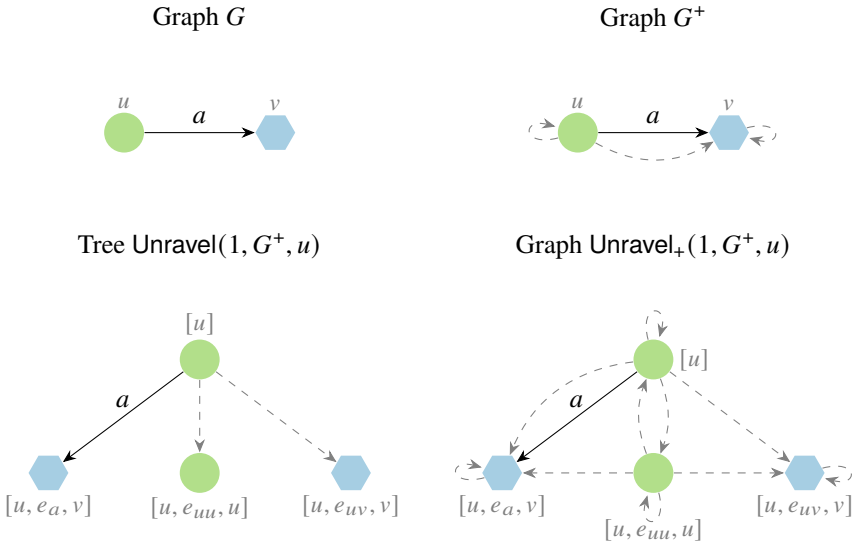


Figure 7.11: An example of graphs G, G^+, $\mathsf{Unravel}(1, G^+, u)$ and $\mathsf{Unravel}_+(1, G^+, u)$. We denote by e_a the edge $(u, (\mathsf{Short}, a), v)$, by e_{uu} the long edge $(u, (\mathsf{Long}, \bot), u)$, and by e_{uv} the long edge $(u, (\mathsf{Long}, \bot), v)$. Note, for example, the edge in $\mathsf{Unravel}_+(1, G^+, u)$ from $[u, e_{uu}, u]$ to $[u, e_{uv}, v]$: because e_{uu} is contained in an SCC in G^+, the latter is a pseudoextension of the former.

We define $\mathsf{GlobalUnravel}_+(n, G^+)$ as the union of $\mathsf{Unravel}_+(n, G^+, v)$ for each node v in G.

Claim 7.16. For every graph G and every non-negative integer n, $\mathsf{GlobalUnravel}_+(n, G^+) \to G^+$.

Proof. Let $H_+ = \mathsf{GlobalUnravel}_+(n, G^+)$, and let h map each node π_+ in H_+ to $\mathsf{LastNode}(\pi_+)$. We claim that h is a homomorphism from H_+ to G^+. Since h is a homomorphism from $\mathsf{GlobalUnravel}(n, G^+)$ to G^+, and H_+ is obtained from $\mathsf{GlobalUnravel}(n, G^+)$ by adding some long edges, it is enough to show that h preserves long edges. Consider a long edge from π_+ to π'_+ in H_+; then, π'_+ is a pseudoextension of π_+ in G^+. Let σ_+ be the walk obtained from π_+ by removing the longest suffix contained in one SCC in G^+; this means that $\mathsf{LastNode}(\sigma_+)$ and $\mathsf{LastNode}(\pi_+)$ are in one SCC in G^+. By SCC Invariance Under Annotations (Fact 7.13 on page 88), they are also in one SCC in G, so there is a walk π from $\mathsf{LastNode}(\pi_+)$ to $\mathsf{LastNode}(\sigma_+)$ in G. Additionally, by the definition of a pseudoextension, σ_+ is a prefix of π'_+, so there is a walk in G^+ from $\mathsf{LastNode}(\sigma_+)$ to $\mathsf{LastNode}(\pi'_+)$; each long edge in this walk corresponds to some walk in G, so there is also a walk π' in G from $\mathsf{LastNode}(\sigma_+)$ to $\mathsf{LastNode}(\pi'_+)$. Concatenating the walks π and π', we get a walk from $\mathsf{LastNode}(\pi_+)$ to $\mathsf{LastNode}(\pi'_+)$ in G, which proves that there is a long edge from $h(\pi_+)$ to $h(\pi'_+)$ in G^+. □

7.4.4 Representatives

One of the definitions of $\mathsf{Unravel}_+(n, G^+, v_0)$ is based on $\mathsf{Unravel}(n, G^+, v_0)$, adding long edges based on the containment of some walks in SCCs in G^+. The information about containment of edges in SCCs is also present in $\mathsf{Unravel}(n, G^*, v_0)$; below we define a function $\mathsf{RecoverReachability}$, intuitively, mapping $\mathsf{Unravel}(n, G^*, v_0)$ to $\mathsf{Unravel}_+(n, G^+, v_0)$, and depending only on the isomorphism type of the input tree.

Definitions

For an inside-outside short-long tree T_*, the short-long graph $\mathsf{RecoverReachability}(T_*)$ is obtained from the deannotation of T_* by adding a long edge from u to v if the walk in T_* from $\mathsf{LCA}_{T_*}(u, v)$ to u traverses only inside edges; recall that inside edges have labels of the form $(a_+, \mathsf{InsideSCC})$. An example of a tree T_* and the graph $\mathsf{RecoverReachability}(T_*)$ is shown in Figure 7.12.

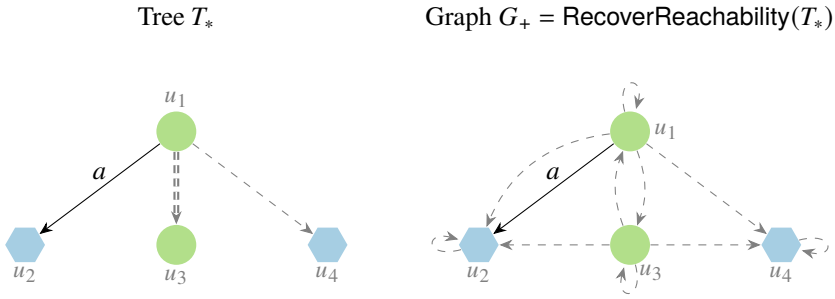


Figure 7.12: An example of an inside-outside short-long tree T_* and the graph $\mathsf{RecoverReachability}(T_*)$. The tree T_* contains one inside edge: the long edge from u_1 to u_3. Consider, for example, the long edge from u_3 to u_4 in G_+: $\mathsf{LCA}_{T_*}(u_3, u_4) = u_1$, and the walk from u_1 to u_3 in T_* traverses only inside edges, so there is a long edge from u_3 to u_4 in $\mathsf{RecoverReachability}(T_*)$. Note the similarities with Figure 7.11.

For an inside-outside short-long forest F_*, we define $\mathsf{RecoverReachability}(F_*)$ as the union of $\mathsf{RecoverReachability}(T_*)$ for each (weakly) connected component T_* in F_*.

The *reachability-annotated global representative* of an inside-outside short-long modal profile $\mathcal{S}_*$, denoted $\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$, is $\mathsf{RecoverReachability}(\mathsf{GlobalRepresentative}(\mathcal{S}_*))$.

Claim 7.17. For an inside-outside short-long modal profile $\mathcal{S}_* = (n, \mathcal{T}_*)$ and a graph G, if $G^* \models \mathcal{S}_*$, then $\mathsf{GlobalUnravel}_+(n, G^+) \leftrightarrow \mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$.

To prove the above claim, we need to first prove several auxiliary properties of reachability-annotated unravellings and representatives.

Properties

Claim 7.18. For a node v_0 in a graph G and a non-negative integer n, $\mathsf{Unravel}_+(n, G^+, v_0)$ is isomorphic to $\mathsf{RecoverReachability}(\mathsf{Unravel}(n, G^*, v_0))$.

Proof. The isomorphism maps a node π_* in $\mathsf{RecoverReachability}(\mathsf{Unravel}(n, G^*, v_0))$ to its *deannotation* π_+, obtained from π_* by replacing the label of each traversed edge by its deannotation. It is straightforward to check that the isomorphism preserves node labels and short edges, so let us focus on long edges. In $\mathsf{Unravel}_+$ they are defined using pseudoextensions, and in $\mathsf{RecoverReachability}$ they are defined using the lowest common ancestors; we need to argue that these definitions are equivalent. Let $T_* = \mathsf{Unravel}(n, G^*, v_0)$. Then, for two nodes π_*, π'_* in T_*, the node $\sigma_* = \mathsf{LCA}_{T_*}(\pi_*, \pi'_*)$ is the longest common prefix of π_* and π'_*. By the definition of $\mathsf{RecoverReachability}$, there is a long edge from π_* to π'_* in $\mathsf{RecoverReachability}(T_*)$ iff the walk from σ_* to π_* in T_* traverses only inside edges. Let π_+, π'_+, σ_+ be the deannotations of π_*, π'_*, σ_*, respectively; it is not difficult to see that σ_+ is the longest common prefix of π_+ and π'_+. By the SCC Invariance Under Annotations (Fact 7.13 on page 88), the walk from σ_* to π_* in T_* traverses only inside edges iff the suffix of π_+ obtained by removing the prefix σ_+ is contained in one SCC in G^+. This is equivalent to π'_+ being a pseudoextension of π_+ in G^+. Thus, there is a long edge from π_* to π'_* in $\mathsf{RecoverReachability}(T_*)$ iff there is a long edge from π_+ to π'_+ in $\mathsf{Unravel}_+(n, G^+, v_0)$. □

Corollary 7.19. For every graph G and every non-negative integer n, $\mathsf{GlobalUnravel}_+(n, G^+)$ is isomorphic to $\mathsf{RecoverReachability}(\mathsf{GlobalUnravel}(n, G^*))$.

Claim 7.20. For two inside-outside short-long forests G_*, H_*, if there is a weak homomorphism from G_* to H_*, then $\mathsf{RecoverReachability}(G_*) \to \mathsf{RecoverReachability}(H_*)$.

Proof. Let $G_+ = \mathsf{RecoverReachability}(G_*)$ and let $H_+ = \mathsf{RecoverReachability}(H_*)$. Let h be a weak homomorphism from G_* to H_*. We claim that h is a homomorphism from G_+ to H_+. The fact that h preserves node labels and short edges follows directly from the definition of $\mathsf{RecoverReachability}$. Consider a long edge from u to v in G_+; let $w = \mathsf{LCA}_{G_*}(u, v)$. The walk from w to u in G_* traverses only inside edges. This walk is mapped by h to a walk in H_* that also traverses only inside edges, by the definition of a weak homomorphism. The walk from $\mathsf{LCA}_{H_*}(h(u), h(v))$ to $h(u)$ in H_* is a suffix of this walk, so it also traverses only inside edges, so there is a long edge from $h(u)$ to $h(v)$ in H_+. □

Now we are ready to prove Claim 7.17; recall its statement.

Claim 7.17. For an inside-outside short-long modal profile $\mathcal{S}_* = (n, \mathcal{T}_*)$ and a graph G, if $G^* \models \mathcal{S}_*$, then $\mathsf{GlobalUnravel}_+(n, G^+) \leftrightarrow \mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$.

Proof. The claim follows from the fact that $\mathsf{GlobalUnravel}(n, G^*) \leftrightarrow \mathsf{GlobalRepresentative}(\mathcal{S}_*)$ (Fact 6.4 on page 66), and Corollary 7.19 and Claim 7.20 above. □

7.4.5 Correctness of the reduction: the first steps

Recall the reduction. For a given modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$, we define $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ as the set of $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*))$ for each inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ that is satisfiable within $\mathbb{G}_*$. The result is the set $\mathsf{MinComputedDownsets}_+(\mathcal{S}_+)$ of minimal downsets in $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$.

We claim that $\mathsf{MinComputedDownsets}_+(\mathcal{S}_+) = \mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$. To prove it, we show that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

Because both $\mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ and $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ are finite, this implies that $\mathsf{MinComputedDownsets}_+(\mathcal{S}_+) = \mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$.

The first claim follows from the properties of reachability-annotated global representatives and unravellings proved above, as follows. Since $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$, there is a graph G such that $G^+ \models \mathcal{S}_+$ and $\mathsf{Downset}_n(G^+) = \mathcal{D}$. Let $\mathcal{S}_*$ be the inside-outside annotation of $\mathcal{S}_+$ such that $G^* \models \mathcal{S}_*$; it exists by Fact 7.10 on page 87. By Claims 7.16 and 7.17, $\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*) \to G^+$; so $\mathcal{D}' \subseteq \mathcal{D}$ for $\mathcal{D}' = \mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)) \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$.

The crux of the proof of the correctness of the reduction is the second claim, stated below with expanded definitions, and proved in the next section.

Proposition 7.21. For every modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$, for each inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ that is satisfiable within $\mathbb{G}_*$ there exists a graph $G_+ \in \mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ such that each subgraph of G_+ with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$.

Assuming that the satisfiability problem for $\mathbb{G}_*$ is decidable, the computability of the minimal downsets problem for $\mathbb{G}_+$ follows, as constructing $\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$ for a given inside-outside modal profile $\mathcal{S}_*$ is easily computable, and the set of all inside-outside annotations of $\mathcal{S}_+$ is computable (Fact 7.9 on page 87).

7.5 Correctness of the reduction: the crux

In this section we prove Proposition 7.21. As in the previous chapter, we will use two ravelling constructions, and using the homomorphism equivalence of (reachability-annotated) global representatives and global unravellings (Claim 7.17 on page 95), we will show homomorphisms to global unravellings instead. Because the proof is much more technically involved than in the previous chapter, we first prove two claims about homomorphisms between unravellings that significantly simplify the proof. Then, we introduce the *Last Edge Appearance Record*, construct a required graph, and prove the existence of appropriate homomorphisms.

7.5.1 Homomorphisms between unravellings

Claim 7.22. For graphs G, H, if $G \to H$, then $\mathsf{GlobalUnravel}_+(n, G^+) \to \mathsf{GlobalUnravel}_+(n, H^+)$ for every non-negative integer n.

Proof. By Fact 7.12 on page 88, there is a weak homomorphism from G^* to H^*. This weak homomorphism can be lifted to a weak homomorphism from $\mathsf{GlobalUnravel}(n, G^*)$ to $\mathsf{GlobalUnravel}(n, H^*)$. By Corollary 7.19 and Claim 7.20 on the facing page, $\mathsf{GlobalUnravel}_+(n, G^+) \to \mathsf{GlobalUnravel}_+(n, H^+)$. □

To prove the next claim, we will use some well-known facts about the lowest common ancestors in a tree. Consider a tree T and a non-empty subset X of its nodes. Let $Y = \{\mathsf{LCA}_T(u, v) : u, v \in X\}$ be the set of the lowest common ancestors of all pairs of nodes in X. Note that $X \subseteq Y$, because $\mathsf{LCA}_T(u, u) = u$ for every node u. It is a well-known fact that the set Y has size at most $2|X| - 1$, and that Y is closed under taking the lowest common ancestors: for all $u, v \in Y$, $\mathsf{LCA}_T(u, v) \in Y$. Moreover, for every node $u \in Y$, there are at most $|X|$ nodes in Y that are ancestors of u in T, including u itself.

Claim 7.23. For a graph G, positive integers n, N, and a short-long graph H_+ with at most n edges, if $H_+ \to \mathsf{GlobalUnravel}_+(N, G^+)$, then $H_+ \to \mathsf{GlobalUnravel}_+(n, G^+)$.

Proof. Assume $n < N$, as otherwise $\mathsf{GlobalUnravel}_+(N, G^+) \to \mathsf{GlobalUnravel}_+(n, G^+)$, which makes the claim trivial. By considering each (weakly) connected component of H_+ separately, we can assume that H_+ is connected, has at most $n + 1$ nodes, and that $H_+ \to \mathsf{Unravel}_+(N, G^+, v_0)$ for some node v_0: since H_+ has at most n edges, each of its connected components has at most $n + 1$ nodes, and if a connected graph maps homomorphically to $\mathsf{GlobalUnravel}_+(N, G^+)$, then it maps homomorphically to $\mathsf{Unravel}_+(N, G^+, v_0)$ for some node v_0. We will show that $H_+ \to \mathsf{Unravel}_+(n, G^+, v_0')$ for some node v_0'. We use the fact that $\mathsf{Unravel}_+(k, G^+, v)$ is isomorphic to $\mathsf{RecoverReachability}(\mathsf{Unravel}(k, G^*, v))$ for every node v in G and every non-negative integer k (Claim 7.18 on page 95).

Let $T_* = \mathsf{Unravel}(N, G^*, v_0)$, and let h be a homomorphism from H_+ to $\mathsf{RecoverReachability}(T_*)$. We construct a homomorphism h' from H_+ to $\mathsf{RecoverReachability}(T_*')$ for an appropriate n-step unravelling T_*' of G^*; we define T_*' by considering only nodes in the image of h and their lowest common ancestors in T_*, and, intuitively, by contracting walks between these nodes to single long edges.

Note that in T_* a node π_* is an ancestor of π_*' iff π_* is a prefix of π_*'. Moreover, π_* is a parent of π_*' iff π_* is the longest proper prefix of π_*'.

Let V be the set of all nodes in the image of h and all their lowest common ancestors in T_*. The *V-root* is the shortest walk in V. The *V-parent* of a walk π_* in V is the longest proper prefix of π_* in V; the V-root has no V-parent. A *V-ancestor* of a walk π_* in V is any prefix of π_* in V. By the facts stated above the claim, the number of V-ancestors of a walk π_* in V is at most $n + 1$.

Let v_0' be the last node in the V-root, and let $T_*' = \mathsf{Unravel}(n, G^*, v_0')$. Let D_+ be the subgraph of $\mathsf{RecoverReachability}(T_*)$ induced by V. Since the image of h is a subset of V, $H_+ \to D_+$. We define a homomorphism h' from D_+ to $\mathsf{RecoverReachability}(T_*')$ recursively, as follows. For the V-root π_*, $h'(\pi_*)$ is the trivial walk $[v_0']$. For every other walk π_* in V, h' is defined recursively based on the V-parent π_*' of π_*, with the following two cases.

- If π_* is an extension of π_*' by one short edge e_*, then $h'(\pi_*) = h'(\pi_*') \cdot e_*$.
- Otherwise, $h'(\pi_*)$ is the extension of $h'(\pi_*')$ by the long edge from $\mathsf{LastNode}(\pi_*')$ to $\mathsf{LastNode}(\pi_*)$ in G^*. This edge has the label $((\mathsf{Long}, \bot), \mathsf{InsideSCC})$ if $\mathsf{LastNode}(\pi_*)$ and $\mathsf{LastNode}(\pi_*')$ are in one SCC in G, or $((\mathsf{Long}, \bot), \mathsf{OutsideSCC})$ otherwise. Note that the first case occurs iff the walk from π_*' to π_* in T_* traverses only inside edges.

It remains to prove that h' is indeed a homomorphism from D_+ to $\mathsf{RecoverReachability}(T_*')$. For each node π_* in D_+, by the recursive definition of h', $h'(\pi_*)$ is a walk in G^* beginning in v_0', and it has length

at most n, because π_* has at most $n+1$ V-ancestors. Moreover, $\mathsf{LastNode}(\pi_*) = \mathsf{LastNode}(h'(\pi_*))$, directly from the definition of h', so h' preserves node labels.

For each short edge $(\pi'_*, (\mathsf{Short}, a), \pi_*)$ in D_+, by the definition of $\mathsf{RecoverReachability}$, there is a short edge $(\pi'_*, ((\mathsf{Short}, a), z), \pi_*)$ in T_* for some $z \in \{\mathsf{InsideSCC}, \mathsf{OutsideSCC}\}$, so π_* is an extension of π'_* by one short edge e_* with the label $((\mathsf{Short}, a), z)$. Since both π_* and π'_* are in D_+, they are also both in V, so π'_* is the V-parent of π_*, because π'_* is the longest proper prefix of π_*. Thus, by the definition of h', $h'(\pi_*)$ is the extension of $h'(\pi'_*)$ by e_*, so there is an edge in T'_* from $h'(\pi'_*)$ to $h'(\pi_*)$ with the label $((\mathsf{Short}, a), z)$, so in $\mathsf{RecoverReachability}(T'_*)$ there is a short edge from $h'(\pi'_*)$ to $h'(\pi_*)$ with the label (Short, a).

Finally, consider a long edge from π_* to π'_* in D_+; the walk in T_* from $\sigma_* = \mathsf{LCA}_{T_*}(\pi_*, \pi'_*)$ to π_* traverses only inside edges. By the definition of h', the walk from $h'(\sigma_*)$ to $h'(\pi_*)$ in T'_* also traverses only inside edges. The same holds for every suffix of this walk; in particular, for the walk from $\mathsf{LCA}_{T'_*}(h'(\pi_*), h'(\pi'_*))$ to $h'(\pi_*)$. Thus, there is a long edge from $h'(\pi_*)$ to $h'(\pi'_*)$ in $\mathsf{RecoverReachability}(T'_*)$. □

7.5.2 Last Edge Appearance Record

Below we define the Last Edge Appearance Record of a walk in a graph; it will be used as a ravelling function in the proof of Proposition 7.21.

Consider a walk π in a graph G. Let π_+ be the corresponding walk in G^+, obtained from π by replacing each edge (u, a, v) with $(u, (\mathsf{Short}, a), v)$. Let $\pi_+ = [v_0, e_1, v_1, \ldots, e_m, v_m]$. An index $i \in \{1, \ldots, m\}$ is the *last appearance* of the edge e_i if $e_j \neq e_i$ for all $j > i$. An infix of π_+ is *avoiding last appearances* if it does not contain the last appearance of any edge. The *Last Edge Appearance Record* of π, denoted $\mathsf{LEAR}(\pi)$, is the walk in G^+ obtained from π_+ by replacing each maximal non-trivial infix that avoids last appearances $[v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ with a single long edge; formally, with $[v_i, (v_i, (\mathsf{Long}, \bot), v_j), v_j]$. An example is shown in Figure 7.13.

Note that in $\mathsf{LEAR}(\pi)$ there are no two consecutive long edges, and each short edge appears at most once, so, for a fixed graph G, the image of the function LEAR is finite. Note also that an edge not contained in any SCC in G is traversed at most once by π, so it does not belong to any infix that avoids last appearances, which means that it is not replaced by a long edge in the construction.

7.5.3 Constructing a witness

We want to prove Proposition 7.21: for every modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$, for each inside-outside annotation $\mathcal{S}_*$ of $\mathcal{S}_+$ that is satisfiable within $\mathbb{G}_*$ there exists a graph $G_+ \in \mathsf{Models}_{\mathbb{G}_+}(\mathcal{S}_+)$ such that each subgraph of G_+ with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$.

Consider a modal profile $\mathcal{S}_+ = (n, \mathcal{T}_+)$ and its inside-outside annotation $\mathcal{S}_*$ that is satisfiable within $\mathbb{G}_*$. Let G_0 be some graph such that $(G_0)^* \models \mathcal{S}_*$.

Let $G_1 = \mathsf{SafeRavel}_f(G_0)$ for f mapping a walk to its length modulo $n+1$, as in the previous chapter.

Let $G_2 = \mathsf{Ravel}_{\mathsf{LEAR}}(G_1)$.

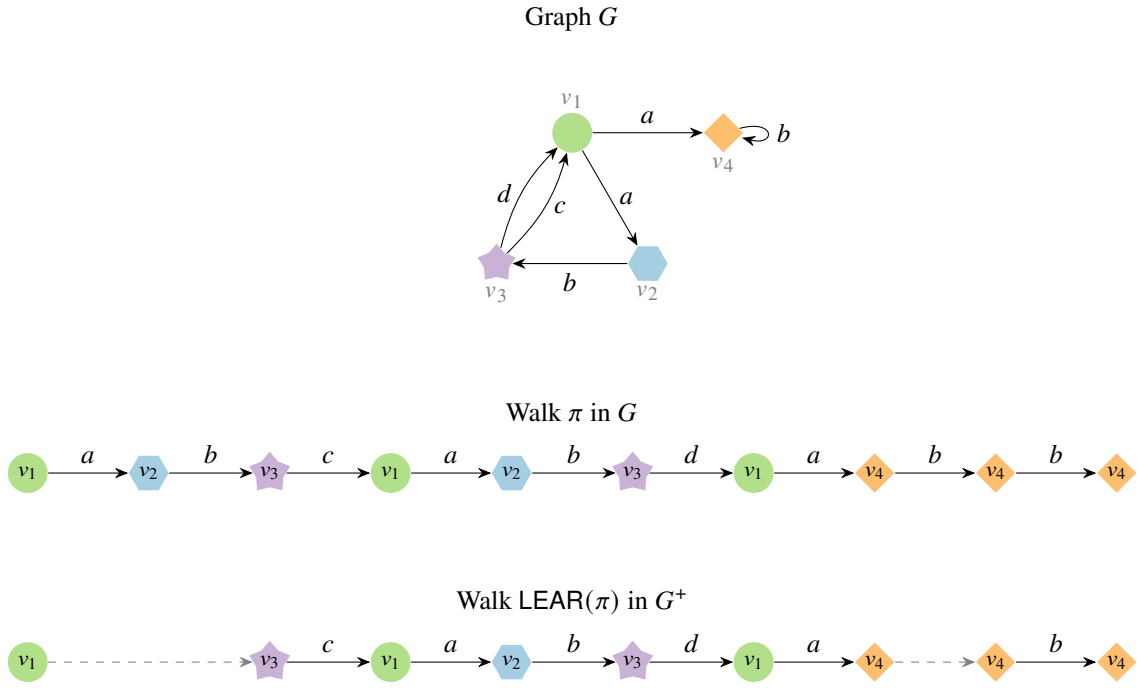


Figure 7.13: An example of a graph G, a walk π in G, and the Last Edge Appearance Record of π.

By Fact 7.7 on page 85, $(G_1)^+ \models \mathcal{S}_+$ and $(G_2)^+ \models \mathcal{S}_+$.

To prove Proposition 7.21 and finish the proof of the correctness of the reduction, we show that, for every subgraph D_+ of $(G_2)^+$ with at most n edges, $D_+ \to \mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$. More precisely, we prove that $D_+ \to \mathsf{GlobalUnravel}_+(n, (G_1)^+)$; the rest follows from the fact that $(G_0)^* \models \mathcal{S}_*$, and from earlier claims:

- by Claim 7.22 on page 97, $\mathsf{GlobalUnravel}_+(n, (G_1)^+) \to \mathsf{GlobalUnravel}_+(n, (G_0)^+)$;
- by Claim 7.17 on page 95, $\mathsf{GlobalUnravel}_+(n, (G_0)^+) \leftrightarrow \mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*)$.

Thus, it remains to prove the following proposition.

Proposition 7.24. Each subgraph D_+ of $(G_2)^+$ with at most n edges maps homomorphically to $\mathsf{GlobalUnravel}_+(n, (G_1)^+)$.

We prove this proposition in two steps, as in the previous chapter. However, to simplify the proofs, this time we define a homomorphism h from D_+ to $\mathsf{GlobalUnravel}_+(N, (G_1)^+)$ for some positive integer N; by Claim 7.23 on page 98, it follows that D_+ maps homomorphically to $\mathsf{GlobalUnravel}_+(n, (G_1)^+)$. We first construct the homomorphism h, and then we prove that it is indeed a homomorphism.

Constructing the homomorphism $h : D_+ \to \mathsf{GlobalUnravel}_+(N, (G_1)^+)$

Let $h_{G_2\to G_1}$ be the ravelling homomorphism from G_2 to G_1; by Fact 7.6 on page 85, it is also a homomorphism from $(G_2)^+$ to $(G_1)^+$. A short edge in $(G_1)^+$ is *important* if it is in $h_{G_2\to G_1}(D_+)$, using the graph-image notation; that is, if some edge in D_+ is mapped to it by $h_{G_2\to G_1}$. Note that long edges are not important. For an important (short) edge in $(G_1)^+$, we refer to the corresponding edge in

G_1 as *important* as well.

Each node in G_2 is a walk in $(G_1)^+$. Consider a walk π_+ in $(G_1)^+$. We want h to map π_+ to a walk obtained from π_+ by replacing some infixes with long edges, similarly to the LEAR construction; this time, intuitively, from each SCC visited by π_+ we want to keep only the longest suffix of important edges. For technical reasons, it is convenient to keep also the last unimportant edge in each SCC, if it exists. Refer to Figure 7.14 on the next page for an example (with a slight inaccuracy of too many important edges in G_1, to keep the example small). Below we formalize the definition of h in a way that might seem overly complicated, but it avoids considering edge cases, and will be useful in the next chapter as well, where the construction is much more difficult to describe intuitively.

Let $(e_1, \ldots, e_m)$ be the sequence of edges traversed by π_+. An index i is *important* if e_i is important. An index i is *dominated* by an index j if $j > i$ and e_i, e_j are in one SCC in $(G_1)^+$. Note that if an edge e_i is not in any SCC in $(G_1)^+$, then the index i is not dominated. An index i is *unimportantly dominated* if it is dominated by some index that is not important. An infix of π_+ is *unimportantly dominated* if it contains only unimportantly dominated indices. Let h map π_+ to the walk in $(G_1)^+$ obtained from π_+ by replacing each maximal non-trivial infix that is unimportantly dominated $[v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ with a single long edge; formally, with $[v_i, (v_i, (\mathsf{Long}, \bot), v_j), v_j]$. If the resulting walk contains some consecutive long edges, we additionally replace each maximal non-trivial infix traversing only long edges with a single long edge.

We want to be able to apply h to a walk π in G_1. There is a natural corresponding walk π_+ in $(G_1)^+$, obtained from π by replacing each edge (u, a, v) with $(u, (\mathsf{Short}, a), v)$. We define $h(\pi)$ as $h(\pi_+)$.

Proving that h is a homomorphism

We claim that h restricted to V_{D_+} is a homomorphism from D_+ to $\mathsf{GlobalUnravel}_+(N, (G_1)^+)$ for some positive integer N. Clearly, for each node π_+ in D_+, $h(\pi_+)$ is a walk in $(G_1)^+$, and we can take a large enough N so that all these walks are nodes in $\mathsf{GlobalUnravel}_+(N, (G_1)^+)$. The proof relies on the following two claims.

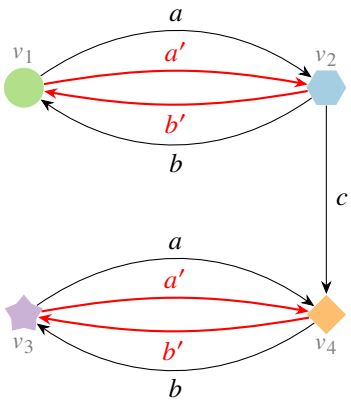
Claim 7.25. For each walk π in G_1, $h(\pi) = h(\mathsf{LEAR}(\pi))$.

Proof. Let π_+ be the walk in $(G_1)^+$ corresponding to π. The walk $h(\pi)$ is obtained from π_+ by replacing each maximal non-trivial unimportantly dominated infix with a long edge. The walk $\mathsf{LEAR}(\pi)$ is obtained from π_+ by replacing each maximal non-trivial infix avoiding last appearances with a long edge. The walk $h(\mathsf{LEAR}(\pi))$ is obtained from $\mathsf{LEAR}(\pi)$ by replacing each maximal non-trivial unimportantly dominated infix with a long edge.

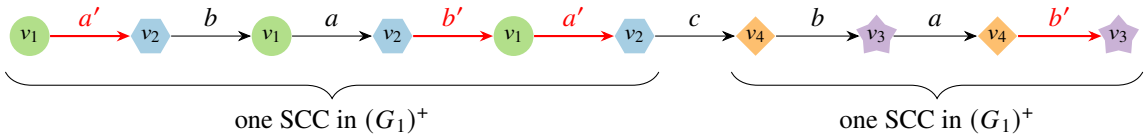
Let $(e_1, \ldots, e_m)$ be the sequence of edges traversed by π_+. Both walks $h(\pi)$ and $h(\mathsf{LEAR}(\pi))$ are obtained from π_+ by replacing some non-trivial infixes by long edges, and they do not contain consecutive long edges, so they can be uniquely characterized by the set of indices of edges e_i that are not replaced by long edges in the constructions h and LEAR. We need to show that in both cases this set of indices is the same. For this, we need to prove that:

- each non-last appearance of an edge in π_+ is unimportantly dominated in π_+;

Graph G_1, with important edges colored red



Walk π_+ in $(G_1)^+$ with only short edges



one SCC in $(G_1)^+$ one SCC in $(G_1)^+$

Walk $h(\pi_+)$

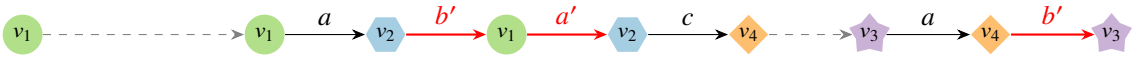


Figure 7.14: An example of h mapping a walk π_+ in $(G_1)^+$ to a walk obtained from π_+ by replacing some infixes with long edges, keeping from each visited SCC only the longest suffix of important edges, and the last unimportant edge. To keep the example small, we increased the number of important edges in G_1; crucially, in the overall proof, there are at most n important edges in G_1, and G_1 has no cycles of length at most n (Claim 6.6 on page 68).

- each last appearance of an edge that is unimportantly dominated in π_+ is also unimportantly dominated in $\mathsf{LEAR}(\pi)$.

The second point is easy to prove: for an index i that is dominated in π_+ by some unimportant index j, the index i is also dominated by the last appearance of e_j in π_+, which is also unimportant. For the first point, consider a non-last appearance e_i of some edge in π_+. Let e_j be the last appearance of e_i in π_+. Consider the infix $\sigma_+ = [v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ of π_+; it is a closed walk in $(G_1)^+$ that consists only of short edges. Each cycle in G_1 has length at least $n+1$ (Claim 6.6 on page 68), and there are at most n important edges in $(G_1)^+$, so σ_+ contains some unimportant edge. Moreover, since σ_+ is a closed walk, it is contained in one SCC in $(G_1)^+$. Thus, i is dominated by some unimportant edge. □

Thanks to the claim above, we will be able to focus on applying the function h to walks in G_1 instead of walks in $(G_1)^+$; or, in other words, we can focus on walks in $(G_1)^+$ that traverse only short edges. The claim below contains the core argument that h is a homomorphism, for walks in G_1; as we will see, the rest of the proof boils down to unravelling definitions and using the two claims.

Claim 7.26. For two walks π, π' in G_1 such that $\pi' = \pi \cdot e$ for some edge e, and their corresponding walks π_+, π'_+ and edge e_+ in $(G_1)^+$:

- if e_+ is important, then $h(\pi'_+) = h(\pi_+) \cdot e_+$;
- otherwise, $h(\pi'_+)$ is a pseudoextension of $h(\pi_+)$ in $(G_1)^+$.

Proof. Let $E_1 = (e_1, \ldots, e_m)$ be the sequence of edges traversed by π_+; then, the sequence of edges traversed by π'_+ is $E_2 = (e_1, \ldots, e_m, e_{m+1})$ for $e_{m+1} = e_+$. The last index $m+1$ is not dominated by any index in E_2.

If e_+ is important, then $h(\pi'_+) = h(\pi_+) \cdot e_+$, because each index $1 \leq i \leq m$ is unimportantly dominated in E_1 iff i is unimportantly dominated in E_2.

If e_+ is not important, then the last index $m+1$ in E_2 dominates all indices in the longest suffix of π'_+ contained in one SCC in $(G_1)^+$. If this suffix is equal to the whole walk π'_+, then clearly $h(\pi'_+)$ is a pseudoextension of $h(\pi_+)$, as both are contained in one SCC in $(G_1)^+$. Otherwise, let e_i be the last edge in E_2 that is not in the same SCC as e_+ in $(G_1)^+$; the edge e_i does not belong to any SCC in $(G_1)^+$, so the index i is not dominated in E_2, nor in E_1. Thus, $h(\pi'_+)$ is obtained from $h(\pi_+)$ by removing the longest suffix contained in one SCC in $(G_1)^+$ and appending some suffix, which is the definition of being a pseudoextension in $(G_1)^+$. □

To prove that h is a homomorphism, we need to show that h preserves node labels, short edges and long edges. Recall that $\mathsf{GlobalUnravel}_+(N, (G_1)^+)$ is the short-long graph with nodes being all walks of length at most N in $(G_1)^+$, each having the label of its last node in $(G_1)^+$, a short edge $(\pi_+, (\mathsf{Short}, a), \pi'_+)$ iff $\pi'_+ \in \mathsf{Extensions}_{(G_1)^+}(\pi_+, (\mathsf{Short}, a))$, and a long edge from π_+ to π'_+ iff π'_+ is a pseudoextension of π_+ in $(G_1)^+$. Recall that $G_2 = \mathsf{Ravel}_{\mathsf{LEAR}}(G_1)$.

It is easy to see that h preserves node labels, because ravelling constructions and $\mathsf{GlobalUnravel}_+$ inherit node labels from the last nodes of walks, and the functions LEAR and h preserve the last nodes of walks.

To show that h preserves both short and long edges, we will prove the following. For each short edge $e_+ = (\pi_+, (\mathsf{Short}, a), \pi'_+)$ in $(G_2)^+$ and for the edge e'_+ in $(G_1)^+$ such that $h_{G_2 \to G_1}$ maps e_+ to e'_+:

- if e_+ is in D_+, then $h(\pi'_+)$ is the extension of $h(\pi_+)$ by e'_+;
- otherwise, $h(\pi'_+)$ is a pseudoextension of $h(\pi_+)$ in $(G_1)^+$.

Note that the two points above are very similar to the formulation of Claim 7.26, but the claim concerns walks traversing only short edges; we will deal with this by considering ravelling witnesses of edges in G_2 and applying Claim 7.25, as we did in the previous chapter. The first point above is exactly what we need for h to preserve short edges. For h to preserve long edges we need to show that if a node π'_+ is reachable from a node π_+ in G_2, then $h(\pi'_+)$ is a pseudoextension of $h(\pi_+)$; this is implied by the two points above, since being an extension implies being a pseudoextension, and because being a pseudoextension is transitive.

It remains to prove the two points above. Let (π, e', π') be a ravelling witness of e_+; recall that $\pi_+ = \mathsf{LEAR}(\pi), \pi'_+ = \mathsf{LEAR}(\pi')$, and π' is the extension of π by e'. Note that e'_+ is the edge in $(G_1)^+$ corresponding to e' in G_1. By Claim 7.25, $h(\pi) = h(\pi_+)$ and $h(\pi') = h(\pi'_+)$. We need to prove that:

- if e_+ is in D_+, then $h(\pi')$ is the extension of $h(\pi)$ by e'_+;
- if e_+ is not in D_+, then $h(\pi')$ is a pseudoextension of $h(\pi)$ in $(G_1)^+$.

This follows from Claim 7.26, since, in the first case, if e_+ is in D_+, then e'_+ is important.

This finishes the proofs of Proposition 7.24 and Proposition 7.21, and of the correctness of the reduction from the minimal downsets problem for $\mathbb{G}_+$ to the satisfiability problem for $\mathbb{G}_*$.

7.5.4 Summary

We reduced the problem of computing minimal downsets for $\mathbb{G}_+$ to the satisfiability problem for $\mathbb{G}_*$: we defined $\mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ as the set of $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{*\to+}(\mathcal{S}_*))$ for all inside-outside annotations $\mathcal{S}_*$ of $\mathcal{S}_+$ that are satisfiable within $\mathbb{G}_*$. As in the previous chapter, to show that $\mathsf{MinComputedDownsets}_+(\mathcal{S}_+) = \mathsf{MinDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$, we proved that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}_+(\mathcal{S}_+)$ there is $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}_+}(\mathcal{S}_+)$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

The first claim followed easily from properties of reachability-annotated global representatives and unravellings. Proving the second claim was much more challenging: we used ravelling constructions with the Last Edge Appearance Record to construct a required graph; to prove the existence of appropriate homomorphisms we used an important technical property of homomorphisms between reachability-annotated unravellings (Claim 7.23 on page 98) that allowed us to focus on homomorphisms to $\mathsf{GlobalUnravel}_+(N, G^+)$ for an arbitrary positive integer N, rather than for N equal to the depth of $\mathcal{S}_+$.

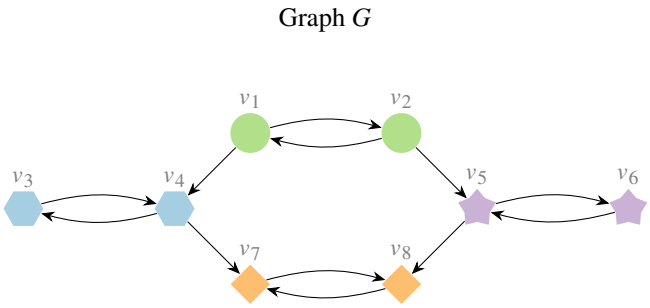
7.6 Satisfiability

In this chapter we solve two variants of the satisfiability problem; we begin with a simplified version of the satisfiability problem for $\mathbb{G}_\times$, to present the main idea of the construction used later to solve the satisfiability problem for $\mathbb{G}_*$. For this, we need to first define *node-level topological orders*.

7.6.1 Node-level topological orders

A *topological order* on SCCs in a graph G is a linear order $\leq$ such that, for each edge (u, a, v) in G, $\mathsf{SCC}_G(u) \leq \mathsf{SCC}_G(v)$. For each graph there exists a topological order on its SCCs.

A *node-level topological order* is a linear order $\leq$ on nodes in G such that, for each edge (u, a, v) in G, either $u \leq v$, or all nodes w such that $v \leq w \leq u$ are in one SCC in G. For each graph there exists a node-level topological order $\leq$, as one can be obtained from a topological order on SCCs by fixing an arbitrary linear order on nodes in each SCC and combining these orders. An example is shown in Figure 7.15.



Graph G, with nodes drawn according to an example node-level topological order

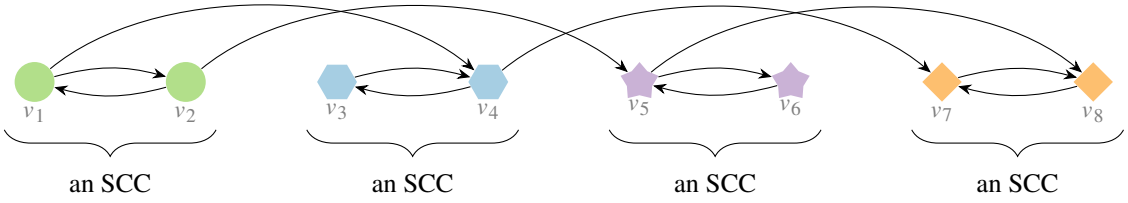


Figure 7.15: An example of a graph G and its node-level topological order. We omit edge labels, as they are not relevant for the example.

Note that a node-level topological order $\leq$ induces a topological order on SCCs, with $S_1 \leq S_2$ iff $v_1 \leq v_2$ for all nodes v_1 in S_1 and all nodes v_2 in S_2.

Descending edge sequences

To work with node-level topological orders, we need the following notion of *descending edge sequences*.

Consider a graph G and a linear order $\leq$ on the nodes in G (not necessarily a node-level topological order). Picture the nodes of G drawn in a single row, in the increasing order, with edges drawn as arcs above them, as in Figure 7.15. A *decreasing edge* is an edge (u, a, v) such that $v < u$; in the picture, a decreasing edge is an arc directed to the left.

Consider nodes $u < v$ and a walk π from v to u. Let $(e_1, \ldots, e_k)$ be the sequence of decreasing edges traversed by π, with each $e_i = (u_i, a_i, v_i)$. Because all other edges traversed by π are non-decreasing,

the following conditions hold:

- $v \leq u_1$;
- $v_i \leq u_{i+1}$ for each $i = 1, \ldots, k-1$;
- $v_k \leq u$.

A *descending edge sequence from v to u* is a sequence of decreasing edges that satisfies the three conditions above.

Fact 7.27. For a graph G and a linear order $\leq$ on its nodes, for nodes $u < v$ in one SCC in G, there exists a descending edge sequence from v to u.

Claim 7.28. For a node-level topological order $\leq$ in a graph G, two nodes $u < v$ are in one SCC in G iff there exists a descending edge sequence from v to u.

Proof. The left-to-right implication follows from the definition of a descending edge sequence, as one can be obtained from a walk from v to u in G. For the right-to-left implication, for each decreasing edge (u_i, a_i, v_i) in the descending edge sequence, by the properties of $\leq$ we have that all nodes w such that $v_i \leq w \leq u_i$ are in one SCC in G. By the inequalities in the definition of a descending edge sequence, and because $u < v$, all nodes w such that $u \leq w \leq v$ are in one SCC in G. □

7.6.2 Bounded satisfiability within SCC-annotations

The main purpose of introducing the simplified satisfiability problem below is to gradually build the solution to the satisfiability problem for $\mathbb{G}_*$. However, the problem itself is not completely arbitrary: it corresponds very closely to a subproblem we solve in (Gutiérrez-Basulto et al. 2022), where the logic $\mathcal{ALC}$ is considered instead of $\mathcal{ALCO}_{\text{reg}}$. We also reference the construction used to solve the simplified problem when discussing counting (the description logic $\mathcal{ALCQ}$) in Chapter 12.

We define a preorder $\preccurlyeq$ on inside-outside modal profiles as follows: $\mathcal{S}_\times \preccurlyeq \mathcal{S}'_\times$ if the deannotations of $\mathcal{S}_\times$ and $\mathcal{S}'_\times$ are equal, and there is a weak homomorphism from $\mathsf{GlobalRepresentative}(\mathcal{S}_\times)$ to $\mathsf{GlobalRepresentative}(\mathcal{S}'_\times)$.

$\preccurlyeq$-BOUNDED SATISFIABILITY OF A MODAL PROFILE WITHIN $\mathbb{G}_\times$

Input: A modal profile $\mathcal{S}^0_\times$.

Question: Is $\mathsf{Models}_{\mathbb{G}_\times}(\mathcal{S}_\times)$ non-empty for some $\mathcal{S}_\times \preccurlyeq \mathcal{S}^0_\times$?

Consider a modal profile $\mathcal{S}^0_\times = (n, \mathcal{T}^0_\times)$ such that $\mathsf{Models}_{\mathbb{G}_\times}(\mathcal{S}_\times)$ is non-empty for some $\mathcal{S}_\times \preccurlyeq \mathcal{S}^0_\times$, and let G be a graph such that $G^\times \models \mathcal{S}_\times$. Let $\mathcal{S}_\times = (n, \mathcal{T}_\times)$.

We will construct a graph H of bounded size such that $\mathcal{S}'_\times \preccurlyeq \mathcal{S}_\times$, where $\mathcal{S}'_\times$ is the modal profile of $H^\times$ of depth n. First, we will show why the strategy of identifying nodes with equal types does not work in this case, and we will present another relatively simple construction, based on node-level topological orders.

Consider the graph G in Figure 7.16 and its SCC-annotation $G^\times$. The 1-bisimilarity types of nodes v_1 and v_3 in $G^\times$ are equal; however, identifying these two nodes results in a graph H with a non-trivial SCC (containing at least one edge), so $H^\times$ has at least one inside edge, while $G^\times$ does not. Consequently,

$\mathcal{S}'_\times \preccurlyeq \mathcal{S}_\times$ does not hold, for the modal profiles $\mathcal{S}_\times, \mathcal{S}'_\times$ of $G^\times, H^\times$ of depth 1, as a weak homomorphism must map inside edges to inside edges.

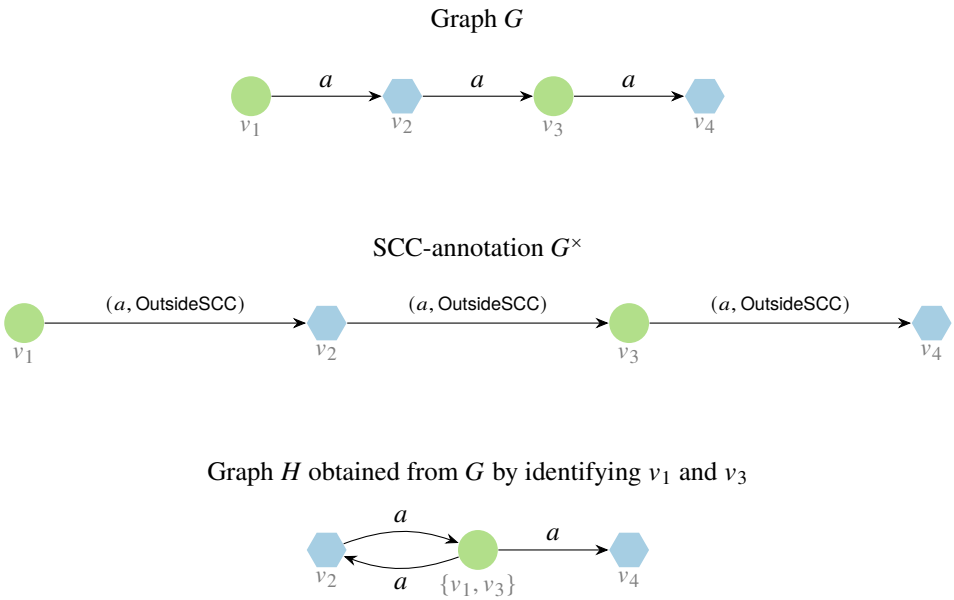


Figure 7.16: An example of graphs G and $G^\times$ for which the strategy of identifying nodes with equal types does not work.

Let $\leq$ be a node-level topological order in G. Intuitively, we want to remove all nodes apart from the greatest (in $\leq$) node of each type, and redirect edges between these nodes to preserve their types. For each $\tau_\times \in \mathcal{T}_\times$, let $v_{\tau_\times}$ be the greatest node v in G such that $\mathsf{Type}^n_{G^\times}(v) = \tau_\times$. Let H be the graph with nodes $\{v_{\tau_\times} : \tau_\times \in \mathcal{T}_\times\}$ and an edge $(v_{\tau_\times}, a, v_{\theta_\times})$ iff there is an edge $(v_{\tau_\times}, a, v)$ in G such that $\mathsf{Type}^n_{G^\times}(v) = \theta_\times$. The label of a node $v_{\tau_\times}$ is $\mathsf{RootLabel}(\tau_\times)$, which is also its label in G.

The number of nodes in H is $|\mathcal{T}_\times|$, and it can be bounded from above in terms of n and $|\mathcal{T}^0_\times|$, because the deannotations of $\mathcal{T}^0_\times$ and $\mathcal{T}_\times$ are equal. This upper bound is computable; it might be enormous, but we note that this is caused by the setting we work in, rather than by the construction itself.

Strongly connected components in H

Claim 7.29. If two nodes $u_0 < v_0$ are in one SCC in H, then they are in the same SCC in G.

Proof. There is a descending edge sequence from v_0 to u_0 in H (by Fact 7.27). By the construction of H, for each edge (u, a, v) in H, there is an edge (u, a, v') in G such that $v' \leq v$, so there is a descending edge sequence from v_0 to u_0 in G as well, so u_0, v_0 are in one SCC in G (by Claim 7.28). □

Claim 7.30. For each edge $(v_{\tau_\times}, a, v_{\theta_\times})$ inside some SCC in H, every edge $(v_{\tau_\times}, a, v)$ in G such that $\mathsf{Type}^n_{G^\times}(v) = \theta_\times$ is inside some SCC in G.

Proof. If $v \leq v_{\tau_\times}$, then, by the properties of the node-level topological order $\leq$, the claim holds; assume $v_{\tau_\times} < v$. By the definition of $v_{\theta_\times}$, we have $v \leq v_{\theta_\times}$. Since $(v_{\tau_\times}, a, v_{\theta_\times})$ is inside some SCC in H, by

Claim 7.29, the nodes $v_{\tau_\times}, v_{\theta_\times}$ are in one SCC in G. By the properties of $\leq$, all nodes between them, including v, are also in the same SCC in G. □

The modal profile of $H^\times$

Let $\mathcal{S}'_\times = (n, \mathcal{T}'_\times)$ be the modal profile of $H^\times$ of depth n. We claim that $\mathcal{S}'_\times \preccurlyeq \mathcal{S}_\times$. To prove it, we show that the n-bisimilarity profiles of G and H are equal, and that for each node u in H there is a weak homomorphism from $\mathsf{Unravel}(n, H^\times, u)$ to $\mathsf{Unravel}(n, G^\times, u)$. Note that these proofs are not directly related to the main result of this thesis; we provide them for completeness, to justify the relevance of the presented construction, but we encourage the reader to skip them.

Claim 7.31. For each node u in H, $\mathsf{Type}^n_H(u) = \mathsf{Type}^n_G(u)$.

Proof. We use the One-Step Bisimulation Lemma for H, with $\mathsf{SupposedType}(u) = \mathsf{Type}^n_G(u)$. The label of each node u in H is the same as in G, so we need to prove, for each edge label a:

$$\mathsf{Subtypes}(\mathsf{Type}^n_G(u), a) = \{\mathsf{Type}^{n-1}_G(v) : v \in \mathsf{Succ}_H(u, a)\}.$$

As usual, we prove two set inclusions. Let $\tau = \mathsf{Type}^n_G(u)$.

We begin by showing that $\mathsf{Subtypes}(\tau, a) \subseteq \{\mathsf{Type}^{n-1}_G(v) : v \in \mathsf{Succ}_H(u, a)\}$. Consider an edge (u, a, v) in G for some node v, and let $\theta_\times = \mathsf{Type}^n_{G^\times}(v)$. By the construction of H, there is an edge in H from u to $v_{\theta_\times}$ such that $\mathsf{Type}^n_{G^\times}(v_{\theta_\times}) = \theta_\times$, which implies that $\mathsf{Type}^n_G(v_{\theta_\times}) = \mathsf{Type}^n_G(v)$ (by Fact 7.8).

Now we show that $\{\mathsf{Type}^{n-1}_G(v) : v \in \mathsf{Succ}_H(u, a)\} \subseteq \mathsf{Subtypes}(\tau, a)$. Consider an edge $(u, a, v_{\theta_\times})$ in H. By the construction of H, $u = v_{\tau_\times}$ for some $\tau_\times$, and there is an edge (u, a, v) in G such that $\mathsf{Type}^n_{G^\times}(u) = \tau_\times$ and $\mathsf{Type}^n_{G^\times}(v) = \theta_\times$. Then, $\mathsf{Type}^n_G(u) = \mathsf{Type}^n_G(v_{\tau_\times})$ and $\mathsf{Type}^n_G(v) = \mathsf{Type}^n_G(v_{\theta_\times})$. □

Because for each $\tau_\times \in \mathcal{T}_\times$ there is a node $v_{\tau_\times}$ in H such that $\mathsf{Type}^n_{G^\times}(v_{\tau_\times}) = \tau_\times$, it follows that the n-bisimilarity profiles of G and H are equal.

Claim 7.32. For each node u in H, there is a weak homomorphism from $\mathsf{Unravel}(n, H^\times, u)$ to $\mathsf{Unravel}(n, G^\times, u)$.

Proof. We prove the claim by induction over n. The node u has the same label in H and in G. For $n = 0$, this ends the proof. For $n \geq 1$, we show that for each edge outgoing from u in $H^\times$ there is a corresponding edge from u in $G^\times$, and use the inductive assumption. For an edge $e = (u, a, v_{\tau_\times})$ outgoing from u in H, by the construction of H, there is an edge (u, a, v) in G such that $\mathsf{Type}^n_{G^\times}(v) = \tau_\times$. By Claim 7.30, if e is inside some SCC in H, then (u, a, v) is inside some SCC in G. By the inductive assumption, there is a weak homomorphism from $\mathsf{Unravel}(n-1, H^\times, v_{\tau_\times})$ to $\mathsf{Unravel}(n-1, G^\times, v_{\tau_\times})$, which is bisimilar to $\mathsf{Unravel}(n-1, G^\times, v)$. If two trees are bisimilar, they are also homomorphically equivalent, so there is a weak homomorphism from $\mathsf{Unravel}(n-1, H^\times, v_{\tau_\times})$ to $\mathsf{Unravel}(n-1, G^\times, v)$. □

The decidability of the $\preccurlyeq$-bounded satisfiability problem for $\mathbb{G}_\times$ follows.

7.6.3 Satisfiability within SCC-reachability annotations

Consider the $\preccurlyeq$-bounded satisfiability problem for $\mathbb{G}_*$; we will show that the construction presented above does not work in this case, and we will show how to adjust it. We will use the adjusted construction to solve the exact (not $\preccurlyeq$-bounded) satisfiability problem for $\mathbb{G}_*$.

$\preccurlyeq$-BOUNDED SATISFIABILITY OF A MODAL PROFILE WITHIN $\mathbb{G}_*$

Input: A modal profile $\mathcal{S}_*^0$.

Question: Is $\mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$ non-empty for some $\mathcal{S}_* \preccurlyeq \mathcal{S}_*^0$?

The construction used in the previous proof does not work for the graph G shown in Figure 7.17. Nodes v_1 and v_2 have the same 1-bisimilarity types in G^*. The graph H arising from the construction from the previous proof can be obtained from G by removing the node v_1, and adding an edge from v_2 to itself. However, the node v_2 does not have any reachable node with label ⬟ in H, so its 1-bisimilarity type in H^+ is not the same as in G^+; consequently, $\mathcal{S}'_* \preccurlyeq \mathcal{S}_*$ does not hold for $\mathcal{S}_*$ and $\mathcal{S}'_*$ being the shallow modal profiles of G^* and H^*, respectively. In this example, the problem could be fixed by considering another node-level topological order, obtained by swapping the places of v_1 and v_2. However, for the graph G' shown in Figure 7.18, there is no such order that would allow us to obtain a suitable graph H' using the previous construction: if we remove v_1, then v_2 stops having a reachable node with label ⬟, and if we remove v_2, then v_1 stops having a reachable node with label ⬟. Our solution is to identify v_1 with v_2 – or, more generally, identify nodes with equal types that are in one SCC.

The reasoning above leads to a construction that allows us to solve the exact (not $\preccurlyeq$-bounded) satisfiability problem for $\mathbb{G}_*$, where a small model can be found by identifying all nodes with equal types within each SCC, and focusing on the greatest (in $\leq$) SCC containing each type, similarly as we focused on the greatest node of each type in the previous proof. Below we use a very similar construction (slightly modified, for a shorter formal definition of edges in the resulting graph) to prove the following proposition.

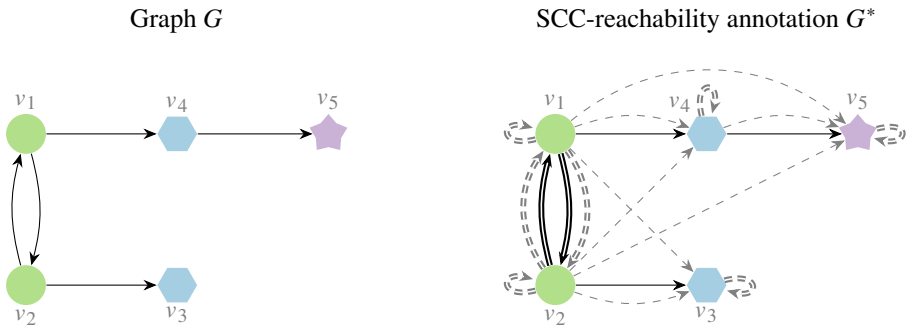
Proposition 7.33. If a modal profile $\mathcal{S}_* = (n, \mathcal{T}_*)$ is satisfiable within $\mathbb{G}_*$, then there exists a graph in $\mathsf{Models}_{\mathbb{G}_*}(\mathcal{S}_*)$ with at most $|\mathcal{T}_*|^2$ nodes.

The decidability of the satisfiability problem for $\mathbb{G}_*$ follows.

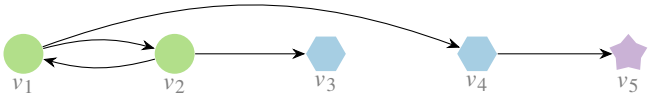
The construction

Consider a modal profile $\mathcal{S}_* = (n, \mathcal{T}_*)$ that is satisfiable within $\mathbb{G}_*$, and let G be a graph such that $G^* \models \mathcal{S}_*$. Let $\mathcal{E}_* = \{(\mathsf{Type}^n_{G^*}(u), a_\times, \mathsf{Type}^n_{G^*}(v)) : (u, a_\times, v) \in E_{G^\times}\}$; note that, because for the construction we are interested only in short edges in G^*, the definition of $\mathcal{E}_*$ uses edges and edge labels in $G^\times$, but n-bisimilarity types in G^*.

For an SCC S_* in G^*, let $\mathsf{Profile}^n_{G^*}(S_*)$ denote the set consisting of $\mathsf{Type}^n_{G^*}(u)$ for all nodes u in S_*. Let $\leq$ be a topological order on SCCs in G^*. For each $\tau_* \in \mathcal{T}_*$, let S_{τ_*} be the greatest (in $\leq$) SCC S_* in G^* such that $\tau_* \in \mathsf{Profile}^n_{G^*}(S_*)$. Let $X = (S_1, \ldots, S_k)$ be the linear ordering of $\{S_{\tau_*} : \tau_* \in \mathcal{T}_*\}$ according to $\leq$. Note that $k \leq |\mathcal{T}_*|$.



Graph G drawn according to a node-level topological order $\leq$



Graph H obtained from G by removing v_1 and adding a self-loop at v_2

Figure 7.17: An example of graphs G and G^*, and a node-level topological order $\leq$ in G, for which the previous construction does not work. All edges in G have the same label, so we omit it.

Graph G'

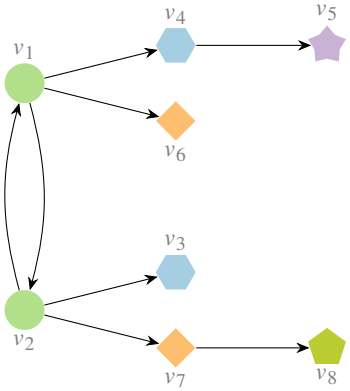


Figure 7.18: An example of a graph G' for which the previous construction does not work, regardless of the chosen node-level topological order. All edges in G' have the same label, so we omit it.

Let H be the graph with a node (i, τ_*) for each $i = 1, \ldots, k$ and each $\tau_* \in \mathsf{Profile}^n_{G^*}(S_i)$, and an edge $((i, \tau_*), a, (j, \theta_*))$ iff either:

- $i = j$ and $(\tau_*, (a, \mathsf{InsideSCC}), \theta_*) \in \mathcal{E}_*$, or
- $i < j$ and $(\tau_*, (a, \mathsf{OutsideSCC}), \theta_*) \in \mathcal{E}_*$.

The label of a node (i, τ_*) is $\mathsf{RootLabel}(\tau_*)$.

Fact 7.34. The graph H has at most $|\mathcal{T}_*|^2$ nodes.

Claim 7.35. Two nodes $(i, \tau_*), (j, \theta_*)$ are in one SCC in H iff $i = j$.

Proof. We begin with the right-to-left implication. Let u, v be some nodes in the SCC S_i such that $\mathsf{Type}^n_{G^*}(u) = \tau_*$ and $\mathsf{Type}^n_{G^*}(v) = \theta_*$. Consider a walk from u to v in S_i; replacing each node w in this walk with $(i, \mathsf{Type}^n_{G^*}(w))$ results in a walk from (i, τ_*) to (i, θ_*) in H.

Now we show the left-to-right implication. By the construction of H, for each edge $((i', \tau'_*), a, (j', \theta'_*))$ in H, $i' \leq j'$. Because (i, τ_*) and (j, θ_*) are in one SCC in H, there is a walk from (i, τ_*) to (j, θ_*), which means that $i \leq j$, and there is a walk from (j, θ_*) to (i, τ_*), which means that $j \leq i$, so $i = j$. □

Claim 7.36. For each node (i, τ_*) in H, $\mathsf{Type}^n_{H^*}((i, \tau_*)) = \tau_*$.

Proof. We use the One-Step Bisimulation Lemma for the graph H^* and the function $\mathsf{SupposedType}$ mapping (i, τ_*) to τ_*. As the node label of (i, τ_*) in H and in H^* is $\mathsf{RootLabel}(\tau_*)$, we only need to show that, for each node (i, τ_*) in H^* and each edge label a_*:

$$\mathsf{Subtypes}(\tau_*, a_*) = \{\theta_*|_{n-1} : (j, \theta_*) \in \mathsf{Succ}_{H^*}((i, \tau_*), a_*)\}.$$

As usual, we show two set inclusions; this time, for each inclusion, we consider four cases: for $a_* = (a_+, z)$, we consider $a_+ = (\mathsf{Long}, \bot)$ or $a_+ = (\mathsf{Short}, a)$, and $z = \mathsf{OutsideSCC}$ or $z = \mathsf{InsideSCC}$.

First, we show that $\mathsf{Subtypes}(\tau_*, a_*) \subseteq \{\theta_*|_{n-1} : (j, \theta_*) \in \mathsf{Succ}_{H^*}((i, \tau_*), a_*)\}$. Consider some node u in S_i such that $\mathsf{Type}^n_{G^*}(u) = \tau_*$. Intuitively, we need to show that for each edge outgoing from u in G^* there is a corresponding edge outgoing from (i, τ_*) in H^*. Consider an edge (u, a_*, v) in G^* for some node v, and let $\theta_* = \mathsf{Type}^n_{G^*}(v)$. Let $a_* = (a_+, z)$.

1. If $a_+ = (\mathsf{Short}, a)$ for some a and $z = \mathsf{InsideSCC}$, the node v is also in the SCC S_i. By the definition of $\mathcal{E}_*$, $(\tau_*, (a, z), \theta_*) \in \mathcal{E}_*$. By the construction of H, (i, θ_*) is a node in H and $((i, \tau_*), a, (i, \theta_*))$ is an edge in H. By Claim 7.35, $((i, \tau_*), (a_+, z), (i, \theta_*))$ is an edge in H^*.
2. If $a_+ = (\mathsf{Short}, a)$ for some a and $z = \mathsf{OutsideSCC}$, let j be such that $S_j = S_{\theta_*}$. We have $i < j$, for the following reason: for S_v being the SCC of v in G^*, we have $S_i < S_v \leq S_j$; the first inequality holds because $z = \mathsf{OutsideSCC}$ and $\leq$ is a topological order on SCCs in G^*, and the second inequality holds because $S_j = S_{\theta_*}$ is the greatest SCC with a node of type θ_*. By the construction of H and by Claim 7.35, $((i, \tau_*), (a_+, z), (j, \theta_*))$ is an edge in H^*.
3. If $a_+ = (\mathsf{Long}, \bot)$ and $z = \mathsf{InsideSCC}$, (i, θ_*) is a node in H, and there is a walk from u to v in G; this whole walk is contained in the SCC S_i. Mapping each node w in this walk to $(i, \mathsf{Type}^n_{G^*}(w))$, we get a walk from (i, τ_*) to (i, θ_*) in H. By Claim 7.35, this walk is contained in one SCC in H, so $((i, \tau_*), (a_+, z), (i, \theta_*))$ is an edge in H^*.

4. If $a_+ = (\mathsf{Long}, \bot)$ and $z = \mathsf{OutsideSCC}$, we perform an inductive reasoning over SCCs $S_1, \ldots, S_k$ in reverse topological order, for $i = k, k-1, \ldots, 1$. More precisely, we prove the claim for a node (i, τ_*) assuming that it holds for all nodes (i', τ'_*) with $i' > i$. We also use the points 2 and 3 above. A schematic picture of the construction is shown in Figure 7.19.

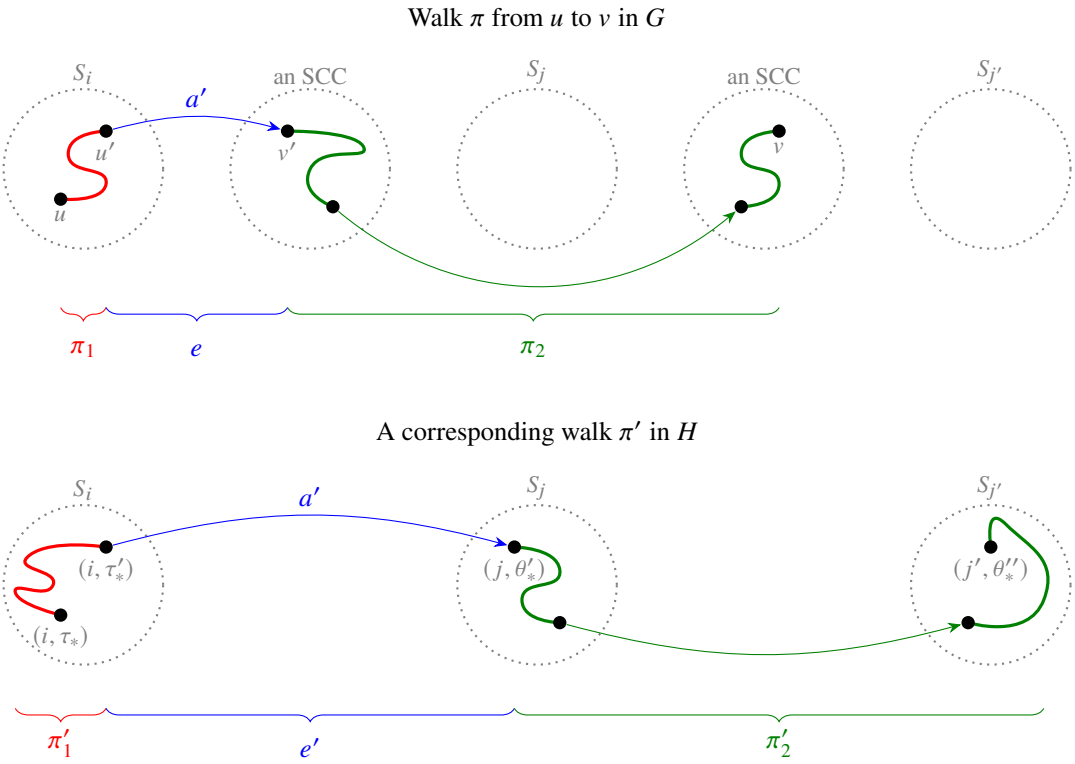


Figure 7.19: A schematic picture of a walk π from u to v in G, and of a corresponding walk in H, split into three parts: the longest prefix contained in one SCC, the next traversed edge, and the remaining suffix.

There is a walk π from u to v in G, and $\mathsf{SCC}_{G^*}(u) < \mathsf{SCC}_{G^*}(v)$; we split π into three parts:

- the longest prefix π_1 of π contained in the SCC S_i;
- the next edge $e = (u', a', v')$ traversed by π;
- the remaining suffix π_2 of π.

Note that π_2 witnesses that v is reachable from v' in G. Let $\tau'_* = \mathsf{Type}^n_{G^*}(u')$ and $\theta'_* = \mathsf{Type}^n_{G^*}(v')$.

- By the point 3 above, there is a walk π'_1 in H from (i, τ_*) to (i, τ'_*).
- By the point 2, there is an edge $e' = ((i, \tau'_*), a', (j, \theta'_*))$ in H, for $j > i$ such that $S_j = S_{\theta'_*}$.

Since v is reachable from v' in G, we have $\theta_*|_{n-1} \in \mathsf{Subtypes}(\theta'_*, ((\mathsf{Long}, \bot), z'))$ for some $z' \in \{\mathsf{InsideSCC}, \mathsf{OutsideSCC}\}$. We claim that there is a walk π'_2 in H from (j, θ'_*) to some node (j', θ''_*) such that $\theta''_*|_{n-1} = \theta_*|_{n-1}$:

- if $z' = \mathsf{InsideSCC}$, it follows from the point 3 above;
- if $z' = \mathsf{OutsideSCC}$, we use the inductive assumption applied to (j, θ'_*).

Concatenating π'_1, e' and π'_2, we obtain a walk from (i, τ_*) to (j', θ''_*); since it traverses the edge e', it is not contained in one SCC in H, so $((i, \tau_*), (a_+, z), (j', \theta''_*))$ is an edge in H^*.

For the proof of the other set inclusion, we will use the following fact, similarly to how we used transitivity of reachability in the case of the satisfiability problem for $\mathbb{G}_+$ at the beginning of this chapter. Consider an edge $(u, (a_+, z), v)$ in G^*, and let $\tau_* = \mathsf{Type}^n_{G^*}(u)$ and $\theta_* = \mathsf{Type}^n_{G^*}(v)$. If $z = \mathsf{InsideSCC}$, exactly the same nodes are reachable from u and from v in G, so we have the following equalities:

- $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})) = \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC}))$;
- $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC})) = \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$.

If $z = \mathsf{OutsideSCC}$, each node in G reachable from v is also reachable from u, and each of them is in a different SCC than u:

- $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})) \subseteq \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$;
- $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC})) \subseteq \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$.

By the definition of $\mathcal{E}_*$, the construction of H, and Claim 7.35, the same set inclusions hold for every short edge $((i, \tau_*), (a_+, z), (j, \theta_*))$ in H^*. For a long edge $((i, \tau_*), ((\mathsf{Long}, \bot), z), (j, \theta_*))$ in H^*, there is a walk from (i, τ_*) to (j, θ_*) in H; applying the above set inclusions to each edge traversed by this walk, we also get the same set inclusions.

Now we prove that $\{\theta_*|_{n-1} : (j, \theta_*) \in \mathsf{Succ}_{H^*}((i, \tau_*), a_*)\} \subseteq \mathsf{Subtypes}(\tau_*, a_*)$. Consider an edge $((i, \tau_*), a_*, (j, \theta_*))$ in H^*. Let $a_* = (a_+, z)$.

- If $a_+ = (\mathsf{Short}, a)$ for some a, then $(\tau_*, (a, z), \theta_*) \in \mathcal{E}_*$, so there is a corresponding edge (u, a_*, v) in G^*, so $\theta_*|_{n-1} \in \mathsf{Subtypes}(\tau_*, a_*)$.
- If $a_+ = (\mathsf{Long}, \bot)$, because each node is reachable from itself, we have

  $$\theta_*|_{n-1} \in \mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})),$$

  and by the set inclusions above we get:

  $$\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})) \subseteq \mathsf{Subtypes}(\tau_*, a_*).$$

□

This finishes the proof of Proposition 7.33, and of the decidability of the satisfiability problem for $\mathbb{G}_*$.

Optimizing the algorithm

In the earlier satisfiability results we showed a construction of a canonical candidate for a model. In this case, it is not clear how to construct such a candidate. Instead, below we describe an algorithm that avoids iterating over exponentially many graphs, and instead uses a polynomial procedure finding the greatest fixed point; the same idea is used in the papers we published (Gogacz et al. 2020; Gutiérrez-Basulto et al. 2022, 2024) to prove a doubly exponential upper bound on the time complexity of the presented algorithms. We only give an overview, without full proofs.

Consider a modal profile $\mathcal{S}_* = (n, \mathcal{T}_*)$. The idea is to construct a graph G_0 such that, if $\mathcal{S}_*$ is satisfiable within $\mathbb{G}_*$, then the graph H constructed in the proof above is a subgraph of G_0. Next, we iteratively remove nodes and edges from G_0 in a way that preserves this property, and eventually results in a graph G such that $G^* \models \mathcal{S}_*$.

Let G_0 be the graph with nodes (i, τ_*) for each $i = 1, 2, \ldots, |\mathcal{T}_*|$ and each $\tau_* \in \mathcal{T}_*$, and an edge $((i, \tau_*), a, (j, \theta_*))$ iff either:

- $i = j$, and:
  - $\theta_*|_{n-1} \in \mathsf{Subtypes}(\tau_*, (a, \mathsf{InsideSCC}))$;
  - $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})) = \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC}))$;
  - $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC})) = \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$;
- $i < j$ and:
  - $\theta_*|_{n-1} \in \mathsf{Subtypes}(\tau_*, (a, \mathsf{OutsideSCC}))$;
  - $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{InsideSCC})) \subseteq \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$;
  - $\mathsf{Subtypes}(\theta_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC})) \subseteq \mathsf{Subtypes}(\tau_*, ((\mathsf{Long}, \bot), \mathsf{OutsideSCC}))$.

The label of a node (i, τ_*) is $\mathsf{RootLabel}(\tau_*)$.

Let G'_0 be the graph obtained from G_0 by removing all edges $((i, \tau_*), a, (j, \theta_*))$ such that $i = j$, but the edge is not contained in an SCC in G_0. Note that we do not have the guarantee that two nodes $(i, \tau_*), (j, \theta_*)$ are in the same SCC iff $i = j$, but the left-to-right implication holds, and we have a similar guarantee for edges in G'_0: an edge $((i, \tau_*), a, (j, \theta_*))$ is contained in some SCC in G'_0 iff $i = j$.

If $\mathcal{S}_*$ is satisfiable within $\mathbb{G}_*$, it is easy to see that the graph H obtained in the decidability proof above is a subgraph of G'_0. Moreover, H^* is a subgraph of $(G'_0)^*$. We will construct a sequence of graphs $G_1, G'_1, G_2, G'_2, \ldots$, with each G'_i being a subgraph of G_i and each G_{i+1} being a subgraph of G'_i, such that H^* is a subgraph of $(G_i)^*$ and $(G'_i)^*$ for each $i = 0, 1, 2, \ldots$.

The graph G_{i+1} is obtained from G'_i by removing all nodes (i, τ_*) such that $\mathsf{Type}^n_{(G'_i)^*}((i, \tau_*)) \neq \tau_*$; intuitively, if this happens, then $\mathsf{Type}^n_{(G'_i)^*}((i, \tau_*))$ is "smaller" than τ_* (formally, there is a weak homomorphism between their representatives), so this node cannot belong to the subgraph H. The graph G'_i is obtained from G_i by removing all edges $((i, \tau_*), a, (j, \theta_*))$ such that $i = j$, but the edge is not inside an SCC in G_i.

Let $G = G_{|\mathcal{T}_*|^2}$. We claim that, if $\mathcal{S}_*$ is satisfiable within $\mathbb{G}_*$, then $G^* \models \mathcal{S}_*$. If $\mathcal{S}_*$ is not satisfiable within $\mathbb{G}_*$, this procedure gives us another interesting result, which is also used when optimizing the algorithm: the n-bisimilarity profile of G^* is the greatest (with respect to containment) subset $\mathcal{T}_*'$ of $\mathcal{T}_*$ such that $(n, \mathcal{T}_*')$ is satisfiable within $\mathbb{G}_*$.

7.7 Summary

We solved the minimal downsets problem for $\mathbb{G}_+$, by reducing it to the satisfiability problem for $\mathbb{G}_*$.

The crucial idea is to extend modal profiles with information about strongly connected components. This builds on ideas developed by Gogacz, Ibañez-García, and Murlak (2018), who consider the finite query entailment problem for a description logic that can express transitivity of a role. The authors reduce the problem to the case without transitive roles, intuitively, by defining a tree automaton recognizing a decomposition of a graph into strongly connected components, and distributing the constraints and the query over the SCCs.

Apart from introducing SCC-annotations, the general structure of the proof is the same as in the previous chapter. We defined reachability-annotated global representatives of modal profiles, constructed graphs with minimal downsets, and solved the satisfiability problem.

To define reachability-annotated global representatives, we introduced the notion of *pseudoextensions* between walks in a graph. It is worth pointing out that, even though the definition of reachability-annotated global representatives is relatively simple, and the combinatorial proofs of their properties are mostly straightforward, this definition is the crucial point in the overall proof. Thus, in some sense, the definition of pseudoextensions can be seen as the most important part of the proof.

To construct graphs with minimal downsets, we introduced the Last Edge Appearance Record of a walk in a graph, and used it with ravelling constructions. The definition is simple, and the resulting construction is very useful – it will be used in the next chapter as well, allowing us to avoid complicated recursive constructions. We also observe a connection between our construction and the "pumping method" used by Baader, Bednarczyk, and Rudolph (2019) in a related setting, for constructing graphs of high girth; there are some similarities between how we use the LEARs of all walks in a graph, and how the authors use all subsets of edges in a graph. In this context, it is worth noting that the LEAR of a walk in a graph G can be thought of as an *ordered* subset of edges in G (up to long edges appearing in the LEAR of a walk, which could likely be omitted in the definition, at the cost of slightly more complicated proofs).

At the beginning of this chapter, we solved the satisfiability problem for $\mathbb{G}_+$, treating it very similarly to the case of the class of all graphs, intuitively, by identifying all nodes of the same type. To solve the satisfiability problem for $\mathbb{G}_*$, we showed that inside each SCC the problem can be treated similarly to the satisfiability problem for $\mathbb{G}_+$; reasoning about the structure of SCCs in the graph requires some care, including an inductive reasoning over SCCs in the reverse topological order in the proofs. We note that this could likely be done as well using tree automata techniques, but the approach we used seems to generalize better to the setting considered in the next chapter.

In the next chapter we use essentially the same techniques in a more general setting of *ranked* graphs. Adjusting the notions and proofs to ranked graphs is quite demanding technically, but the general strategy remains exactly the same.

Chapter 8

Ranked graphs

In this chapter we solve a variant of the minimal downsets problem for *ranked* graphs. This result will be used to prove the main result of this thesis.

A *ranked graph* is a graph in which all edges have labels of the form (a, r), where r is a positive integer, called the *rank* of the edge. In this section we work exclusively with ranked graphs. For a ranked graph G, let $\mathsf{MaxRank}_G$ denote the maximal rank of edges in G, or $\mathsf{MaxRank}_G = 1$ if G has no edges. For an integer $r \in \{1, 2, \ldots, \mathsf{MaxRank}_G\}$, an $r^{\uparrow}$*-walk* in G is a walk traversing only edges of ranks at least r.

The *ranked reachability annotation* of a ranked graph G, denoted $G^{\oplus}$, is the ranked graph obtained from G by replacing each edge label (a, r) with $((\mathsf{Short}, a), r)$, and adding an edge from u to v with label $((\mathsf{Long}, \bot), r)$ iff there is an $r^{\uparrow}$-walk from u to v in G, for $r \in \{1, 2, \ldots, \mathsf{MaxRank}_G\}$. An example of a graph and its ranked reachability annotation is shown in Figure 8.1 on the following page.

Let $\mathbb{G}_{\odot}$ be the class of all ranked graphs, and let $\mathbb{G}_{\oplus} = \{G^{\oplus} : G \in \mathbb{G}_{\odot}\}$ be the class of all ranked reachability annotations. Similarly to the previous chapter, we will also define a class $\mathbb{G}_{\circledast}$ of *ranked SCC-reachability annotations* of ranked graphs. In this chapter we solve the minimal downsets problem for $\mathbb{G}_{\oplus}$, by proving the following propositions.

Proposition 8.1. There is a Turing reduction from the minimal downsets problem for $\mathbb{G}_{\oplus}$ to the satisfiability problem for $\mathbb{G}_{\circledast}$.

Proposition 8.2. The satisfiability problem for $\mathbb{G}_{\circledast}$ is decidable.

The structure of this chapter

The focus of this chapter, rather than developing new techniques, is defining notions suitable for techniques developed in previous chapters.

The overall strategy is exactly the same as in the previous chapter: we define (ranked) SCC-annotations, annotated unravellings and annotated representatives, construct graphs with minimal downsets using ravelling constructions and the Last Edge Appearance Record, and solve the satisfiability problem. We manage to define notions related to ranked graphs in such a way that most of the techniques and

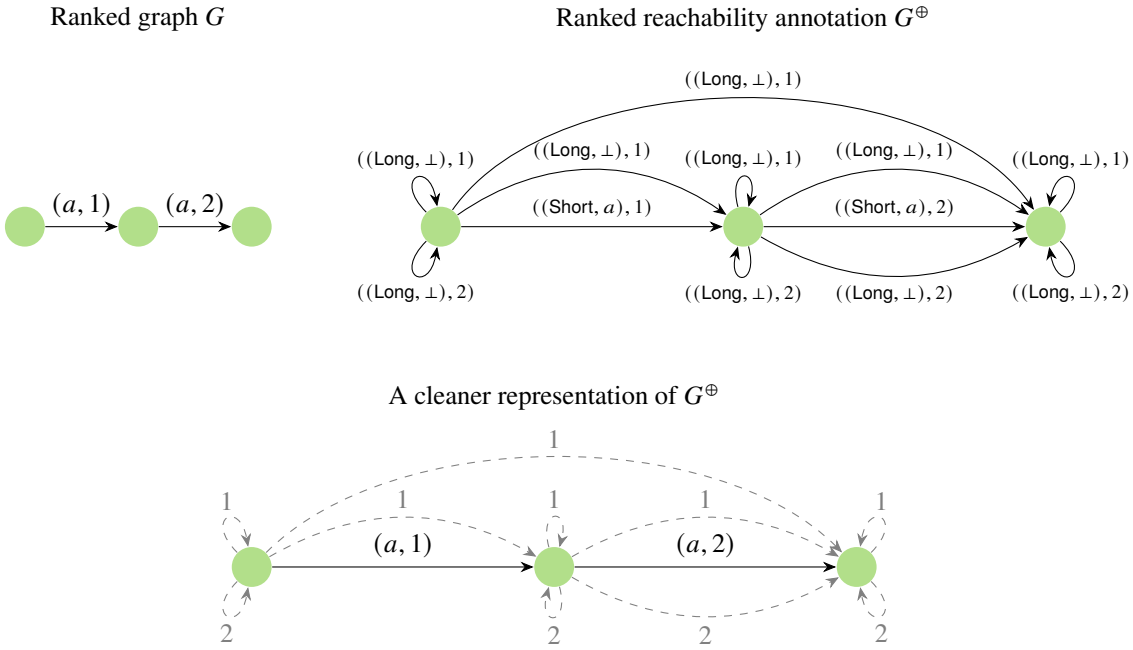


Figure 8.1: A ranked graph G and its ranked reachability annotation $G^{\oplus}$. In the cleaner representation we denote long edges by dashed gray lines, and omit the components Short, Long and $\bot$ in the labels, as in the previous chapter. Note that a single walk in G might correspond to multiple long edges in $G^{\oplus}$.

constructions introduced in the previous chapters generalize naturally to the ranked setting. The biggest adjustments need to be made in the solution of the satisfiability problem, where we split the construction into two steps.

We begin by defining notions related to ranked graphs and their annotations, including *ranked SCCs*, and proving their fundamental properties; while in the previous chapter we could rely on the reader having some intuitions regarding SCCs in unranked graphs, in this chapter we take time to build them.

8.1 Definitions

8.1.1 Ranked graphs

As defined earlier, a ranked graph is a graph in which all edges have labels of the form (a, r), where r is a positive integer, called the rank of the edge. For a ranked graph G, $\mathsf{MaxRank}_G$ denotes the maximal rank of an edge in G, or $\mathsf{MaxRank}_G = 1$ if G has no edges. For an integer $r \in \{1, 2, \ldots, \mathsf{MaxRank}_G\}$, an $r^{\uparrow}$-walk in G is a walk traversing only edges of ranks at least r.

A *ranked modal profile* is a modal profile in which each edge label is of the form (a, r) for some positive integer r.

Ranked SCCs

In the context of a ranked graph G, we use the word *rank* to refer to an integer in $\{1, 2, \ldots, \mathsf{MaxRank}_G\}$, and *weak rank* to refer to an integer in $\{0, 1, \ldots, \mathsf{MaxRank}_G\}$.

Consider a ranked graph G. For a rank r, let $G|_r$ denote the ranked graph obtained from G by removing all edges of ranks strictly smaller than r. Note that an $r^\uparrow$-walk in G can be alternatively defined as a walk in $G|_r$. An *$r^\uparrow$-SCC* in G is an SCC in $G|_r$; thus, each $r^\uparrow$-SCC is a subgraph of G, but not necessarily an *induced* subgraph. Note that $1^\uparrow$-SCCs correspond to (unranked) SCCs. For convenience, we also define the $0^\uparrow$-SCC of G as the whole graph G. By a *ranked SCC* in G we mean an $r^\uparrow$-SCC in G for some weak rank r. An example of a ranked graph and its ranked SCCs is shown in Figure 8.2.

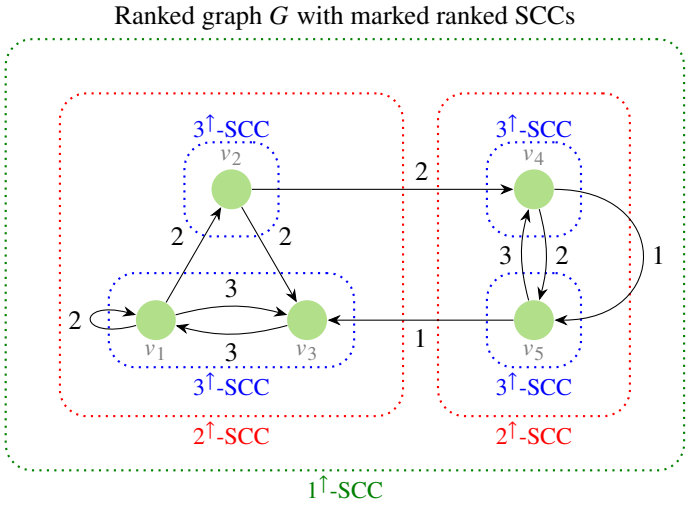


Figure 8.2: A ranked graph G and its ranked SCCs. All edge labels are of the form (a, r) for the same edge label a, so we omit the component a and write only the rank of each edge. Note that the loop at v_1, of rank 2, does not belong to the $3^\uparrow$-SCC of v_1, but only to its $2^\uparrow$-SCC and $1^\uparrow$-SCC. Similarly, the edge of rank 1 from v_4 to v_5 does not belong to the $2^\uparrow$-SCC of v_4 and v_5, but only to their $1^\uparrow$-SCC.

For a node u and a rank r in G, we denote by $\mathsf{SCC}^r_G(u)$ the $r^\uparrow$-SCC of u in G (the SCC of u in $G|_r$).

Note that the nested structure of ranked SCCs in a graph can be represented as a tree, with nodes being all ranked SCCs, including the $0^\uparrow$-SCC as the root, and edges from an $r^\uparrow$-SCC S to all $(r+1)^\uparrow$-SCCs contained in S. We could also extend this tree by adding all nodes in G as leaves, with edges from each $(\mathsf{MaxRank}_G)^\uparrow$-SCC to all nodes contained in it. We do not use this tree representation directly, but the intuitions related to this nested structure of ranked SCCs are important, especially for recursive constructions.

We note that ranked SCCs of *greater* ranks are *smaller* graphs (inclusion-wise). Unfortunately, if we adjusted the definitions to avoid this mismatch, some crucial parts of reasoning would become less intuitive, including the reduction in the next chapter.

The *SCC-rank* of an edge e in G is the maximal weak rank s such that e belongs to some $s^\uparrow$-SCC in G. Note that the SCC-rank of an edge is similar to the information in unranked graphs whether an edge is inside some SCC or outside all SCCs; in particular, an edge of SCC-rank 0 does not belong to any (unranked) SCC. Note also that the rank of an edge is always greater than or equal to its SCC-rank.

For two nodes u, v in G, we denote by $\kappa_G(u, v)$ the maximal weak rank s such that u, v are in one $s^\uparrow$-SCC in G. Equivalently, $\kappa_G(u, v)$ is the rank of the *smallest* (inclusion-wise) common ranked SCC

of u and v in G. To avoid confusing names based on minimality or maximality, we opt for a neutral symbol κ. This technical notion is crucial for the proofs, and we will use it extensively to show that ranked graphs and their annotations are well-behaved[1].

Fact 8.3. For each edge $(u, (a, r), v)$ of SCC-rank s in G, $s = \min\{r, \kappa_G(u, v)\}$.

One can interpret the fact above as follows. For an edge e and a weak rank s in a ranked graph G, there are two independent reasons why e might not belong to any $s^{\uparrow}$-SCC: either the rank of e is strictly smaller than s, which means that e is not an edge in the graph $G|_s$; or the endpoints of e are in different $s^{\uparrow}$-SCCs in G. The independence of these two reasons requires us to consider two separate cases in multiple proofs in this chapter, but it also leads to elegant properties of the ranked SCC-annotations defined later.

Ranks and SCC-ranks of walks

The *rank of a walk* π in a ranked graph G is the minimal rank of edges traversed by π, or $\mathsf{MaxRank}_G$ for a trivial walk. Equivalently, the rank of π is the maximal rank r such that π is an $r^{\uparrow}$-walk. The *SCC-rank of a walk* π in G is the minimal SCC-rank of edges traversed by π, or $\mathsf{MaxRank}_G$ for a trivial walk. Equivalently, the SCC-rank of π is the maximal weak rank s such that π is contained in one $s^{\uparrow}$-SCC in G. Note that the rank of a walk is always greater than or equal to its SCC-rank, because the same inequality holds for every edge traversed by the walk.

From the definitions, it immediately follows that the rank of the concatenation of two walks is the minimal rank of the walks, and similarly for SCC-ranks, which is formalized in the fact below.

Fact 8.4. In a ranked graph, for a walk π_1 of rank r_1 and SCC-rank s_1, a walk π_2 of rank r_2 and SCC-rank s_2, and their concatenation $\pi_1 \cdot \pi_2$ of rank r and SCC-rank s, $r = \min\{r_1, r_2\}$ and $s = \min\{s_1, s_2\}$.

In unranked graphs, informally, once a walk leaves some SCC, it cannot come back. Formally, if a walk in a graph begins and ends in the same SCC, then the whole walk is contained in this SCC. Note that this allows us to determine whether a walk is contained in one SCC by looking only at its endpoints. In ranked graphs, for a rank s, informally, once a walk leaves some $s^{\uparrow}$-SCC, it needs to traverse an edge of rank strictly smaller than s to come back. Taking this observation slightly further, we get the following claim, which states that the earlier equation involving the rank and the SCC-rank of an edge holds also for walks.

Claim 8.5. For a walk π of rank r and SCC-rank s from a node u to a node v in a ranked graph G, $s = \min\{r, \kappa_G(u, v)\}$.

Proof. The claim holds for trivial walks, as $\mathsf{MaxRank}_G = \min\{\mathsf{MaxRank}_G, \mathsf{MaxRank}_G\}$, so assume that π is not trivial. Moreover, from unranked graphs we know that $\kappa_G(u, v) = 0$ iff $s = 0$; that is, the nodes u, v are in different (unranked) SCCs in G iff π traverses some edge not contained in any SCC in

[1]We conjecture that this notion is related to another technical notion used in (Canavoi and Otto 2017), described as "the minimal group of agents that connects the worlds" in a multi-modal Kripke frame, in the context of "multi-agent epistemic modal logic with common knowledge modalities for groups of agents". Pursuing this conjecture might reveal deeper connections between the results, which could be very fruitful, since mathematical tools used in the paper are significantly different from the ones we use.

G. The claim holds in this case, so assume $s \geq 1$ and $\kappa_G(u, v) \geq 1$. We will show two inequalities, $s \leq \min\{r, \kappa_G(u, v)\}$ and $s \geq \min\{r, \kappa_G(u, v)\}$.

To see that $s \leq \min\{r, \kappa_G(u, v)\}$, observe that $r \geq s$, as the rank of a walk is always greater than or equal to its SCC-rank, and $\kappa_G(u, v) \geq s$, as π is a walk from u to v contained in one $s^{\uparrow}$-SCC in G.

For the other inequality, let $s' = \kappa_G(u, v)$; we want to prove that $s \geq \min\{r, s'\}$. Since u, v are in one $(s')^{\uparrow}$-SCC in G and $s' \geq 1$, there is an $(s')^{\uparrow}$-walk π' from v to u in G. Concatenating π with π', we get a *closed* walk of rank at least $\min\{r, s'\}$, so the whole walk is contained in one $(\min\{r, s'\})^{\uparrow}$-SCC in G; in particular, this holds for the prefix π, so $s \geq \min\{r, s'\}$. □

Note that, similarly to the unranked case, the claim above allows us to determine the SCC-rank of a walk based only on its endpoints and rank, without knowing the exact SCC-ranks of traversed edges.

8.1.2 Ranked reachability annotations

As defined earlier, the ranked reachability annotation of a ranked graph G, denoted $G^{\oplus}$, is the ranked graph obtained from G by replacing each edge label (a, r) with $((\mathsf{Short}, a), r)$, and adding an edge from u to v with label $((\mathsf{Long}, \bot), r)$ iff there is an $r^{\uparrow}$-walk from u to v in G, for $r \in \{1, 2, \ldots, \mathsf{MaxRank}_G\}$.

Fact 8.6. For nodes u, v and a rank r in a ranked graph G, the following are equivalent:

- there is an $r^{\uparrow}$-walk from u to v in G;
- there is an edge $(u, ((\mathsf{Long}, \bot), r), v)$ in $G^{\oplus}$;
- there is an $r^{\uparrow}$-walk from u to v in $G^{\oplus}$.

Annotation as a trace-based graph transformation

Similarly to the unranked case, for each non-negative integer n, a single trace-based graph transformation can be used to define the ranked reachability annotations of all graphs with at most n nodes. Applying Fact 5.8 on page 60 and Corollary 5.12 on page 61 to this transformation, we get the following.

Fact 8.7. For ranked graphs G, H, every homomorphism from G to H is also a homomorphism from $G^{\oplus}$ to $H^{\oplus}$.

Fact 8.8. For a ranked graph G, a G-ravelling function f, and $H = \mathsf{Ravel}_f(G)$, the n-bisimilarity profiles of $G^{\oplus}$ and $H^{\oplus}$ are equal for each non-negative integer n.

Note that we do not redefine the notion of a homomorphism for ranked graphs: a homomorphism must preserve edge labels, including ranks of edges.

Ranked short-long graphs

A *long edge label* is $((\mathsf{Long}, \bot), r)$ for some positive integer r. A *short edge label* is $((\mathsf{Short}, a), r)$ for some edge label a and some positive integer r. *Long edges* and *short edges* are edges with long and short edge labels, respectively. A *ranked short-long graph* is a ranked graph in which all edges are either short or long. A *ranked short-long modal profile* is a ranked modal profile in which all edge labels are either short or long.

8.1.3 Ranked SCC-annotations

The *ranked SCC-annotation* of a ranked graph G, denoted $G^{\otimes}$, is the ranked graph obtained from G by replacing the label (a, r) of each edge with $((a, s), r)$, where s is the SCC-rank of this edge in G. We omit the inner parentheses and refer to such edge labels as (a, s, r). We note that the order of components s, r in edge labels is chosen so that $G^{\otimes}$ can be treated formally as a ranked graph, but it might also be more intuitive, since $s \leq r$. An example is shown in Figure 8.3.

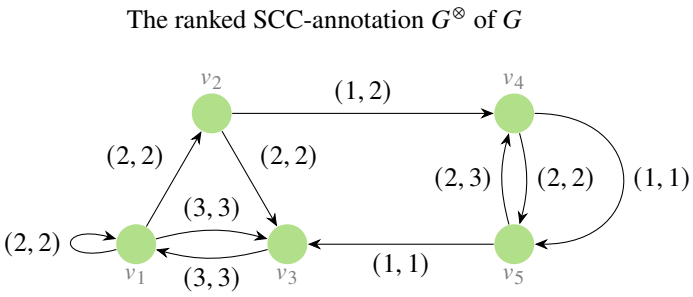


Figure 8.3: The ranked SCC-annotation of the graph G from Figure 8.2 on page 119; again, we omit the common component a in edge labels, and write only the remaining part (s, r). For most edges in G, the SCC-rank s is equal to the rank r; the exceptions are edges from v_2 to v_4 and from v_5 to v_4, where $s < r$.

Doubly-ranked graphs

An edge label is *doubly-ranked* if it is of the form (a, s, r) for some edge label a, positive integer r, and $s \in \{0, 1, \ldots, r\}$. A graph $G_{\otimes}$ is *doubly-ranked* if all its edges have doubly-ranked labels. A modal profile $\mathcal{S}_{\otimes}$ is *doubly-ranked* if all its edge labels are doubly-ranked.

Deannotations

The *deannotation* of a doubly-ranked edge label (a, s, r) is (a, r). The deannotation of a doubly-ranked graph $G_{\otimes}$ is obtained from $G_{\otimes}$ by replacing each edge label with its deannotation. The deannotation of a doubly-ranked modal profile $\mathcal{S}_{\otimes}$ is obtained from $\mathcal{S}_{\otimes}$ by replacing each edge label with its deannotation.

Doubly-ranked annotations of ranked modal profiles

A *doubly-ranked annotation* of a ranked modal profile $\mathcal{S}$ is any doubly-ranked modal profile $\mathcal{S}_{\otimes}$ whose deannotation is $\mathcal{S}$.

Fact 8.9. For a ranked modal profile $\mathcal{S}$, the set of all doubly-ranked annotations of $\mathcal{S}$ is finite and computable.

Fact 8.10. For a ranked graph G and a modal profile $\mathcal{S}$, $G \models \mathcal{S}$ iff $\mathcal{S}$ is a ranked modal profile and $G^{\otimes} \models \mathcal{S}_{\otimes}$ for some doubly-ranked annotation $\mathcal{S}_{\otimes}$ of $\mathcal{S}$.

Ranked weak homomorphism

For doubly-ranked graphs $G_\otimes, H_\otimes$, a *ranked weak homomorphism* from $G_\otimes$ to $H_\otimes$ is, intuitively, a homomorphism from $G_\otimes$ to $H_\otimes$ that is additionally allowed to map an edge with label (a, s, r) to an edge with label (a, s', r) for $s' \geq s$. Formally, it is a function h from $V_{G_\otimes}$ to $V_{H_\otimes}$ such that:

- the label of each node u in $G_\otimes$ is the same as the label of $h(u)$ in $H_\otimes$;
- for each edge $(u, (a, s, r), v)$ in $G_\otimes$, there is an edge $(h(u), (a, s', r), h(v))$ for some $s' \geq s$.

Claim 8.11. For ranked graphs G, H, there is a ranked weak homomorphism from $G^\otimes$ to $H^\otimes$ iff $G \to H$.

Proof. The left-to-right implication is trivial. For the right-to-left implication, a homomorphism h from G to H maps each $r^\uparrow$-walk in G to an $r^\uparrow$-walk in H, so if two nodes u, v are in one $r^\uparrow$-SCC in G, then $h(u), h(v)$ are in one $r^\uparrow$-SCC in H. Thus, the SCC-rank of an edge $(u, (a, r), v)$ in G is smaller than or equal to the SCC-rank of $(h(u), (a, r), h(v))$ in H. □

8.1.4 Ranked SCC-reachability annotations

For a ranked graph G, we define the *ranked SCC-reachability annotation* of G, denoted $G^\circledast$, as $(G^\oplus)^\otimes$. Let $\mathbb{G}_\circledast = \{G^\circledast : G \in \mathbb{G}_\odot\}$ be the class of all ranked SCC-reachability annotations.

A *doubly-ranked short-long graph* is a doubly-ranked graph whose deannotation is a ranked short-long graph. In a doubly-ranked short-long graph, an edge is *long* if its label is of the form $((\mathsf{Long}, \bot), s, r)$, and an edge is *short* if its label is of the form $((\mathsf{Short}, a), s, r)$. Similarly, a *doubly-ranked short-long modal profile* is a modal profile whose deannotation is a ranked short-long modal profile.

Fact 8.12. For ranked graphs G, H, there is a ranked weak homomorphism from $G^\circledast$ to $H^\circledast$ iff $G \to H$.

Fact 8.13 (*Ranked SCC Invariance Under Annotations*)**.** For nodes u, v and a rank r in a ranked graph G, the following statements are equivalent:

- $\mathsf{SCC}^r_G(u) = \mathsf{SCC}^r_G(v)$;
- $\mathsf{SCC}^r_{G^\oplus}(u) = \mathsf{SCC}^r_{G^\oplus}(v)$;
- $\mathsf{SCC}^r_{G^\otimes}(u) = \mathsf{SCC}^r_{G^\otimes}(v)$;
- $\mathsf{SCC}^r_{G^\circledast}(u) = \mathsf{SCC}^r_{G^\circledast}(v)$.

Properties of ranked SCC-reachability annotations

Before we move on to the reduction from the minimal downsets problem for $\mathbb{G}_\oplus$ to the satisfiability problem for $\mathbb{G}_\circledast$, we prove two technical properties of ranked SCC-reachability annotations that show that they are a well-behaved notion.

As mentioned at the beginning of this chapter, a single walk of rank r in a ranked graph G might correspond to multiple long edges in $G^\oplus$; more precisely, for a walk from u to v of rank r in G, there are long edges $(u, ((\mathsf{Long}, \bot), r'), v)$ in $G^\oplus$ for each $r' \in \{1, \ldots, r\}$. Moreover, the SCC-ranks of these long edges in $G^\oplus$ might differ, because the SCC-rank of an edge is always smaller than or equal to its rank. For this reason, the correspondence between walks in G and long edges in $G^\circledast$ is not obvious: for each walk from u to v in G of rank r and SCC-rank s, there are long edges $(u, ((\mathsf{Long}, \bot), s', r'), v)$ in

$G^{\circledast}$ for each $r' \in \{1, \ldots, r\}$ and $s' = \min\{r', s\}$. In the claim below we state the correspondence in the other direction. We provide the proof for completeness, but we encourage the reader to skip it, as the proof is technical and it is not likely to deliver new intuitions.

Claim 8.14. For nodes u, v, a rank r and a weak rank s in a ranked graph G, there is an edge $(u, ((\mathsf{Long}, \bot), s, r), v)$ in $G^{\circledast}$ iff either:

- $s < r$ and there is an $r^{\uparrow}$-walk of SCC-rank exactly s from u to v in G, or
- $s = r$ and there is an $r^{\uparrow}$-walk of SCC-rank at least s from u to v in G.

Proof. We begin with the left-to-right implication. By the definition of $G^{\circledast}$, s is the SCC-rank of the edge $(u, ((\mathsf{Long}, \bot), r), v)$ in $G^{\oplus}$. Using Fact 8.3 on page 120 and the Ranked SCC Invariance Under Annotations, we get $s = \min\{r, \kappa_G(u, v)\}$. By Fact 8.6 on page 121, there exists an $r^{\uparrow}$-walk π from u to v in G; let r' be the rank of π. Note that $r' \geq r$. The SCC-rank of π is $s' = \min\{r', \kappa_G(u, v)\}$.

- If $s < r$, then $s = \kappa_G(u, v) = s'$, so π is an $r^{\uparrow}$-walk of SCC-rank exactly s.
- If $s = r$, then $\kappa_G(u, v) \geq s$ and $r' \geq r = s$, so π is an $r^{\uparrow}$-walk of SCC-rank at least s.

Now we show the right-to-left implication. In both cases there is an $r^{\uparrow}$-walk π from u to v in G, so $e_{\oplus} = (u, ((\mathsf{Long}, \bot), r), v)$ is an edge in $G^{\oplus}$ of SCC-rank $\min\{r, \kappa_G(u, v)\}$; we need to prove that this SCC-rank is equal to s. Let r' be the rank of π; note that $r' \geq r$, and that the SCC-rank of π is $\min\{r', \kappa_G(u, v)\}$.

If $s < r$, the SCC-rank of π is exactly s. Since $s = \min\{r', \kappa_G(u, v)\}$ and $s < r \leq r'$, $s = \kappa_G(u, v)$, so the SCC-rank of $e_{\oplus}$ in $G^{\oplus}$ is $\min\{r, \kappa_G(u, v)\} = s$.

If $s = r$, the SCC-rank of π is at least s. The SCC-rank of π is $\min\{r', \kappa_G(u, v)\} \geq s$. Thus, $\kappa_G(u, v) \geq s$, so the SCC-rank of $e_{\oplus}$ in $G^{\oplus}$ is $\min\{r, \kappa_G(u, v)\} = s$. □

In the statement of Fact 8.4 on page 120 we observed that the rank of the concatenation of two walks is equal to the minimum of their ranks, and, similarly, the SCC-rank of the concatenation is equal to the minimum of their SCC-ranks. As the last property of ranked SCC-reachability annotations, we show a similar property for long edges in the graph $G^{\circledast}$. Again, we encourage the reader to skip the proof.

Claim 8.15. For three nodes u, v, w in a ranked graph G and two edges in $G^{\circledast}$:

- from u to v with a label $((\mathsf{Long}, \bot), s_1, r_1)$,
- from v to w with a label $((\mathsf{Long}, \bot), s_2, r_2)$,

there is an edge in $G^{\circledast}$ from u to w with the label $((\mathsf{Long}, \bot), s, r)$ for $r = \min\{r_1, r_2\}$, $s = \min\{s_1, s_2\}$.

Proof. There is a walk π_1 from u to v of rank $r_1' \geq r_1$ and SCC-rank $s_1' \geq s_1$, with $s_1 = s_1'$ if $s_1 < r_1$. Similarly, there is a walk π_2 from v to w of rank $r_2' \geq r_2$ and SCC-rank $s_2' \geq s_2$, with $s_2 = s_2'$ if $s_2 < r_2$. We claim that the walk π obtained by concatenating π_1 and π_2 is a walk from u to w of rank $r' \geq r$ and SCC-rank $s' \geq s$, with $s' = s$ if $s < r$.

We have $r' \geq r$, $s' \geq s$, as $r' = \min\{r_1', r_2'\} \geq \min\{r_1, r_2\} = r$, and $s' = \min\{s_1', s_2'\} \geq \min\{s_1, s_2\} = s$.

It remains to prove that $s' = s$ if $s < r$. We consider several cases, using equalities $s' = \min\{s_1', s_2'\}$ and $s = \min\{s_1, s_2\}$.

- If $s_1 < r_1$ and $s_2 < r_2$, then $s_1 = s_1'$ and $s_2 = s_2'$, so $s' = \min\{s_1', s_2'\} = \min\{s_1, s_2\} = s$.

- The case $s_1 = r_1$ and $s_2 = r_2$ is impossible, as $s < r$, which means that $\min\{s_1, s_2\} < \min\{r_1, r_2\}$.
- If $s_1 < r_1$ and $s_2 = r_2$, then $s_1 = s'_1$ and:
  - $s_1 < s_2$, as otherwise $r_2 = s_2 \leq s_1 < r_1$, so $r = s$, which contradicts $r < s$;
  - $s'_1 < s'_2$, because $s'_1 = s_1 < s_2 \leq s'_2$;
  - $s' = s'_1 = s_1 = s$, by the (in)equalities above.
- If $s_1 = r_1$ and $s_2 < r_2$, then $s_2 = s'_2$ and, symmetrically to the point above, $s_2 < s_1$ because $r < s$, $s'_2 < s'_1$ because $s'_2 = s_2 < s_1 \leq s'_1$, so $s' = s'_2 = s_2 = s$.

□

The properties of ranked SCCs and ranked SCC-reachability annotations of ranked graphs presented in this section show that they are well-behaved concepts, even though low-level proofs often require considering multiple cases.

8.2 Reduction to satisfiability

The proof strategy is exactly the same as in the previous chapter, and almost all notions and proofs are straightforward generalizations of their counterparts in the previous chapter. We define an analogue of global representatives, and the corresponding function $\mathsf{GlobalRepresentative}_{\circledast \to \oplus}$.

We use the following reduction. If the given modal profile $\mathcal{S}_\oplus = (n, \mathcal{T}_\oplus)$ is not a ranked modal profile, the result is the empty set. Otherwise, we define $\mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$ as the set consisting of $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{\circledast \to \oplus}(\mathcal{S}_\circledast))$ for all doubly-ranked annotations $\mathcal{S}_\circledast$ of $\mathcal{S}_\oplus$ that are satisfiable within $\mathbb{G}_\circledast$. The result is the set $\mathsf{MinComputedDownsets}_\oplus(\mathcal{S}_\oplus)$ of minimal downsets in $\mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$.

8.2.1 Unravellings

Pseudoextensions

Recall that in the previous chapter we defined the notion of a *pseudoextension* as follows. Consider two walks π_1, π_2 in a graph G. Let π_0 be the walk obtained from π_1 by removing the longest suffix contained in one SCC in G. Then, π_2 is a pseudoextension of π_1 if π_0 is a prefix of π_2; in other words, if π_2 is an extension of π_0. Below we adjust the notions of extensions and pseudoextensions to ranked graphs.

For walks π_1, π_2 and a rank r in a ranked graph G, we say that π_2 is an $r^\uparrow$-*extension* of π_1 if π_2 can be obtained by concatenating π_1 with some $r^\uparrow$-walk. The walk π_2 is an $r^\uparrow$-*pseudoextension* of π_1 if π_2 is an $r^\uparrow$-extension of the walk obtained from π_1 by removing the longest suffix contained in one $r^\uparrow$-SCC. Equivalently, π_2 is an $r^\uparrow$-pseudoextension of π_1 if π_1, π_2 begin in the same node and, for the longest common prefix π_0 of π_1 and π_2:

- π_2 is an $r^\uparrow$-extension of π_0, and
- the suffix of π_1 obtained by removing the prefix π_0 is contained in one $r^\uparrow$-SCC in G.

Fact 8.16. For walks π_1, π_2 and a rank r in a ranked graph, if π_2 is an $r^\uparrow$-extension of π_1, then π_2 is an $r^\uparrow$-pseudoextension of π_1.

Fact 8.17. For walks π_1, π_2 and a rank r in a ranked graph, if π_2 is an $r^{\uparrow}$-pseudoextension of π_1, then π_2 is an $s^{\uparrow}$-pseudoextension of π_1 for all ranks $s \leq r$.

Fact 8.18 (Pseudoextension transitivity)**.** For walks π_1, π_2, π_3 and ranks r_1, r_2 in a ranked graph, if π_2 is an $(r_1)^{\uparrow}$-pseudoextension of π_1 and π_3 is an $(r_2)^{\uparrow}$-pseudoextension of π_2, then π_3 is an $(r_3)^{\uparrow}$-pseudoextension of π_1, for $r_3 = \min\{r_1, r_2\}$.

Unravellings

For a node v_0 in a ranked graph G and a non-negative integer n, the *ranked reachability-annotated n-step unravelling* of $G^{\oplus}$ from v_0, denoted $\mathsf{Unravel}_{\oplus}(n, G^{\oplus}, v_0)$, is the ranked short-long graph obtained from $\mathsf{Unravel}(n, G^{\oplus}, v_0)$ by adding a long edge of rank r from $\pi_{\oplus}$ to $\pi'_{\oplus}$ if $\pi'_{\oplus}$ is an $r^{\uparrow}$-pseudoextension of $\pi_{\oplus}$ in $G^{\oplus}$. An example of ranked graphs G, $G^{\oplus}$, $\mathsf{Unravel}(1, G^{\oplus}, u)$ and $\mathsf{Unravel}_{\oplus}(1, G^{\oplus}, u)$ is shown in Figure 8.4.

The graph $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$ is the union of $\mathsf{Unravel}_{\oplus}(n, G^{\oplus}, u)$ for all nodes u in G. Equivalently, $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$ is the ranked short-long graph with nodes being all walks of length at most n in $G^{\oplus}$, each having the label of its last node in $G^{\oplus}$, a short edge $(\pi_{\oplus}, (\mathsf{Short}, a, r), \pi'_{\oplus})$ iff $\pi'_{\oplus} \in \mathsf{Extensions}_{G^{\oplus}}(\pi_{\oplus}, (\mathsf{Short}, a, r))$, and a long edge of rank r from $\pi_{\oplus}$ to $\pi'_{\oplus}$ iff $\pi'_{\oplus}$ is an $r^{\uparrow}$-pseudoextension of $\pi_{\oplus}$ in $G^{\oplus}$.

Claim 8.19. For a ranked graph G and a non-negative integer n, $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus}) \to G^{\oplus}$.

Proof. Let $H_{\oplus} = \mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$, and let h map each node $\pi_{\oplus}$ in $H_{\oplus}$ to $\mathsf{LastNode}(\pi_{\oplus})$. We claim that h is a homomorphism from $H_{\oplus}$ to $G^{\oplus}$. It is enough to prove that h preserves long edges, because $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$ is obtained from $\mathsf{GlobalUnravel}(n, G^{\oplus})$ by adding some long edges, and h is a homomorphism from $\mathsf{GlobalUnravel}(n, G^{\oplus})$ to $G^{\oplus}$ (by Fact 6.3 on page 66).

Consider a long edge of rank r from $\pi_{\oplus}$ to $\pi'_{\oplus}$ in $H_{\oplus}$; then, $\pi'_{\oplus}$ is an $r^{\uparrow}$-pseudoextension of $\pi_{\oplus}$ in $G^{\oplus}$. Let $\sigma_{\oplus}$ be the walk obtained from $\pi_{\oplus}$ by removing the longest suffix contained in one $r^{\uparrow}$-SCC in $G^{\oplus}$; this means that $\mathsf{LastNode}(\sigma_{\oplus})$ and $\mathsf{LastNode}(\pi_{\oplus})$ are in one $r^{\uparrow}$-SCC in $G^{\oplus}$, so there is an $r^{\uparrow}$-walk in $G^{\oplus}$ from $\mathsf{LastNode}(\sigma_{\oplus})$ to $\mathsf{LastNode}(\pi_{\oplus})$. Additionally, by the definition of an $r^{\uparrow}$-pseudoextension, $\pi'_{\oplus}$ is an $r^{\uparrow}$-extension of $\sigma_{\oplus}$, so there is an $r^{\uparrow}$-walk in $G^{\oplus}$ from $\mathsf{LastNode}(\sigma_{\oplus})$ to $\mathsf{LastNode}(\pi'_{\oplus})$. By concatenating these two walks we obtain an $r^{\uparrow}$-walk in $G^{\oplus}$ from $\mathsf{LastNode}(\pi_{\oplus})$ to $\mathsf{LastNode}(\pi'_{\oplus})$, so there is a long edge of rank r from $\mathsf{LastNode}(\pi_{\oplus}) = h(\pi_{\oplus})$ to $\mathsf{LastNode}(\pi'_{\oplus}) = h(\pi'_{\oplus})$ in $G^{\oplus}$ (by Fact 8.6 on page 121). □

8.2.2 Representatives

For a doubly-ranked short-long tree $T_{\circledast}$, we define $\mathsf{RecoverReachability}_{\odot}(T_{\circledast})$ as the ranked short-long graph with:

- the same nodes as $T_{\circledast}$, with the same labels;
- a short edge $(u, ((\mathsf{Short}, a), r), v)$ iff there is an edge $(u, ((\mathsf{Short}, a), s, r), v)$ in $T_{\circledast}$ for some s;
- a long edge of rank r from u to v iff the walk from $\mathsf{LCA}_{T_{\circledast}}(u, v)$ to u traverses only edges with labels of the form $(a_{\circledast}, s, r')$ with $s \geq r$, and the walk from $\mathsf{LCA}_{T_{\circledast}}(u, v)$ to v is an $r^{\uparrow}$-walk.

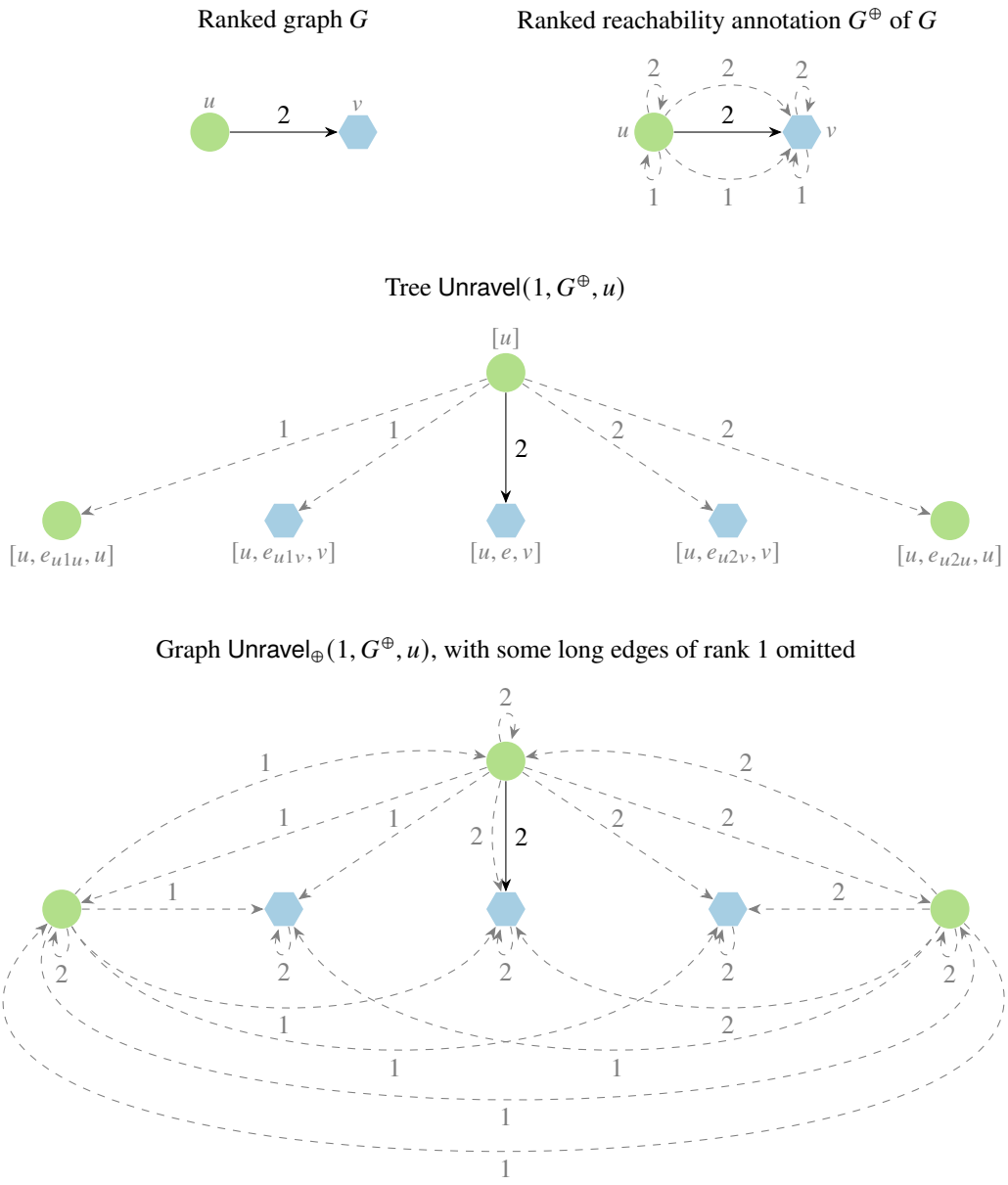


Figure 8.4: An example of ranked graphs G, $G^{\oplus}$, $\mathsf{Unravel}(1, G^{\oplus}, u)$ and $\mathsf{Unravel}_{\oplus}(1, G^{\oplus}, u)$. As usual, we omit the component a in edge labels, and write only their ranks r. We denote the unique short edge in $G^{\oplus}$ by e, and by e_{xry} the unique long edge of rank r from a node x to a node y in $G^{\oplus}$. In $\mathsf{Unravel}_{\oplus}(1, G^{\oplus}, u)$ we omit node identifiers and some long edges of rank 1; namely, for each long edge of rank 2, there should be a parallel edge of rank 1.

For a doubly-ranked short-long forest $F_{\circledast}$, we define $\mathsf{RecoverReachability}_{\odot}(F_{\circledast})$ as the union of $\mathsf{RecoverReachability}_{\odot}(T_{\circledast})$ for all (weakly) connected components $T_{\circledast}$ in $F_{\circledast}$.

The *ranked reachability-annotated global representative* of a doubly-ranked short-long modal profile $\mathcal{S}_{\circledast}$, denoted $\mathsf{GlobalRepresentative}_{\circledast \to \oplus}(\mathcal{S}_{\circledast})$, is $\mathsf{RecoverReachability}_{\odot}(\mathsf{GlobalRepresentative}(\mathcal{S}_{\circledast}))$.

Claim 8.20. For a node v_0 in a ranked graph G and a non-negative integer n, $\mathsf{Unravel}_{\oplus}(n, G^{\oplus}, v_0)$ is isomorphic to $\mathsf{RecoverReachability}_{\odot}(\mathsf{Unravel}(n, G^{\circledast}, v_0))$.

Proof. The isomorphism maps a node $\pi_{\circledast}$ in $\mathsf{RecoverReachability}_{\odot}(\mathsf{Unravel}(n, G^{\circledast}, v_0))$ to its *deannotation* $\pi_{\oplus}$, obtained from $\pi_{\circledast}$ by replacing the label of each traversed edge by its deannotation. It is straightforward to verify that the isomorphism preserves node labels and short edges, so let us focus on long edges. Let $T_{\circledast} = \mathsf{Unravel}(n, G^{\circledast}, v_0)$.

For two nodes $\pi_{\circledast}, \pi'_{\circledast}$ in $T_{\circledast}$, the node $\sigma_{\circledast} = \mathsf{LCA}_{T_{\circledast}}(\pi_{\circledast}, \pi'_{\circledast})$ is the longest common prefix of $\pi_{\circledast}$ and $\pi'_{\circledast}$. Let $\pi_{\oplus}, \pi'_{\oplus}, \sigma_{\oplus}$ be the deannotations of $\pi_{\circledast}, \pi'_{\circledast}, \sigma_{\circledast}$, respectively. It is not difficult to see that $\sigma_{\oplus}$ is the longest common prefix of $\pi_{\oplus}$ and $\pi'_{\oplus}$. There is a long edge of rank r from $\pi_{\circledast}$ to $\pi'_{\circledast}$ in $\mathsf{RecoverReachability}_{\odot}(T_{\circledast})$ iff both:

- the walk from $\sigma_{\circledast}$ to $\pi_{\circledast}$ in $T_{\circledast}$ traverses only edges with labels of the form (a_+, s, r') with $s \geq r$,
- the walk from $\sigma_{\circledast}$ to $\pi'_{\circledast}$ in $T_{\circledast}$ is an $r^{\uparrow}$-walk.

We claim that these two conditions are equivalent to $\pi'_{\oplus}$ being an $r^{\uparrow}$-pseudoextension of $\pi_{\oplus}$ in $G^{\oplus}$. The following conditions are equivalent:

- the walk from $\sigma_{\circledast}$ to $\pi'_{\circledast}$ in $T_{\circledast}$ is an $r^{\uparrow}$-walk;
- $\pi'_{\oplus}$ is an $r^{\uparrow}$-extension of $\sigma_{\oplus}$.

The following conditions are also equivalent:

- the walk from $\sigma_{\circledast}$ to $\pi_{\circledast}$ in $T_{\circledast}$ traverses only edges with labels of the form (a_+, s, r') with $s \geq r$;
- the suffix of $\pi_{\oplus}$ obtained by removing the prefix $\sigma_{\oplus}$ is contained in one $r^{\uparrow}$-SCC in $G^{\oplus}$.

□

Corollary 8.21. For every ranked graph G and every non-negative integer n, $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$ is isomorphic to $\mathsf{RecoverReachability}_{\odot}(\mathsf{GlobalUnravel}(n, G^{\circledast}))$.

Claim 8.22. For two doubly-ranked short-long trees $T_{\circledast}, T'_{\circledast}$, if there is a ranked weak homomorphism from $T_{\circledast}$ to $T'_{\circledast}$, then $\mathsf{RecoverReachability}_{\odot}(T_{\circledast}) \to \mathsf{RecoverReachability}_{\odot}(T'_{\circledast})$.

Proof. Let $T_{\oplus} = \mathsf{RecoverReachability}_{\odot}(T_{\circledast})$ and let $T'_{\oplus} = \mathsf{RecoverReachability}_{\odot}(T'_{\circledast})$. Let h be a ranked weak homomorphism from $T_{\circledast}$ to $T'_{\circledast}$; recall that a ranked weak homomorphism is, intuitively, a homomorphism that is additionally allowed to map an edge with label (a, s, r) to an edge with label (a, s', r) for $s' \geq s$. We claim that h is a homomorphism from $T_{\oplus}$ to $T'_{\oplus}$. The fact that h preserves node labels and short edges follows directly from the definition of $\mathsf{RecoverReachability}_{\odot}$, so we focus on long edges.

Consider a long edge of rank r from u to v in $T_{\oplus}$, and let $w = \mathsf{LCA}_{T_{\circledast}}(u, v)$. We need to show that there is a long edge from $h(u)$ to $h(v)$ in $T'_{\oplus}$; recall that $\mathsf{LCA}_{T'_{\circledast}}(h(u), h(v))$ is a descendant of $h(w)$.

The walk from w to u in $T_\circledast$ traverses only edges with labels of the form (a_+, s, r') with $s \geq r$. This walk is mapped by h to a walk in $T'_\circledast$ that also traverses only edges with labels of the form (a_+, s, r') with $s \geq r$, by the definition of a ranked weak homomorphism. The walk from $\mathsf{LCA}_{T'_\circledast}(h(u), h(v))$ to $h(u)$ in $T'_\circledast$ is a suffix of this walk, so it also traverses only edges with labels of the form (a_+, s, r') with $s \geq r$.

The walk from w to v in $T_\circledast$ is an $r^\uparrow$-walk, so it is mapped by h to an $r^\uparrow$-walk in $T'_\circledast$. The walk from $\mathsf{LCA}_{T'_\circledast}(h(u), h(v))$ to $h(v)$ in $T'_\circledast$ is a suffix of this walk, so it is also an $r^\uparrow$-walk. □

Corollary 8.23. For two doubly-ranked short-long forests $G_\circledast, H_\circledast$, if there is a ranked weak homomorphism from $G_\circledast$ to $H_\circledast$, then $\mathsf{RecoverReachability}_\odot(G_\circledast) \to \mathsf{RecoverReachability}_\odot(H_\circledast)$.

Claim 8.24. For each doubly-ranked short-long modal profile $\mathcal{S}_\circledast$ of depth n and each ranked graph G such that $G^\circledast \models \mathcal{S}_\circledast$, $\mathsf{GlobalUnravel}_\oplus(n, G^\oplus) \leftrightarrow \mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)$.

Proof. By Fact 6.4 on page 66, $\mathsf{GlobalUnravel}(n, G_\circledast) \leftrightarrow \mathsf{GlobalRepresentative}(\mathcal{S}_\circledast)$. By Corollary 8.21, $\mathsf{GlobalUnravel}_\oplus(n, G^\oplus)$ is isomorphic to $\mathsf{RecoverReachability}_\odot(\mathsf{GlobalUnravel}(n, G^\circledast))$. By definition $\mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)$ is $\mathsf{RecoverReachability}_\odot(\mathsf{GlobalRepresentative}(\mathcal{S}_\circledast))$. Since a homomorphism between doubly-ranked graphs is also a ranked weak homomorphism, by Corollary 8.23 we get the required homomorphism equivalence. □

8.2.3 Correctness of the reduction: the first steps

Recall the reduction. For a given ranked modal profile $\mathcal{S}_\oplus = (n, \mathcal{T}_\oplus)$, we compute the set $\mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$ of $\mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast))$ for all doubly-ranked annotations $\mathcal{S}_\circledast$ of $\mathcal{S}_\oplus$ that are satisfiable within $\mathbb{G}_\circledast$. The result is $\mathsf{MinComputedDownsets}_\oplus(\mathcal{S}_\oplus)$: the set of minimal downsets in $\mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$.

We claim that $\mathsf{MinComputedDownsets}_\oplus(\mathcal{S}_\oplus) = \mathsf{MinDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$. To prove it, we show that:

- for each $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$, there is $\mathcal{D}' \in \mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$ such that $\mathcal{D}' \subseteq \mathcal{D}$;
- for each $\mathcal{D} \in \mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$, there is $\mathcal{D}' \in \mathsf{AllDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

Because both $\mathsf{AllDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$ and $\mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$ are finite sets, it follows that $\mathsf{MinComputedDownsets}_\oplus(\mathcal{S}_\oplus) = \mathsf{MinDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$.

The first claim follows from Claims 8.19 and 8.24, as follows. Since $\mathcal{D} \in \mathsf{AllDownsets}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$, there exists a ranked graph G such that $G^\oplus \models \mathcal{S}_\oplus$ and $\mathsf{Downset}_n(G^\oplus) = \mathcal{D}$. Let $\mathcal{S}_\circledast$ be the doubly-ranked annotation of $\mathcal{S}_\oplus$ such that $G^\circledast \models \mathcal{S}_\circledast$; it exists by Fact 8.10 on page 122. By Claims 8.19 and 8.24, $\mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast) \to G^\oplus$, so indeed $\mathcal{D}' \subseteq \mathcal{D}$ for the downset $\mathcal{D}' = \mathsf{Downset}_n(\mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)) \in \mathsf{AllComputedDownsets}_\oplus(\mathcal{S}_\oplus)$.

The crux of the proof of the correctness of the reduction is the second claim, stated below with expanded definitions, and proved in the next section.

Proposition 8.25. For every ranked modal profile $\mathcal{S}_\oplus = (n, \mathcal{T}_\oplus)$, for each doubly-ranked annotation $\mathcal{S}_\circledast$ of $\mathcal{S}_\oplus$ that is satisfiable within $\mathbb{G}_\circledast$ there exists a graph $G_\oplus \in \mathsf{Models}_{\mathbb{G}_\oplus}(\mathcal{S}_\oplus)$ such that each subgraph of $G_\oplus$ with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)$.

Assuming that the satisfiability problem for $\mathbb{G}_{\circledast}$ is decidable, the computability of the minimal downsets problem for $\mathbb{G}_{\oplus}$ follows, as constructing $\mathsf{GlobalRepresentative}_{\circledast \to \oplus}(\mathcal{S}_{\circledast})$ is easily computable, and the set of all doubly-ranked annotations of $\mathcal{S}_{\oplus}$ is computable (Fact 8.9 on page 122).

8.3 Correctness of the reduction: the crux

In this section we prove Proposition 8.25. The structure of the section is exactly the same as in the previous chapter: first, we prove additional technical claims about homomorphisms between unravellings, then we introduce the ranked analogue of the Last Edge Appearance Record, construct the required graph using ravelling constructions, and prove the existence of appropriate homomorphisms.

8.3.1 Homomorphisms between unravellings

Claim 8.26. For ranked graphs G, H, if $G \to H$, then $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$ maps homomorphically to $\mathsf{GlobalUnravel}_{\oplus}(n, H^{\oplus})$ for every non-negative integer n.

Proof. By Fact 8.12 on page 123, there is a ranked weak homomorphism from $G^{\circledast}$ to $H^{\circledast}$; it can be lifted to a ranked weak homomorphism from $\mathsf{GlobalUnravel}(n, G^{\circledast})$ to $\mathsf{GlobalUnravel}(n, H^{\circledast})$. By Corollaries 8.21 and 8.23, $\mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus}) \to \mathsf{GlobalUnravel}_{\oplus}(n, H^{\oplus})$. □

Claim 8.27. For a ranked graph G, positive integers n, N, and a ranked short-long graph $H_{\oplus}$ with at most n edges, if $H_{\oplus} \to \mathsf{GlobalUnravel}_{\oplus}(N, G^{\oplus})$, then $H_{\oplus} \to \mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$.

Proof. Assume $n < N$, as otherwise $\mathsf{GlobalUnravel}_{\oplus}(N, G^{\oplus}) \to \mathsf{GlobalUnravel}_{\oplus}(n, G^{\oplus})$, which makes the claim trivial. Let us assume that $H_{\oplus}$ is connected, has at most $n+1$ nodes, and that $H_{\oplus} \to \mathsf{Unravel}_{\oplus}(N, G^{\oplus}, v_0)$ for some node v_0. We can assume this, because we can consider each (weakly) connected component in $H_{\oplus}$ separately: since $H_{\oplus}$ has at most n edges, each of its connected components has at most $n+1$ nodes, and if a connected graph maps homomorphically to $\mathsf{GlobalUnravel}_{\oplus}(N, G^{\oplus})$, then it maps homomorphically to $\mathsf{Unravel}_{\oplus}(N, G^{\oplus}, v_0)$ for some node v_0.

We show that $H_{\oplus} \to \mathsf{Unravel}_{\oplus}(n, G^{\oplus}, v_0')$ for some node v_0'. We use the fact that $\mathsf{Unravel}_{\oplus}(k, G^{\oplus}, v)$ is isomorphic to $\mathsf{RecoverReachability}_{\odot}(\mathsf{Unravel}(k, G^{\circledast}, v))$ for every node v in G and every non-negative integer k (Claim 8.20 on page 128).

Let $T_{\circledast} = \mathsf{Unravel}(N, G^{\circledast}, v_0)$, and let h be a homomorphism from $H_{\oplus}$ to $\mathsf{RecoverReachability}_{\odot}(T_{\circledast})$. As in the previous chapter, we construct a homomorphism h' from $H_{\oplus}$ to $\mathsf{RecoverReachability}_{\odot}(T'_{\circledast})$ for some n-step unravelling $T'_{\circledast}$ of $G^{\circledast}$, intuitively, by considering only nodes in the image of h and their lowest common ancestors in $T_{\circledast}$, and by contracting walks between these nodes to long edges; the rank of this long edge is equal to the rank of the walk.

Let V be the set of all nodes in the image of h and all their lowest common ancestors in $T_{\circledast}$. The *V-root* is the shortest walk in V. The *V-parent* of a walk $\pi_{\circledast}$ in V is the longest proper prefix of $\pi_{\circledast}$ in V; the V-root has no V-parent. A *V-ancestor* of a walk $\pi_{\circledast}$ in V is any prefix of $\pi_{\circledast}$ in V. Recall that the number of V-ancestors of a walk $\pi_{\circledast}$ in V is at most $n+1$, because the size of the image of h is at most $n+1$ and because of the properties of the lowest common ancestors.

Let $D_\oplus$ be the subgraph of $\mathsf{RecoverReachability}_\odot(T_\circledast)$ induced by V. Since the image of h is a subset of V, $H_\oplus$ maps homomorphically to $D_\oplus$. Let $v'_0 = \mathsf{LastNode}(\pi_\circledast)$ for the V-root $\pi_\circledast$, and let $T'_\circledast = \mathsf{Unravel}(n, G^\circledast, v'_0)$. We define a homomorphism h' from $D_\oplus$ to $\mathsf{RecoverReachability}_\odot(T'_\circledast)$ recursively, as follows. For the V-root $\pi_\circledast$, let $h'(\pi_\circledast)$ be the trivial walk $[v'_0]$. For every other walk $\pi_\circledast$ in V, h' is defined recursively based on the V-parent $\pi'_\circledast$ of $\pi_\circledast$, with the following two cases.

- If $\pi_\circledast$ is an extension of $\pi'_\circledast$ by one short edge $e_\circledast$, then $h'(\pi_\circledast) = h'(\pi'_\circledast) \cdot e_\circledast$.
- Otherwise, $h'(\pi_\circledast)$ is the extension of $h'(\pi'_\circledast)$ by the long edge $e_\circledast$ from $\mathsf{LastNode}(\pi'_\circledast)$ to $\mathsf{LastNode}(\pi_\circledast)$ in $G^\circledast$ of the rank r equal to the rank of the suffix of $\pi_\circledast$ of length $|\pi_\circledast| - |\pi'_\circledast|$. Note that this is also the rank of the walk from $\pi'_\circledast$ to $\pi_\circledast$ in $T_\circledast$. There is exactly one such edge in $G^\circledast$: the label of $e_\circledast$ is $((\mathsf{Long}, \bot), s, r)$ for the weak rank s equal to the minimum value of s' in the labels (a_+, s', r') of edges traversed by the walk in $T_\circledast$ from $\pi'_\circledast$ to $\pi_\circledast$, by Claim 8.15 on page 124.

It remains to prove that h' is indeed a homomorphism from $D_\oplus$ to $\mathsf{RecoverReachability}_\odot(T'_\circledast)$. For each node $\pi_\circledast$ in $D_\oplus$, by the recursive definition of h', $h'(\pi_\circledast)$ is a walk in $G^\circledast$ beginning in v'_0, and it has length at most n, because $\pi_\circledast$ has at most $n+1$ V-ancestors, including itself. Moreover, $\mathsf{LastNode}(\pi_\circledast) = \mathsf{LastNode}(h'(\pi_\circledast))$, directly from the definition of h', so h' preserves node labels.

For each short edge $(\pi'_\circledast, ((\mathsf{Short}, a), r), \pi_\circledast)$ in $D_\oplus$, by the definition of $\mathsf{RecoverReachability}_\odot$, there is a short edge $(\pi'_\circledast, ((\mathsf{Short}, a), s, r), \pi_\circledast)$ in $T_\circledast$ for some weak rank s, so $\pi_\circledast$ is an extension of $\pi'_\circledast$ by one short edge $e_\circledast$ with the label $((\mathsf{Short}, a), s, r)$. Since both $\pi_\circledast$ and $\pi'_\circledast$ are in $D_\oplus$, they are also both in V, so $\pi'_\circledast$ is the V-parent of $\pi_\circledast$, because $\pi'_\circledast$ is the longest proper prefix of $\pi_\circledast$. Thus, by the definition of h', $h'(\pi_\circledast)$ is the extension of $h'(\pi'_\circledast)$ by $e_\circledast$, so there is an edge in $T'_\circledast$ from $h'(\pi'_\circledast)$ to $h'(\pi_\circledast)$ with the label $((\mathsf{Short}, a), s, r)$, so in $\mathsf{RecoverReachability}_\odot(T'_\circledast)$ there is a short edge from $h'(\pi'_\circledast)$ to $h'(\pi_\circledast)$ with the label $((\mathsf{Short}, a), r)$.

Finally, consider a long edge of rank r from $\pi_\circledast$ to $\pi'_\circledast$ in $D_\oplus$. This means that the walk in $T_\circledast$ from $\sigma_\circledast = \mathsf{LCA}_{T_\circledast}(\pi_\circledast, \pi'_\circledast)$ to $\pi_\circledast$ traverses only edges with labels of the form (a_+, s, r') with $s \geq r$, and the walk in $T_\circledast$ from $\sigma_\circledast$ to $\pi'_\circledast$ is an $r^\uparrow$-walk. Then, $\sigma_\circledast$ is in the domain of h', because V contains the lowest common ancestors in $T_\circledast$ of all pairs of nodes in the image of h. By the definition of h', the walk in $T'_\circledast$ from $h'(\sigma_\circledast)$ to $h'(\pi_\circledast)$ also traverses only edges with labels of the form (a_+, s, r') with $s \geq r$, and the walk in $T'_\circledast$ from $h'(\sigma_\circledast)$ to $h'(\pi'_\circledast)$ is also an $r^\uparrow$-walk. The same conditions hold for the suffixes of these walks; in particular, for $\sigma'_\circledast = \mathsf{LCA}_{T'_\circledast}(h'(\pi_\circledast), h'(\pi'_\circledast))$, the walk from $\sigma'_\circledast$ to $h'(\pi_\circledast)$ traverses only edges with labels of the form (a_+, s, r') with $s \geq r$, and the walk from $\sigma'_\circledast$ to $h'(\pi'_\circledast)$ is an $r^\uparrow$-walk. Thus, there is a long edge of rank r from $h'(\pi_\circledast)$ to $h'(\pi'_\circledast)$ in $\mathsf{RecoverReachability}_\odot(T'_\circledast)$. □

8.3.2 Last Edge Appearance Record

Consider a walk π in a ranked graph G. Let $\pi_\oplus$ be the corresponding walk in $G^\oplus$, obtained from π by replacing each edge $(u, (a, r), v)$ with $(u, ((\mathsf{Short}, a), r), v)$. Let $\pi_\oplus = [v_0, e_1, v_1, \ldots, e_m, v_m]$. As in the previous chapter, an index $i \in \{1, \ldots, m\}$ is the *last appearance* of the edge e_i if $e_j \neq e_i$ for all $j > i$. An infix of $\pi_\oplus$ is *avoiding last appearances* if it does not contain the last appearance of any edge. The *ranked Last Edge Appearance Record* of π, denoted $\mathsf{LEAR}_\odot(\pi)$, is the walk in $G^\oplus$ obtained from $\pi_\oplus$ by replacing each maximal non-trivial infix that avoids last appearances $[v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$, informally, with a single long edge of the rank equal to the rank of this infix; formally, with $[v_i, (v_i, ((\mathsf{Long}, \bot), r), v_j)]$, where r is the rank of this infix: the minimal rank of edges

$e_{i+1}, \ldots, e_j$. An example of a walk and its ranked LEAR is presented in Figure 8.5.

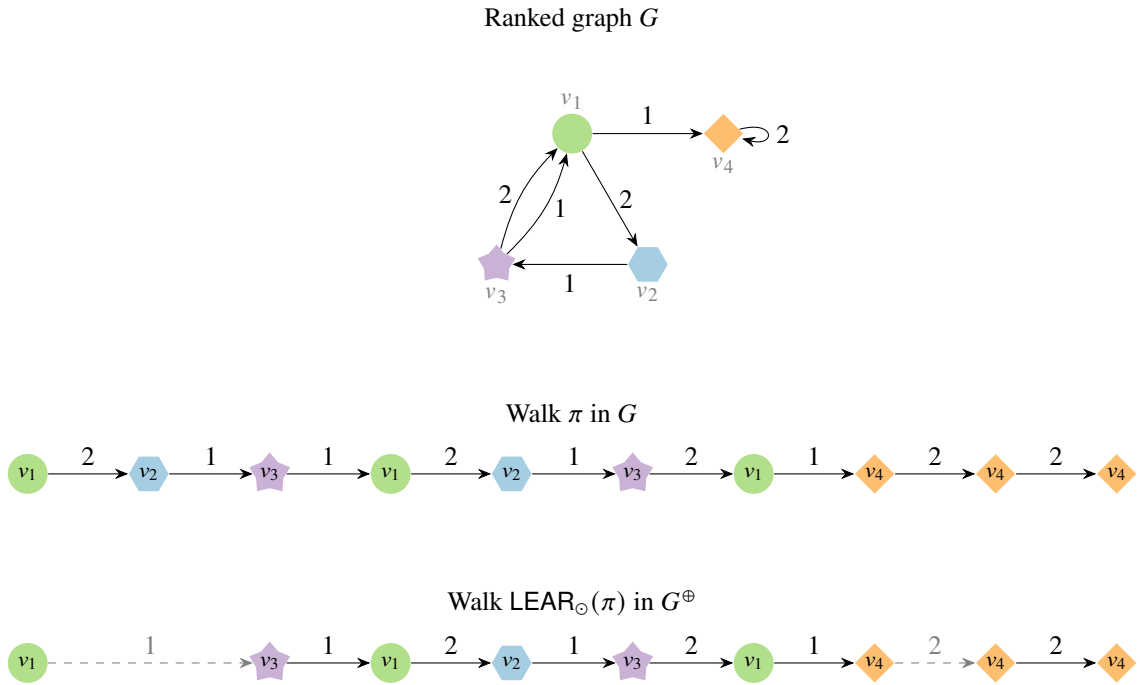


Figure 8.5: An example of a walk π in a ranked graph G and its ranked Last Edge Appearance Record $\mathsf{LEAR}_{\odot}(\pi)$. As usual, we write only ranks of edges instead of their full labels.

8.3.3 Constructing a witness

We want to prove Proposition 8.25: for every ranked modal profile $\mathcal{S}_{\oplus} = (n, \mathcal{T}_{\oplus})$, for each doubly-ranked annotation $\mathcal{S}_{\circledast}$ of $\mathcal{S}_{\oplus}$ that is satisfiable within $\mathbb{G}_{\circledast}$ there exists a graph $G_{\oplus} \in \mathsf{Models}_{\mathbb{G}_{\oplus}}(\mathcal{S}_{\oplus})$ such that each subgraph of $G_{\oplus}$ with at most n edges maps homomorphically to $\mathsf{GlobalRepresentative}_{\circledast \to \oplus}(\mathcal{S}_{\circledast})$.

Consider a ranked modal profile $\mathcal{S}_{\oplus}$ of depth n, and its doubly-ranked annotation $\mathcal{S}_{\circledast}$ that is satisfiable within $\mathbb{G}_{\circledast}$. Let G_0 be a ranked graph such that $(G_0)^{\circledast} \models \mathcal{S}_{\circledast}$, and let $R = \mathsf{MaxRank}_{G_0}$. Note that $(G_0)^{\oplus} \models \mathcal{S}_{\oplus}$, by Fact 8.10 on page 122.

Let $G_1 = \mathsf{SafeRavel}_f(G_0)$ for f mapping a walk π to the sequence $(\ell_1, \ldots, \ell_R)$, where ℓ_r is the number of edges of rank r traversed by π modulo $n+1$.

Claim 8.28. Each cycle of rank r in G_1 traverses at least $n+1$ edges of rank r.

Proof. Let $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$ be a cycle of rank r in G_1. By the definitions of $\mathsf{SafeRavel}$ and f, for each edge e_i of rank s, the nodes v_{i-1} and v_i are of the following form:

- $v_{i-1} = ((\ell_1, \ldots, \ell_R), u)$ for some node u in G_0;
- $v_i = ((\ell'_1, \ldots, \ell'_R), v)$ for some node v in G_0;

such that $\ell'_s = (\ell_s + 1) \bmod (n+1)$, and for each rank $s' \neq s$, $\ell'_{s'} = \ell_{s'}$. Thus, because π is a cycle, for each rank s, the number of edges of rank s traversed by π is divisible by $n+1$. Because π has rank r, it traverses at least one edge of rank r, so it traverses at least $n+1$ edges of rank r. □

Corollary 8.29. Each non-trivial closed walk of rank r in G_1 traverses at least $n+1$ different edges of rank r.

Let $G_2 = \mathsf{Ravel}_{\mathsf{LEAR}_\circ}(G_1)$.

Because $(G_0)^\oplus \models \mathcal{S}_\oplus$, we have $(G_1)^\oplus \models \mathcal{S}_\oplus$ and $(G_2)^\oplus \models \mathcal{S}_\oplus$ (by Fact 8.8 on page 121).

To prove Proposition 8.25 and finish the proof of the correctness of the reduction, we need to show that, for each subgraph $D_\oplus$ of $(G_2)^\oplus$ with at most n edges, $D_\oplus \to \mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)$. More precisely, we prove that $D_\oplus \to \mathsf{GlobalUnravel}_\oplus(n, (G_1)^\oplus)$; the rest follows from the fact that $(G_0)^\circledast \models \mathcal{S}_\circledast$, and from earlier claims:

- by Claim 8.26 on page 130, $\mathsf{GlobalUnravel}_\oplus(n, (G_1)^\oplus) \to \mathsf{GlobalUnravel}_\oplus(n, (G_0)^\oplus)$;
- by Claim 8.24 on page 129, $\mathsf{GlobalUnravel}_\oplus(n, (G_0)^\oplus) \leftrightarrow \mathsf{GlobalRepresentative}_{\circledast\to\oplus}(\mathcal{S}_\circledast)$.

Thus, it remains to prove the following proposition.

Proposition 8.30. Each subgraph $D_\oplus$ of $(G_2)^\oplus$ with at most n edges maps homomorphically to $\mathsf{GlobalUnravel}_\oplus(n, (G_1)^\oplus)$.

As in the previous chapters, we prove the proposition in two steps. We define a homomorphism h from $D_\oplus$ to $\mathsf{GlobalUnravel}_\oplus(N, (G_1)^\oplus)$ for some positive integer N; by Claim 8.27 on page 130, this implies that $D_\oplus \to \mathsf{GlobalUnravel}_\oplus(n, (G_1)^\oplus)$. We first construct the homomorphism h, and then we prove that it is indeed a homomorphism.

Constructing the homomorphism $h : D_\oplus \to \mathsf{GlobalUnravel}_\oplus(N, (G_1)^\oplus)$

Let $h_{G_2\to G_1}$ be the ravelling homomorphism from G_2 to G_1; it is also a homomorphism from $(G_2)^\oplus$ to $(G_1)^\oplus$ (by Fact 8.7 on page 121). A short edge in $(G_1)^\oplus$ is *important* if it is in $h_{G_2\to G_1}(D_\oplus)$; recall that we denote by $h_{G_2\to G_1}(D_\oplus)$ the graph-image of $D_\oplus$ under $h_{G_2\to G_1}$. Note that long edges are not important.

Each node in G_2 is a walk in $(G_1)^\oplus$. Consider a walk $\pi_\oplus$ in $(G_1)^\oplus$ traversing edges $(e_1, \ldots, e_m)$. An index i is *important* if e_i is important. An index i is *dominated* by an index j if $j > i$ and e_i, e_j are in one $r^\uparrow$-SCC in $(G_1)^\oplus$, where r is the rank of e_j; note that we do not require the whole infix $e_i, e_{i+1}, \ldots, e_j$ to be in the same $r^\uparrow$-SCC – adding such a restriction might be more intuitive, but omitting it makes the proofs much simpler, and we will see that both definitions would ultimately result in the same homomorphism. An index i is *unimportantly dominated* if it is dominated by some index that is not important. An infix of $\pi_\oplus$ is *unimportantly dominated* if it contains only unimportantly dominated indices.

Let h map $\pi_\oplus$ to the walk in $(G_1)^\oplus$ obtained from $\pi_\oplus$ by replacing each maximal non-trivial infix that is unimportantly dominated $[v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ with a single long edge; formally, with $[v_i, (v_i, ((\mathsf{Long}, \bot), r), v_j), v_j]$, where r is the rank of the infix: the minimal rank of edges $e_{i+1}, \ldots, e_j$. If the resulting walk contains some consecutive long edges, we additionally replace each maximal non-trivial infix traversing only long edges with a long edge of rank equal to the rank of this infix.

For a walk π in G_1, let $h(\pi)$ be defined as $h(\pi_\oplus)$, where $\pi_\oplus$ is the walk in $(G_1)^\oplus$ corresponding to π,

obtained from π by replacing the label (a, r) of each traversed edge with $((\mathsf{Short}, a), r)$.

Proving that h is a homomorphism

We claim that h restricted to $V_{D_\oplus}$ is a homomorphism from $D_\oplus$ to $\mathsf{GlobalUnravel}_\oplus(N, (G_1)^\oplus)$ for some positive integer N. Clearly, for each node $\pi_\oplus$ in $D_\oplus$, $h(\pi_\oplus)$ is a walk in $(G_1)^\oplus$, so the claim holds for a large enough N. As in the previous chapter, the proof relies on two claims: the first one allows us to focus on walks in $G^\oplus$ that traverse only short edges, and the second one states the crucial guarantees of h when applied to some walk and its extension by a single short edge. Then, it suffices to show how these guarantees indeed imply that h is a homomorphism, which, as in the previous chapter, requires looking at the ravelling homomorphism $h_{G_2 \to G_1}$ and ravelling witnesses.

Claim 8.31. For each walk π in G_1, $h(\pi) = h(\mathsf{LEAR}_\odot(\pi))$.

Proof. Let $\pi_\oplus$ be the walk in $(G_1)^\oplus$ corresponding to π. The walk $h(\pi)$ is obtained from $\pi_\oplus$ by replacing each maximal non-trivial unimportantly dominated infix with a long edge. The walk $\mathsf{LEAR}_\odot(\pi)$ is obtained from $\pi_\oplus$ by replacing each maximal non-trivial infix avoiding last appearances with a long edge. The walk $h(\mathsf{LEAR}_\odot(\pi))$ is obtained from $\mathsf{LEAR}_\odot(\pi)$ by replacing each maximal non-trivial unimportantly dominated infix with a long edge. In all three cases, the rank of the long edge is equal to the rank of the replaced infix, and the resulting walks have no two consecutive long edges.

Let $(e_1, \ldots, e_m)$ be the sequence of edges traversed by $\pi_\oplus$. The walks $h(\pi)$ and $h(\mathsf{LEAR}_\odot(\pi))$, by the above properties, can be uniquely characterized by the set of indices of edges e_i that are not replaced by long edges in the constructions h and $\mathsf{LEAR}_\odot$. We need to show that in both cases this set of indices is the same. For this, we need to prove that:

- each non-last appearance of an edge in $\pi_\oplus$ is unimportantly dominated in $\pi_\oplus$;
- each last appearance of an edge that is unimportantly dominated in $\pi_\oplus$ is also unimportantly dominated in $\mathsf{LEAR}_\odot(\pi_\oplus)$.

The second point is easy to prove: for an index i that is dominated in $\pi_\oplus$ by some unimportant index j, the index i is also dominated by the last appearance of e_j in $\pi_\oplus$, which is also unimportant. For the first point, consider a non-last appearance i of some edge e_i in $\pi_\oplus$. Let e_j be the last appearance of e_i in $\pi_\oplus$. Consider the infix $\sigma_\oplus = [v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ of $\pi_\oplus$, and let r be its rank. The infix $\sigma_\oplus$ is a closed walk in $(G_1)^\oplus$ that consists only of short edges. By Corollary 8.29 on the preceding page, each closed walk of rank r in G_1 traverses at least $n + 1$ different edges of rank r, and there are at most n important edges in $(G_1)^\oplus$, so $\sigma_\oplus$ traverses some unimportant edge of rank r. Moreover, since $\sigma_\oplus$ is a closed walk of rank r, it is contained in one $r^\uparrow$-SCC in $(G_1)^\oplus$, and it includes the edge $e_j = e_i$. Thus, i is dominated by some unimportant index. □

Claim 8.32. For two walks π, π' in G_1 such that $\pi' = \pi \cdot e$ for some edge e of rank r, and for their corresponding walks $\pi_\oplus, \pi'_\oplus$ and edge $e_\oplus$ in $(G_1)^\oplus$:

- if $e_\oplus$ is important, then $h(\pi'_\oplus) = h(\pi_\oplus) \cdot e_\oplus$;
- otherwise, $h(\pi'_\oplus)$ is an $r^\uparrow$-pseudoextension of $h(\pi_\oplus)$ in $(G_1)^\oplus$.

Proof. Let $E_1 = (e_1, \ldots, e_m)$ be the sequence of edges traversed by $\pi_\oplus$. Then, the sequence of edges traversed by $\pi'_\oplus$ is $E_2 = (e_1, \ldots, e_m, e_{m+1})$, with $e_{m+1} = e_\oplus$. The last index $m + 1$ is not dominated by

any index in E_2.

If $e_\oplus$ is important, then $h(\pi'_\oplus) = h(\pi_\oplus) \cdot e_\oplus$, because each index $1 \leq i \leq m$ is unimportantly dominated in E_1 iff i is unimportantly dominated in E_2.

If $e_\oplus$ is not important, then the last index $m+1$ in E_2 is unimportant and dominates all indices i such that e_i is in the same $r^\uparrow$-SCC as $e_\oplus$ in $(G_1)^\oplus$. Let $\sigma'_\oplus$ be the longest suffix of $\pi'_\oplus$ contained in the $r^\uparrow$-SCC of $e_\oplus$ in $(G_1)^\oplus$. If the suffix $\sigma'_\oplus$ is equal to the whole walk $\pi'_\oplus$, then clearly $h(\pi'_\oplus)$ is an $r^\uparrow$-pseudoextension of $h(\pi_\oplus)$, as both are contained in one $r^\uparrow$-SCC in $(G_1)^\oplus$. Otherwise, contrary to the case in the previous chapter, there might be some indices dominated by the index $m+1$ in E_2 that are not in $\sigma'_\oplus$, because the same $r^\uparrow$-SCC might be visited multiple times by a walk in a ranked graph, as long as this walk traverses an edge of rank strictly smaller than r in between. However, all indices dominated by $m+1$ in E_2 that are not in $\sigma'_\oplus$ are already unimportantly dominated in E_1; before we prove it, let us finish the main reasoning. Let e_i be the last edge in E_2 that is not in $\sigma'_\oplus$. The index i is not dominated in E_2, nor in E_1, because it does not belong to any $r^\uparrow$-SCC in $(G_1)^\oplus$, and all edges on greater indices have ranks at least r. Thus, $h(\pi'_\oplus)$ is obtained from $h(\pi_\oplus)$ by removing the longest suffix contained in one $r^\uparrow$-SCC in $(G_1)^\oplus$ and appending some $r^\uparrow$-walk in $(G_1)^\oplus$, which is the definition of being an $r^\uparrow$-pseudoextension in $(G_1)^\oplus$.

It remains to prove that all indices dominated by $m+1$ in E_2 that are not in $\sigma'_\oplus$ are already unimportantly dominated in E_1. Consider an index i that is dominated by $m+1$ in E_2 and is not in $\sigma'_\oplus$; then, e_i is in the $r^\uparrow$-SCC of $e_\oplus$ in $(G_1)^\oplus$. Consider the suffix $\sigma_\oplus = [v_{i-1}, e_i, v_i, \ldots, e_m, v_m]$ of $\pi_\oplus$, and let s be its rank; note that $s < r$. Because e_{m+1} and e_i are in one $r^\uparrow$-SCC in $(G_1)^\oplus$, and by the Ranked SCC Invariance Under Annotations, there is an $r^\uparrow$-walk from v_m to v_{i-1} in G_1; extending $\sigma_\oplus$ with the corresponding walk in $(G_1)^\oplus$ results in a closed walk of rank s that traverses only short edges. By Corollary 8.29 on page 133, each non-trivial closed walk of rank s in G_1 traverses at least $n+1$ different edges of rank s, and there are at most n important edges in $(G_1)^\oplus$, so the closed walk defined above traverses some unimportant edge of rank s; because $s < r$, this unimportant edge of rank s must be in $\sigma_\oplus$. Thus, i is dominated by some unimportant edge in E_1. □

To prove that h is a homomorphism, we need to show that h preserves node labels, short edges and long edges. Recall that $\mathsf{GlobalUnravel}_\oplus(N, (G_1)^\oplus)$ is the ranked short-long graph with nodes being all walks of length at most N in $(G_1)^\oplus$, each having the label of its last node in $(G_1)^\oplus$, a short edge $(\pi_\oplus, a_\oplus, \pi'_\oplus)$ iff $\pi'_\oplus \in \mathsf{Extensions}_{(G_1)^\oplus}(\pi_\oplus, a_\oplus)$, and a long edge of rank r from $\pi_\oplus$ to $\pi'_\oplus$ iff $\pi'_\oplus$ is an $r^\uparrow$-pseudoextension of $\pi_\oplus$ in $(G_1)^\oplus$. Recall that $G_2 = \mathsf{Ravel}_{\mathsf{LEAR}_\odot}(G_1)$.

It is straightforward to verify that h preserves node labels, because the ravelling constructions and $\mathsf{GlobalUnravel}_\oplus$ inherit node labels from the last nodes of walks, and the functions $\mathsf{LEAR}_\odot$ and h preserve the last nodes of walks.

To show that h preserves short edges, we will prove that $h(\pi'_\oplus) = h(\pi_\oplus) \cdot e'_\oplus$ for each short edge $e_\oplus = (\pi_\oplus, ((\mathsf{Short}, a), r), \pi'_\oplus)$ in $D_\oplus$, where $e'_\oplus$ is the edge in $(G_1)^\oplus$ such that $e_\oplus$ is mapped by $h_{G_2 \to G_1}$ to $e'_\oplus$. Let (π, e', π') be a ravelling witness of $e_\oplus$; recall that $\pi_\oplus = \mathsf{LEAR}_\odot(\pi)$, $\pi'_\oplus = \mathsf{LEAR}_\odot(\pi')$, and $\pi' = \pi \cdot e'$. Note that $e'_\oplus$ is the edge in $(G_1)^\oplus$ corresponding to e' in G_1. By Claim 8.31, $h(\pi) = h(\pi_\oplus)$ and $h(\pi') = h(\pi'_\oplus)$. The fact that $h(\pi'_\oplus) = h(\pi_\oplus) \cdot e'_\oplus$ follows from Claim 8.32.

To show that h preserves long edges, we will prove that, for each rank r, for each $r^{\uparrow}$-walk in G_2 from a node $\pi_{\oplus}$ to a node $\pi'_{\oplus}$, $h(\pi'_{\oplus})$ is an $r^{\uparrow}$-pseudoextension of $h(\pi_{\oplus})$ in $(G_1)^{\oplus}$. Because being an $r^{\uparrow}$-pseudoextension is transitive, it suffices to prove that, for each short edge $e_{\oplus} = (\pi_{\oplus}, ((\mathsf{Short}, a), r), \pi'_{\oplus})$ in $(G_2)^{\oplus}$, $h(\pi'_{\oplus})$ is an $r^{\uparrow}$-pseudoextension of $h(\pi_{\oplus})$ in $(G_1)^{\oplus}$. Let (π, e', π') be a ravelling witness of $e_{\oplus}$. By Claim 8.31 on page 134, $h(\pi) = h(\pi_{\oplus})$ and $h(\pi') = h(\pi'_{\oplus})$. The fact that $h(\pi'_{\oplus})$ is an $r^{\uparrow}$-pseudoextension of $h(\pi_{\oplus})$ follows from Claim 8.32 on page 134, and from the fact that being an $r^{\uparrow}$-extension implies being an $r^{\uparrow}$-pseudoextension.

This finishes the proof of Proposition 8.30, Proposition 8.25, and of the correctness of the reduction from the minimal downsets problem for $\mathbb{G}_{\oplus}$ to the satisfiability problem for $\mathbb{G}_{\circledast}$.

8.3.4 Summary

We reduced the minimal downsets problem for $\mathbb{G}_{\oplus}$ to the satisfiability problem for $\mathbb{G}_{\circledast}$, using exactly the same techniques as in the previous chapter. This required defining ranked graphs and ranked SCCs in a way that allows generalizing the notions and techniques. The proofs are more involved technically than the ones in the previous chapter, but they use the same main ideas – the core work is done by choosing well-behaved definitions.

The situation is different in the next section, where we solve the satisfiability problem for $\mathbb{G}_{\circledast}$ – while the main ideas of the proof for $\mathbb{G}_{*}$ transfer to the ranked case, the presence of ranks requires a significantly more involved combinatorial construction.

8.4 Satisfiability

In this section we solve the satisfiability problem for $\mathbb{G}_{\circledast}$, which is the most general satisfiability result solved in this thesis. We prove the following proposition.

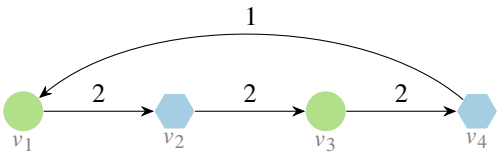
Proposition 8.33. For every ranked modal profile $\mathcal{S}_{\circledast} = (n, \mathcal{T}_{\circledast})$ satisfiable within $\mathbb{G}_{\circledast}$ there exists a graph $G_{\circledast} \in \mathsf{Models}_{\mathbb{G}_{\circledast}}(\mathcal{S}_{\circledast})$ with at most $(|\mathcal{T}_{\circledast}| + 1)^{3R}$ nodes, where R is the maximal rank of edge labels in $\mathcal{S}_{\circledast}$.

The decidability of the satisfiability problem for $\mathbb{G}_{\circledast}$ follows, as it is enough to iterate over all (isomorphism types of) graphs G with at most $(|\mathcal{T}_{\circledast}| + 1)^{3R}$ nodes, using only appropriate labels (node labels from $\mathcal{S}_{\circledast}$ and edge labels (a, r) such that $((\mathsf{Short}, a), s, r)$ appears as an edge label in $\mathcal{S}_{\circledast}$ for some s), and check if $G^{\circledast} \models \mathcal{S}_{\circledast}$. Note that the decidability of the constructive satisfiability problem for $\mathbb{G}_{\circledast}$ also follows; recall that it is a computational problem, where the output is some graph $G_{\circledast} \in \mathsf{Models}_{\mathbb{G}_{\circledast}}(\mathcal{S}_{\circledast})$, if it exists.

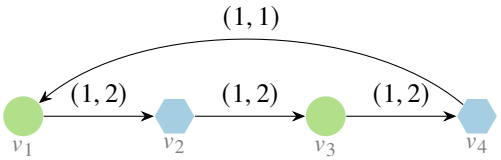
8.4.1 Comparing the approaches

We begin by considering an example, showing why the strategy from the previous chapter fails in this setting. Then, we discuss the differences in the approaches in the previous chapters and this one.

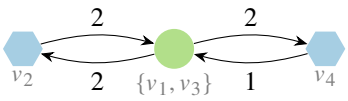
Graph G



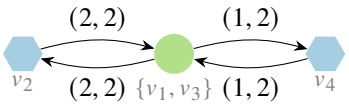
Ranked SCC-annotation $G^{\otimes}$



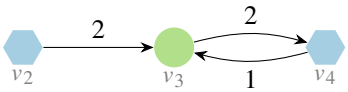
Graph G_1 obtained from G by identifying v_1 and v_3



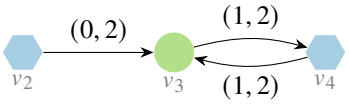
Ranked SCC-annotation $(G_1)^{\otimes}$



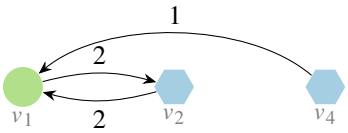
Graph G_2 obtained from G by removing v_1 and redirecting its incoming edges



Ranked SCC-annotation $(G_2)^{\otimes}$



Graph G_3 obtained from G by removing v_3 and redirecting its incoming edges



Ranked SCC-annotation $(G_3)^{\otimes}$

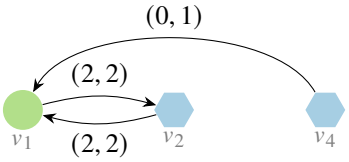


Figure 8.6: A ranked graph G and graphs obtained by applying strategies from previous chapters to it, along with their ranked SCC-annotation. As usual, we omit the component a in edge labels (a, r) and (a, s, r), and assume it is equal for all edges.

Consider the graph G presented in Figure 8.6. The strategies from previous chapters do not work for this graph, even when we are interested only in types of nodes in the ranked SCC-annotation $G^{\otimes}$ of G. Note that the whole graph G is a single $1^{\uparrow}$-SCC, each node is in a separate $2^{\uparrow}$-SCC, and v_1 and v_3 have equal 1-bisimilarity types in $G^{\otimes}$. Identifying these nodes would create a non-trivial $2^{\uparrow}$-SCC, while removing one of them and redirecting incoming edges to the other would result in some edge not being contained in any $1^{\uparrow}$-SCC. In all these cases, the ranked SCC-annotation of the resulting graph contains some edge label that is not present in $G^{\otimes}$, which results in a different modal profile.

In the descriptions below, we use the word *type* informally, for the n-bisimilarity type of a node in an appropriate graph, for an appropriate n.

In Chapter 6, to solve the satisfiability problem, we constructed a small model by considering an arbitrary model G and identifying all nodes with equal types. This operation can be alternatively described, up to isomorphism, as the following two-step process.

1. Add to G all edges (u, a, v) such that there is already an edge (u', a, v') in G such that u, u' have equal types and v, v' have equal types.
2. Take a subgraph of the resulting graph induced by a set containing one node of each type.

In Chapter 7 we needed to be more careful to preserve the SCC structure of the graph, and we argued that simply identifying nodes of equal types might significantly change this structure. However, the construction we used can be described as a similar two-step process, where we additionally take into account the SCC structure of G. Recall that we fixed an arbitrary topological order $\leq$ on SCCs in G.

1. Add to G all edges (u, a, v) such that there is already an edge (u', a, v') in G such that u, u' have equal types, v, v' have equal types, and either:
   - $\mathsf{SCC}_G(u) = \mathsf{SCC}_G(v)$ and $\mathsf{SCC}_G(u') = \mathsf{SCC}_G(v')$, or
   - $\mathsf{SCC}_G(u) < \mathsf{SCC}_G(v)$ and $\mathsf{SCC}_G(u') < \mathsf{SCC}_G(v')$.
2. Take a subgraph of the resulting graph induced by a set containing one node of each type from each SCC of interest: the greatest (in $\leq$) SCC containing a node of some type.

We managed to describe this process in one step, intuitively, because the set of nodes chosen in the second step is relatively easy to describe in terms of the original graph G.

In this chapter we need to actually separate these two steps, because to define the set of nodes in the second step, we use the properties of the intermediate graph with added edges. More precisely, we prove that the intermediate graph contains walks of bounded length between every two nodes in one ranked SCC, and we include such walks in the induced subgraph to preserve the structure of ranked SCCs.

Before we proceed with the proof, we define *universal topological orders* in ranked graphs.

8.4.2 Universal topological orders

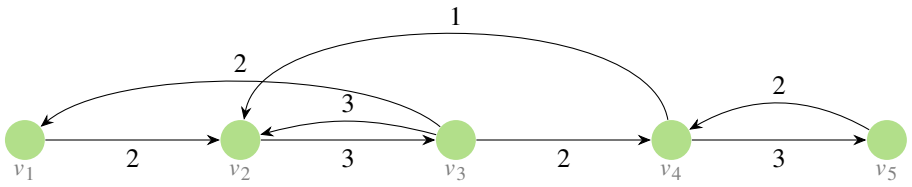
Consider a ranked graph G. We will be interested in a topological order on (unranked) SCCs in $G|_r$ for each rank r. Recall that a topological order on SCCs in $G|_r$ is a linear order $\leq$ such that, for each edge from a node u to a node v in $G|_r$, $\mathsf{SCC}_{G|_r}(u) \leq \mathsf{SCC}_{G|_r}(v)$. This can be equivalently written as $\mathsf{SCC}^r_G(u) \leq \mathsf{SCC}^r_G(v)$. To avoid dealing with multiple orders, we show that it is possible to construct

a single linear order on nodes in G that induces topological orders on SCCs in $G|_r$ for each rank r, similarly to the node-level topological order in the previous chapter.

A linear order $\leq$ on nodes in G is a *universal topological order* if, for each edge $(u, (a, r), v)$ in G, either $u \leq v$, or, for each node w such that $v \leq w \leq u$, $\mathsf{SCC}^r_G(v) = \mathsf{SCC}^r_G(w) = \mathsf{SCC}^r_G(u)$.

An example of a graph and a universal topological order on its nodes is presented in Figure 8.7.

Graph G with nodes drawn according to an example node-level topological order



Graph G with marked ranked SCCs

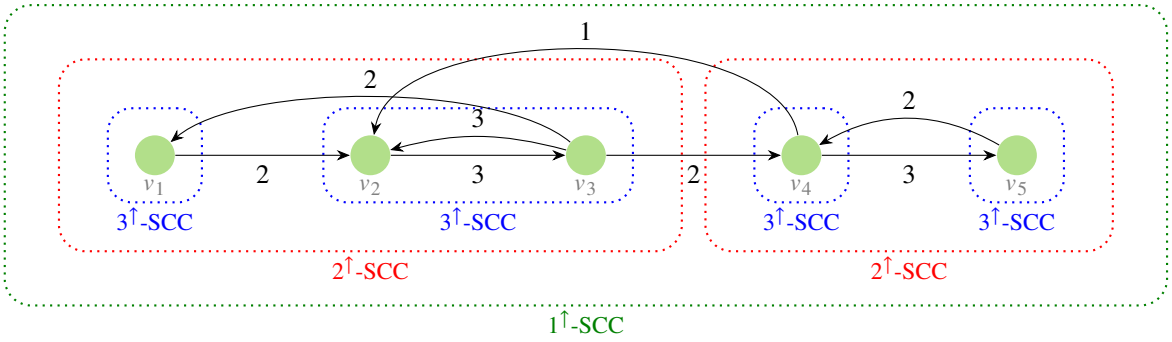


Figure 8.7: A ranked graph G, with nodes drawn according to an example universal topological order, and the same graph with marked ranked SCCs. We write only ranks of edges instead of their full labels. Note that, for each edge of rank r directed to the left on the picture, all nodes below it are in one $r^\uparrow$-SCC.

Claim 8.34. For each ranked graph G there exists a universal topological order.

Proof. We define such an order recursively, by defining a universal topological order $\leq_S$ in each $s^\uparrow$-SCC S in G, for each $s = \mathsf{MaxRank}_G, \ldots, 0$. For a $(\mathsf{MaxRank}_G)^\uparrow$-SCC S, let $\leq_S$ be an arbitrary linear order on nodes in S; it is a universal topological order in S, as all edges in S have rank $\mathsf{MaxRank}_G$, and $\mathsf{SCC}_S^{\mathsf{MaxRank}_G}(u) = \mathsf{SCC}_S^{\mathsf{MaxRank}_G}(v)$ for all nodes u, v in S. For $s < \mathsf{MaxRank}_G$ and an $s^\uparrow$-SCC S, $S|_{s+1}$ is the graph obtained from S by removing all edges of rank s, and its (unranked) SCCs are $(s+1)^\uparrow$-SCCs in G. Let $\leq_S$ be obtained by combining an arbitrary topological order $\preccurlyeq$ on SCCs in $S|_{s+1}$ and the universal topological order $\leq_{S'}$ in each SCC S' in $S|_{s+1}$, which is constructed recursively. To be precise, for two nodes u, v in S, $u <_S v$ iff either $\mathsf{SCC}_S^{s+1}(u) \prec \mathsf{SCC}_S^{s+1}(v)$, or u, v are in one SCC S' in $S|_{s+1}$ and $u <_{S'} v$. To prove that this is a universal topological order in S, we need to show that for all edges $(u, (a, r), v)$ in S such that $u >_S v$, for each node w such that $v \leq_S w \leq_S u$, $\mathsf{SCC}^r_S(v) = \mathsf{SCC}^r_S(w) = \mathsf{SCC}^r_S(u)$. For each edge $(u, (a, r), v)$ in S:

- if $r \geq s+1$ and u, v are in one SCC S' in $S|_{s+1}$, the condition holds by the properties of $\leq_{S'}$;
- if $r \geq s+1$ and u, v are in different SCCs in $S|_{s+1}$, then $\mathsf{SCC}^{s+1}_S(u) \prec \mathsf{SCC}^{s+1}_S(v)$, so $u <_S v$;
- if $r = s$, the condition holds because $\mathsf{SCC}^s_S(w) = S$ for all nodes w in S.

□

Note that a universal topological order $\leq$ in a ranked graph G induces a topological order on $r^\uparrow$-SCCs in G (or, more precisely, a topological order on unranked SCCs in $G|_r$) for each rank r: $S_1 < S_2$ iff, for every node v_1 in S_1 and every node v_2 in S_2, $v_1 < v_2$.

Fact 8.35. For a universal topological order $\leq$ and nodes u, v in a ranked graph G:

1. for each rank r, if $\mathsf{SCC}^r_G(u) < \mathsf{SCC}^r_G(v)$, then $u < v$;
2. for each rank r, if v is $r^\uparrow$-reachable from u, then $\mathsf{SCC}^r_G(u) \leq \mathsf{SCC}^r_G(v)$;
3. for ranks $r_1 \leq r_2$, if $\mathsf{SCC}^{r_1}_G(u) < \mathsf{SCC}^{r_1}_G(v)$, then $\mathsf{SCC}^{r_2}_G(u) < \mathsf{SCC}^{r_2}_G(v)$.

The contrapositives of the statements above are as follows:

1. if $u \geq v$, then $\mathsf{SCC}^r_G(u) \geq \mathsf{SCC}^r_G(v)$ for each rank r;
2. for each rank r, if $\mathsf{SCC}^r_G(u) > \mathsf{SCC}^r_G(v)$, then there is no $r^\uparrow$-walk from u to v in G;
3. for ranks $r_1 \leq r_2$, if $\mathsf{SCC}^{r_2}_G(u) \geq \mathsf{SCC}^{r_2}_G(v)$, then $\mathsf{SCC}^{r_1}_G(u) \geq \mathsf{SCC}^{r_1}_G(v)$.

8.4.3 Saturating a graph

Consider a modal profile $\mathcal{S}_\circledast$ that is satisfiable within $\mathbb{G}_\circledast$, and let G be a ranked graph such that $G^\circledast \models \mathcal{S}_\circledast$. Let $\leq$ be a universal topological order in G.

Adding edges while preserving ranked SCCs and the universal topological order

A triple $e = (u, (a, r), v)$ is a *potential edge* in G if u, v are nodes in G, r is a rank in G, and a is an edge label. Consider the graph G_e obtained from G by adding the edge e. We say that *e preserves ranked SCCs* if, for all nodes u, v and all ranks r, $\mathsf{SCC}^r_G(u) = \mathsf{SCC}^r_G(v)$ iff $\mathsf{SCC}^r_{G_e}(u) = \mathsf{SCC}^r_{G_e}(v)$. We say that *$e$ preserves the universal topological order* $\leq$ if $\leq$ is a universal topological order in G_e. Below we prove that a potential edge $(u, (a, r), v)$ preserves ranked SCCs and preserves the universal topological order $\leq$ iff $\mathsf{SCC}^r_G(u) \leq \mathsf{SCC}^r_G(v)$. We are also interested in the SCC-rank s of e in G_e; recall that $s = \min\{r, \kappa_{G_e}(u, v)\}$. If e preserves ranked SCCs, then $\kappa_{G_e}(u, v) = \kappa_G(u, v)$, so $s = \min\{r, \kappa_G(u, v)\}$.

Consider again the graph from Figure 8.7 on the preceding page. The intuition behind the claim below is that adding to the graph edges directed to the right, of arbitrary rank, does not change the structure of ranked SCCs, and preserves the universal topological order; similarly for adding an edge of rank r directed to the left, but contained in one $r^\uparrow$-SCC. Otherwise – if we add an edge of rank r, directed to the left, and not contained in one $r^\uparrow$-SCC – either the universal topological order is broken (because there is an edge of rank r directed to the left and connecting two different $r^\uparrow$-SCCs), or the structure of ranked SCCs is changed (if the edge merged two or more $r^\uparrow$-SCCs). Below we provide a formal proof, but we encourage the reader to skip it if the intuitive argument is convincing.

Claim 8.36. A potential edge $e = (u_0, (a_0, r_0), v_0)$ preserves ranked SCCs and preserves the universal topological order $\leq$ iff $\mathsf{SCC}^{r_0}_G(u_0) \leq \mathsf{SCC}^{r_0}_G(v_0)$.

Proof. Let G_e be the graph obtained from G by adding the edge e.

First we prove the left-to-right implication. Since e preserves the universal topological order $\leq$, $\mathsf{SCC}^{r_0}_{G_e}(u_0) \leq \mathsf{SCC}^{r_0}_{G_e}(v_0)$ (by Fact 8.35 on the preceding page, property 2 on the facing page), so either:

- $\mathsf{SCC}^{r_0}_{G_e}(u_0) = \mathsf{SCC}^{r_0}_{G_e}(v_0)$, and because e preserves ranked SCCs, $\mathsf{SCC}^{r_0}_{G}(u_0) = \mathsf{SCC}^{r_0}_{G}(v_0)$;
- $\mathsf{SCC}^{r_0}_{G_e}(u_0) < \mathsf{SCC}^{r_0}_{G_e}(v_0)$, which means that $u_0 < v_0$ (by Fact 8.35 on the preceding page, property 1 on the facing page), and $\mathsf{SCC}^{r_0}_{G}(u_0) \leq \mathsf{SCC}^{r_0}_{G}(v_0)$ (by the contrapositive of the same property); because e preserves ranked SCCs and u_0, v_0 are in different $(r_0)^{\uparrow}$-SCCs in G_e, they are also in different $(r_0)^{\uparrow}$-SCCs in G, so $\mathsf{SCC}^{r_0}_{G}(u_0) < \mathsf{SCC}^{r_0}_{G}(v_0)$.

Now we prove the right-to-left implication. By the definition of a universal topological order, for each edge $(u, (a, r), v)$ in G, $\mathsf{SCC}^{r}_{G}(u) \leq \mathsf{SCC}^{r}_{G}(v)$, because either $u \leq v$, or $\mathsf{SCC}^{r}_{G}(u) = \mathsf{SCC}^{r}_{G}(v)$. Because the same inequality holds for the edge e ($\mathsf{SCC}^{r_0}_{G}(u_0) \leq \mathsf{SCC}^{r_0}_{G}(v_0)$), it holds for each edge in G_e: for each edge $(u, (a, r), v)$ in G_e, $\mathsf{SCC}^{r}_{G}(u) \leq \mathsf{SCC}^{r}_{G}(v)$. This implies that, for nodes u, v and a rank r such that there is an $r^{\uparrow}$-walk from u to v in G_e, we have $\mathsf{SCC}^{r}_{G}(u) \leq \mathsf{SCC}^{r}_{G}(v)$; equivalently, if $\mathsf{SCC}^{r}_{G}(u) > \mathsf{SCC}^{r}_{G}(v)$, then there is no $r^{\uparrow}$-walk from u to v in G_e.

To prove that e preserves ranked SCCs, consider nodes u, v and a rank r. Because G is a subgraph of G_e, if $\mathsf{SCC}^{r}_{G}(u) = \mathsf{SCC}^{r}_{G}(v)$, then $\mathsf{SCC}^{r}_{G_e}(u) = \mathsf{SCC}^{r}_{G_e}(v)$. If $\mathsf{SCC}^{r}_{G}(u) \neq \mathsf{SCC}^{r}_{G}(v)$, then either $\mathsf{SCC}^{r}_{G}(u) < \mathsf{SCC}^{r}_{G}(v)$ or $\mathsf{SCC}^{r}_{G}(u) > \mathsf{SCC}^{r}_{G}(v)$. In the first case, there is no $r^{\uparrow}$-walk from v to u in G_e, and in the second case there is no $r^{\uparrow}$-walk from u to v in G_e, so $\mathsf{SCC}^{r}_{G_e}(u) \neq \mathsf{SCC}^{r}_{G_e}(v)$.

To prove that e preserves the universal topological order $\leq$, we need to prove that, for each edge $(u, (a, r), v)$ in G_e, either $u \leq v$, or, for each node w such that $v \leq w \leq u$, $\mathsf{SCC}^{r}_{G_e}(v) = \mathsf{SCC}^{r}_{G_e}(w) = \mathsf{SCC}^{r}_{G_e}(u)$. For each edge apart from e in G_e, the condition holds because it holds in G and because e preserves ranked SCCs. For the edge e, since $\mathsf{SCC}^{r_0}_{G}(u_0) \leq \mathsf{SCC}^{r_0}_{G}(v_0)$, either:

- $\mathsf{SCC}^{r_0}_{G}(u_0) < \mathsf{SCC}^{r_0}_{G}(v_0)$, which implies $u_0 < v_0$, so the first condition $u_0 \leq v_0$ holds;
- $\mathsf{SCC}^{r_0}_{G}(u_0) = \mathsf{SCC}^{r_0}_{G}(v_0)$, and since e preserves ranked SCCs, the second condition holds.

□

Let $\mathcal{E}_{\circledast} = \{(\mathsf{Type}^{n}_{G^{\circledast}}(u), a_{\otimes}, \mathsf{Type}^{n}_{G^{\circledast}}(v)) : (u, a_{\otimes}, v) \in E_{G^{\otimes}}\}$. We say that G is $(\leq, n)$-*saturated* if there is an edge $(u, (a, r), v)$ in G iff:

- $\mathsf{SCC}^{r}_{G}(u) \leq \mathsf{SCC}^{r}_{G}(v)$, and
- $(\tau_{\circledast}, (a, s, r), \theta_{\circledast}) \in \mathcal{E}_{\circledast}$ for $s = \min\{r, \kappa_G(u, v)\}, \tau_{\circledast} = \mathsf{Type}^{n}_{G^{\circledast}}(u), \theta_{\circledast} = \mathsf{Type}^{n}_{G^{\circledast}}(v)$.

Constructing a saturated model

We want to prove that there exists a ranked graph H and a universal topological order $\leq_H$ in H such that $H^{\circledast} \models \mathcal{S}_{\circledast}$ and H is $(\leq_H, n)$-saturated. Let H be the ranked graph with the same nodes and node labels as G, and with an edge $(u, (a, r), v)$ iff it satisfies the $(\leq, n)$-saturation conditions above for G:

- $\mathsf{SCC}^{r}_{G}(u) \leq \mathsf{SCC}^{r}_{G}(v)$, and
- $(\tau_{\circledast}, (a, s, r), \theta_{\circledast}) \in \mathcal{E}_{\circledast}$ for $s = \min\{r, \kappa_G(u, v)\}, \tau_{\circledast} = \mathsf{Type}^{n}_{G^{\circledast}}(u), \theta_{\circledast} = \mathsf{Type}^{n}_{G^{\circledast}}(v)$.

Fact 8.37. The graph G is a subgraph of H.

Because the graph H can be obtained from G by iteratively adding edges $e = (u, (a, r), v)$ such that $\mathsf{SCC}^r_G(u) \leq \mathsf{SCC}^r_G(v)$, which means that e is a potential edge in G that preserves ranked SCCs and preserves the universal topological order $\leq$, we get the following.

Corollary 8.38 (*Ranked SCC Invariance Under Saturation*)**.** For all nodes u, v and ranks r in H, $\mathsf{SCC}^r_H(u) = \mathsf{SCC}^r_H(v)$ iff $\mathsf{SCC}^r_G(u) = \mathsf{SCC}^r_G(v)$.

Corollary 8.39. The order $\leq$ is a universal topological order in H.

Additionally, from the fact that G is a subgraph of H and from the Ranked SCC Invariance Under Saturation we get the following.

Corollary 8.40. The graph $G^{\circledast}$ is a subgraph of $H^{\circledast}$.

Note that the Ranked SCC Invariance Under Saturation also implies that, for each edge $(u, (a, r), v)$ in H, its SCC-rank s satisfies $s = \min\{r, \kappa_G(u, v)\}$.

Before we prove that $H^{\circledast} \models \mathcal{S}_{\circledast}$, let us recall the statement of Claim 8.15 on page 124, as it will be used in the proof.

Claim 8.15. For three nodes u, v, w in a ranked graph G and two edges in $G^{\circledast}$:
- from u to v with a label $((\mathsf{Long}, \bot), s_1, r_1)$,
- from v to w with a label $((\mathsf{Long}, \bot), s_2, r_2)$,

there is an edge in $G^{\circledast}$ from u to w with the label $((\mathsf{Long}, \bot), s, r)$ for $r = \min\{r_1, r_2\}$, $s = \min\{s_1, s_2\}$.

Claim 8.41. For each node u in H, $\mathsf{Type}^n_{H^{\circledast}}(u) = \mathsf{Type}^n_{G^{\circledast}}(u)$.

Proof. We use the One-Step Bisimulation Lemma for the graph $H^{\circledast}$ and the function $\mathsf{SupposedType}$ mapping a node u to $\mathsf{Type}^n_{G^{\circledast}}(u)$. Since the label of each node is the same in G and in H, it suffices to show that, for each node u in $H^{\circledast}$ and each edge label $a_{\circledast}$:

$$\mathsf{Subtypes}(\mathsf{Type}^n_{G^{\circledast}}(u), a_{\circledast}) = \left\{\mathsf{Type}^{n-1}_{G^{\circledast}}(v) : v \in \mathsf{Succ}_{H^{\circledast}}(u, a_{\circledast})\right\}.$$

As usual, we show two set inclusions. Let $\tau_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(u)$.

The first inclusion, $\mathsf{Subtypes}(\tau_{\circledast}, a_{\circledast}) \subseteq \left\{\mathsf{Type}^{n-1}_{G^{\circledast}}(v) : v \in \mathsf{Succ}_{H^{\circledast}}(u, a_{\circledast})\right\}$, follows from the fact that $G^{\circledast}$ is a subgraph of $H^{\circledast}$.

To prove the other set inclusion, we use the following fact, similarly to how we used transitivity of reachability in earlier variants of the satisfiability problem. Consider an edge $(u, (a, r_1), v)$ in G of SCC-rank s_1. Because the walk in G traversing only this edge has rank r_1 and SCC-rank s_1, in $G^{\circledast}$ there is an edge from u to v with the label $((\mathsf{Long}, \bot), s_1, r_1)$. Let $\tau_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(u)$ and $\theta_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(v)$. By Claim 8.15 on page 124, for each rank r_2 and weak rank $s_2 \leq r_2$, and for $r = \min\{r_1, r_2\}$ and $s = \min\{s_1, s_2\}$,

$$\mathsf{Subtypes}(\theta_{\circledast}, ((\mathsf{Long}, \bot), s_2, r_2)) \subseteq \mathsf{Subtypes}(\tau_{\circledast}, ((\mathsf{Long}, \bot), s, r)).$$

The same set inclusions hold for every edge $(u, (a, r_1), v)$ of SCC-rank s_1 in H, by the definition of $\mathcal{E}_{\circledast}$, the construction of H, and the Ranked SCC Invariance Under Saturation.

Now we can prove that $\left\{\mathsf{Type}^{n-1}_{G^\circledast}(v) : v \in \mathsf{Succ}_{H^\circledast}(u, a_\circledast)\right\} \subseteq \mathsf{Subtypes}(\tau_\circledast, a_\circledast)$. Consider an edge $(u, a_\circledast, v)$ in $H^\circledast$ for some node v, and let $\theta_\circledast = \mathsf{Type}^n_{G^\circledast}(v)$.

- If $a_\circledast = ((\mathsf{Short}, a), s, r)$ for some a, s, r, by the construction of H and by the Ranked SCC Invariance Under Saturation, $(\tau_\circledast, (a, s, r), \theta_\circledast) \in \mathcal{E}_\circledast$. Thus, by the definition of $\mathcal{E}_\circledast$, there is an edge $(u', (a, s, r), v')$ in $G^\otimes$ such that $\mathsf{Type}^n_{G^\circledast}(u') = \tau_\circledast$ and $\mathsf{Type}^n_{G^\circledast}(v') = \theta_\circledast$. This means that $(u', a_\circledast, v')$ is an edge in $G^\circledast$, so $\theta_\circledast|_{n-1} \in \mathsf{Subtypes}(\tau_\circledast, a_\circledast)$.
- If $a_\circledast = ((\mathsf{Long}, \bot), s, r)$ for some s, r, then there exists an $r^\uparrow$-walk π from u to v in H of SCC-rank s' such that $s = \min\{s', r\}$. Because v is $r^\uparrow$-reachable from itself in G, we have $\theta_\circledast|_{n-1} \in \mathsf{Subtypes}(\theta_\circledast, ((\mathsf{Long}, \bot), r, r))$. Applying the set inclusion proved above iteratively for each edge traversed by π, for $r_2 = s_2 = r$, we get:

$$\mathsf{Subtypes}(\theta_\circledast, ((\mathsf{Long}, \bot), r, r)) \subseteq \mathsf{Subtypes}(\tau_\circledast, ((\mathsf{Long}, \bot), s, r)).$$

□

Corollary 8.42. For each modal profile $\mathcal{S}_\circledast$ of depth n that is satisfiable within $\mathbb{G}_\circledast$, there is a ranked graph H and its universal topological order $\leq$ such that $H^\circledast \models \mathcal{S}_\circledast$ and H is $(\leq, n)$-saturated.

This finishes the first step of our construction: adding edges to the graph G. Before we move on to the second step – defining an induced subgraph of bounded size that preserves ranked SCCs and types – we prove that, in a $(\leq, n)$-saturated graph, inside an $s^\uparrow$-SCC for some rank $s < \mathsf{MaxRank}_G$, there is a walk of bounded length between every two $(s+1)^\uparrow$-SCCs. For this, we first formulate simple conditions that, given an edge of SCC-rank s in a $(\leq, n)$-saturated graph, imply existence of other edges in the same $s^\uparrow$-SCC.

8.4.4 Proving existence of an edge in a saturated graph

Consider a positive integer n and a ranked graph G with a universal topological order $\leq$ such that G is $(\leq, n)$-saturated. Let $\mathcal{E}_\circledast = \{(\mathsf{Type}^n_{G^\circledast}(u), a_\otimes, \mathsf{Type}^n_{G^\circledast}(v)) : (u, a_\otimes, v) \in E_{G^\otimes}\}$. We will often consider the following scenario.

Consider an edge $e = (u, (a, r), v)$ of SCC-rank s in G, and let S be the $s^\uparrow$-SCC of e in G. Note that in $G^\otimes$ there is an edge $(u, (a, s, r), v)$. Consider nodes u', v' in S such that $\mathsf{Type}^n_{G^\circledast}(u) = \mathsf{Type}^n_{G^\circledast}(u')$ and $\mathsf{Type}^n_{G^\circledast}(v) = \mathsf{Type}^n_{G^\circledast}(v')$. We would like to define simple conditions sufficient for $(u', (a, s, r), v')$ to be an edge in $G^\otimes$. This is equivalent to the fact that $e' = (u', (a, r), v')$ is an edge of SCC-rank s in G. This, in turn, is equivalent to the following three conditions, because G is $(\leq, n)$-saturated:

- $s = \min\{r, \kappa_G(u', v')\}$;
- $\mathsf{SCC}^r_G(u') \leq \mathsf{SCC}^r_G(v')$;
- $(\tau_\circledast, (a, s, r), \theta_\circledast) \in \mathcal{E}_\circledast$, where $\tau_\circledast = \mathsf{Type}^n_{G^\circledast}(u'), \theta_\circledast = \mathsf{Type}^n_{G^\circledast}(v')$.

The last condition is satisfied, as witnessed by the edge $e = (u, (a, r), v)$ of SCC-rank s in G, so it is enough to satisfy the first two conditions. It turns out that they can be further simplified under some conditions, as stated in the claim below.

Claim 8.43 (*Saturated Edge Conditions*)**.** For a positive integer n and a ranked graph G with a universal topological order $\leq$ such that G is $(\leq, n)$-saturated, for:

- an edge $e = (u, (a, r), v)$ of SCC-rank s in G, and

- nodes u', v' in the $s^{\uparrow}$-SCC of e, with $\mathsf{Type}^n_{G^{\circledast}}(u) = \mathsf{Type}^n_{G^{\circledast}}(u')$ and $\mathsf{Type}^n_{G^{\circledast}}(v) = \mathsf{Type}^n_{G^{\circledast}}(v')$,

each of the following conditions is sufficient for $(u', (a, s, r), v')$ to be an edge in $G^{\otimes}$:

- $s = r$;
- $\mathsf{SCC}^{s+1}_G(u') < \mathsf{SCC}^{s+1}_G(v')$;
- $u' \leq u$ and $v \leq v'$.

Proof. By the argument above the claim, we only need to prove that $s = \min\{r, \kappa_G(u', v')\}$ and $\mathsf{SCC}^r_G(u') \leq \mathsf{SCC}^r_G(v')$. If $s = r$, both these conditions are true, because u', v' are in the $s^{\uparrow}$-SCC of e in G, so $\kappa_G(u', v') \geq s$ and $\mathsf{SCC}^r_G(u') = \mathsf{SCC}^r_G(v') = S$. Assume $s < r$; note that this means that $\mathsf{SCC}^{s+1}_G(u) < \mathsf{SCC}^{s+1}_G(v)$, since $(u, (a, r), v)$ is an edge of SCC-rank $s < r$ in G.

- If $\mathsf{SCC}^{s+1}_G(u') < \mathsf{SCC}^{s+1}_G(v')$, then $\kappa_G(u', v') = s$ and $\mathsf{SCC}^r_G(u') < \mathsf{SCC}^r_G(v')$ by Fact 8.35 on page 140 (the contrapositive of property 3 on page 140).
- If $u' \leq u$ and $v \leq v'$, then $\mathsf{SCC}^{s+1}_G(u') \leq \mathsf{SCC}^{s+1}_G(u) < \mathsf{SCC}^{s+1}_G(v) \leq \mathsf{SCC}^{s+1}_G(v')$ by Fact 8.35 on page 140 (the contrapositive of property 1 on page 140), which implies the condition above.

□

8.4.5 Short walks within ranked SCCs in a saturated graph

In a ranked graph G, for a rank r, an $r^{\uparrow}$-*SCC bridge* is an edge in $G|_r$ that does not belong to any SCC. Equivalently, it is an edge of rank at least r that does not belong to any $r^{\uparrow}$-SCC in G.

Claim 8.44 (*Short Walks Claim*)**.** For a modal profile $\mathcal{S}_{\circledast} = (n, \mathcal{T}_{\circledast})$ and a ranked graph G with a universal topological order $\leq$ such that G is $(\leq, n)$-saturated and $G^{\circledast} \models \mathcal{S}_{\circledast}$, for each rank $s < \mathsf{MaxRank}_G$, in each $s^{\uparrow}$-SCC, between every two nodes there is an $s^{\uparrow}$-walk visiting at most $(|\mathcal{T}_{\circledast}| + 1)^2$ different $(s+1)^{\uparrow}$-SCCs.

Proof. Consider an $s^{\uparrow}$-SCC S in G, and two nodes u_0, v_0 in S. Consider an $s^{\uparrow}$-walk π in S from u_0 to v_0 of minimal length. We will show that π satisfies the condition in the claim, by contradiction: assume that π visits more than $(|\mathcal{T}_{\circledast}| + 1)^2$ different $(s+1)^{\uparrow}$-SCCs. We will construct a shorter $s^{\uparrow}$-walk from u_0 to v_0 in S, by replacing some infix of π with a single edge. Consider the sequence of $(s+1)^{\uparrow}$-SCCs $(S_1, \ldots, S_k)$ visited by π. For each $i = 1, \ldots, k-1$, the edge traversed by π from S_i to S_{i+1} either has rank s, or is an $(s+1)^{\uparrow}$-SCC bridge and $S_i < S_{i+1}$. Thus, either:

- π traverses at least $|\mathcal{T}_{\circledast}| + 1$ edges of rank s, or
- there is an infix π' of π that is an $(s+1)^{\uparrow}$-walk and traverses at least $|\mathcal{T}_{\circledast}| + 1$ $(s+1)^{\uparrow}$-SCC bridges.

In the first case, let $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, (a_i, r_i), v_i)$. Since π traverses at least $|\mathcal{T}_{\circledast}|+1$ edges of rank s, there are two edges e_i, e_j of rank s such that $\mathsf{Type}^n_{G^{\circledast}}(v_{i-1}) = \mathsf{Type}^n_{G^{\circledast}}(v_{j-1})$ and $i < j$. Note that the SCC-rank of e_i and e_j is s, as they are in the $s^{\uparrow}$-SCC S and have rank s. We claim that $e = (v_{i-1}, (a_j, s), v_j)$ is an edge in G, so there is an $s^{\uparrow}$-walk from u_0 to v_0 shorter than π, obtained by replacing the infix $[v_{i-1}, e_i, v_i, \ldots, e_j, v_j]$ with $[v_{i-1}, e, v_j]$. The fact that e is an edge in G follows from the Saturated Edge Conditions applied to the edge $e_j = (v_{j-1}, (a_j, s), v_j)$, with $u' = v_{i-1}$ and $v' = v_j$, using the condition $s = r$.

In the second case, let $\pi' = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, (a_i, r_i), v_i)$. Since π' traverses at least $|\mathcal{T}_{\circledast}| + 1$ $(s+1)^{\uparrow}$-SCC bridges, it traverses two $(s+1)^{\uparrow}$-SCC bridges e_i, e_j such

that $i < j$ and $\mathsf{Type}^n_{G^\circledast}(v_{i-1}) = \mathsf{Type}^n_{G^\circledast}(v_{j-1})$. Note that the SCC-rank of e_i and e_j is s, because they are $(s+1)^\uparrow$-SCC bridges inside the $s^\uparrow$-SCC S. We claim that $e = (v_{i-1}, (a_j, r_j), v_j)$ is an edge in G, so there is an $s^\uparrow$-walk from u_0 to v_0 shorter than π, obtained by replacing the infix $\sigma = [v_{i-1}, e_i, v_i, \ldots, e_j, v_j]$ with $[v_{i-1}, e, v_j]$. The fact that e is an edge in G follows from the Saturated Edge Conditions applied to the edge $e_j = (v_{j-1}, (a_j, r_j), v_j)$, with $u' = v_{i-1}$ and $v' = v_j$, using the condition $\mathsf{SCC}^{s+1}_G(u') < \mathsf{SCC}^{s+1}_G(v')$: the infix σ of π is an $(s+1)^\uparrow$-walk that traverses at least one $(s+1)^\uparrow$-SCC bridge e_i, so $\mathsf{SCC}^{s+1}_G(v_{i-1}) < \mathsf{SCC}^{s+1}_G(v_j)$. □

8.4.6 A small submodel of a saturated model

Consider a modal profile $\mathcal{S}_\circledast = (n, \mathcal{T}_\circledast)$ that is satisfiable within $\mathbb{G}_\circledast$. Let G be a ranked graph with a universal topological order $\leq$ such that G is $(\leq, n)$-saturated and $G^\circledast \models \mathcal{S}_\circledast$. We assume that $\mathcal{S}_\circledast$ is a ranked modal profile, and that it contains at least one edge label; otherwise the satisfiability problem for $\mathcal{S}_\circledast$ within $\mathbb{G}_\circledast$ is trivial. Note that the maximal rank of an edge label in $\mathcal{S}_\circledast$ is equal to $\mathsf{MaxRank}_G$.

Defining an appropriate subset of nodes

We will define a subgraph of G of bounded size, induced by some subset $f(G)$ of its nodes, that, intuitively, preserves ranked SCCs and contains a node of each type. Below we give an informal description of the construction of $f(G)$. We will define recursively, for each weak rank s, for each $s^\uparrow$-SCC S in G, an appropriate subset $f(S)$ of nodes, such that the subgraph of S induced by $f(S)$ preserves ranked SCCs and contains a node of each type in S. The first two cases essentially describe the construction from the previous chapter.

- For $s = \mathsf{MaxRank}_G$, $f(S)$ can be an arbitrary set containing one node of each type in S.
- For $s = 0$, recall that the $0^\uparrow$-SCC S is the whole graph G; for each $\tau_\circledast \in \mathcal{T}_\circledast$ we consider the greatest $1^\uparrow$-SCC S' that contains a node of type $\tau_\circledast$, and we define $f(S)$ as the union of $f(S')$ for all such $1^\uparrow$-SCCs S'.
- For $1 \leq s < \mathsf{MaxRank}_G$, for an $s^\uparrow$-SCC S, consider $f(S)$ defined using the same idea as for $s = 0$; the subgraph of S induced by $f(S)$ defined this way might not be strongly connected, and we want to preserve ranked SCCs, so apart from the union of $f(S')$ for all appropriate $(s+1)^\uparrow$-SCCs S', we fix walks of minimal lengths between them, and include $f(S')$ for all $(s+1)^\uparrow$-SCCs S' visited by these walks. Thanks to the Short Walks Claim, the total number of $(s+1)^\uparrow$-SCCs taken into account is bounded.

Formally, for each weak rank s, for each $s^\uparrow$-SCC S in G, we will define $f(S)$: a subset of nodes in S. For a ranked SCC S in G, let $\mathsf{Profile}^n_{G^\circledast}(S)$ denote the set consisting of $\mathsf{Type}^n_{G^\circledast}(u)$ for all nodes u in S.

For $s = \mathsf{MaxRank}_G$ and an $s^\uparrow$-SCC S in G, let $f(S)$ be the set containing, for each $\tau_\circledast \in \mathsf{Profile}^n_{G^\circledast}(S)$, the greatest node u (in $\leq$) in S such that $\mathsf{Type}^n_{G^\circledast}(u) = \tau_\circledast$.

Claim 8.45. For each $(\mathsf{MaxRank}_G)^\uparrow$-SCC S in G:

- the subgraph of S induced by $f(S)$ is strongly connected;
- $|f(S)| \leq |\mathcal{T}_\circledast|$;
- for each $\tau_\circledast \in \mathsf{Profile}^n_{G^\circledast}(S)$, $f(S)$ contains the greatest node u in S such that $\mathsf{Type}^n_{G^\circledast}(u) = \tau_\circledast$.

Proof. The second and the third point follow directly from the construction of $f(S)$. To prove the first point, consider two nodes u, v in $f(S)$. There is a $(\mathsf{MaxRank}_G)^{\uparrow}$-walk from u to v in S, because S is an $(\mathsf{MaxRank}_G)^{\uparrow}$-SCC. Note that each edge traversed by this walk has rank $\mathsf{MaxRank}_G$ and SCC-rank $\mathsf{MaxRank}_G$. Replacing each node w in this walk with the greatest node w' in S such that $\mathsf{Type}^n_{G^{\circledast}}(w) = \mathsf{Type}^n_{G^{\circledast}}(w')$ results in an $(\mathsf{MaxRank}_G)^{\uparrow}$-walk from u to v in the subgraph of S induced by $f(S)$; the fact that each edge traversed by this walk is in G follows from the Saturated Edge Conditions, using the condition $s = r$. □

For a rank s such that $1 \leq s < \mathsf{MaxRank}_G$ and for an $s^{\uparrow}$-SCC S in G, let X be the set containing, for each $\tau_{\circledast} \in \mathsf{Profile}^n_{G^{\circledast}}(S)$, the greatest $(s+1)^{\uparrow}$-SCC S' in S such that $\tau_{\circledast} \in \mathsf{Profile}^n_{G^{\circledast}}(S')$. Let $(S_1, \ldots, S_k)$ be an arbitrary linear ordering of $(s+1)^{\uparrow}$-SCCs in X. For each $i = 1, \ldots, k-1$, let π_i be a fixed $s^{\uparrow}$-walk of minimal length from S_i to S_{i+1} in S, and let π_k be a fixed $s^{\uparrow}$-walk of minimal length from S_k to S_1 in S. We define $f(S)$ as the union of $f(S')$ for all $(s+1)^{\uparrow}$-SCCs S' visited by π_i for some $i \in \{1, \ldots, k\}$; note that this includes all $(s+1)^{\uparrow}$-SCCs in X.

Claim 8.46. For each rank s, for each $s^{\uparrow}$-SCC S in G:

- the subgraph of S induced by $f(S)$ is strongly connected;
- $|f(S)| \leq (|\mathcal{T}_{\circledast}| + 1)^{3(\mathsf{MaxRank}_G - s)+1}$;
- for each $\tau_{\circledast} \in \mathsf{Profile}^n_{G^{\circledast}}(S)$, $f(S)$ contains the greatest node u in S such that $\mathsf{Type}^n_{G^{\circledast}}(u) = \tau_{\circledast}$.

Proof. We prove the claim inductively for $s = \mathsf{MaxRank}_G, \ldots, 1$. The base case, for $s = \mathsf{MaxRank}_G$, follows from Claim 8.45 on the previous page. Below we prove the inductive step for $s < \mathsf{MaxRank}_G$.

The third point follows directly from the construction of $f(S)$ and the inductive assumption.

The second point follows from the construction, the Short Walks Claim, and the inductive assumption, as follows: $k \leq |\mathcal{T}_{\circledast}|$ by the construction, and each walk π_i for $i = 1, \ldots, k$ visits at most $(|\mathcal{T}_{\circledast}|+1)^2$ $(s+1)^{\uparrow}$-SCCs by the Short Walks Claim, so $f(S)$ is the union of $f(S')$ for at most $|\mathcal{T}_{\circledast}| \cdot (|\mathcal{T}_{\circledast}|+1)^2 \leq (|\mathcal{T}_{\circledast}|+1)^3$ $(s+1)^{\uparrow}$-SCCs S' in S; by the inductive assumption, $|f(S')| \leq (|\mathcal{T}_{\circledast}| + 1)^{3(\mathsf{MaxRank}_G - s - 1)+1}$.

To prove the first point, consider the walk π_i for some $i \in \{1, \ldots, k\}$. Consider an edge $e = (u, (a, r), v)$ traversed by π between two different $(s+1)^{\uparrow}$-SCCs S'_1, S'_2, and let $\tau_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(u)$, $\theta_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(v)$. Note that the SCC-rank of e is s. By the inductive assumption, there is a node u' in $f(S'_1)$ and a node v' in $f(S'_2)$ such that $\mathsf{Type}^n_{G^{\circledast}}(u') = \tau_{\circledast}$ and $\mathsf{Type}^n_{G^{\circledast}}(v') = \theta_{\circledast}$. We claim that $(u', (a, r), v')$ is an edge in G, so it is also an edge in the subgraph of S induced by $f(S)$; we get that the subgraph of S induced by $f(S)$ is strongly connected by applying the same reasoning to each edge between different $(s+1)^{\uparrow}$-SCCs traversed by each walk π_i for $i = 1, \ldots, k$, and using the inductive assumption that the subgraph of S' induced by $f(S')$ is strongly connected for each $(s+1)^{\uparrow}$-SCC S' in S. It remains to prove that $(u', (a, r), v')$ is an edge in G. We use the Saturated Edge Conditions applied to the edge $e = (u, (a, r), v)$ of SCC-rank s, and for u', v' defined above, using the conditions $s = r$ or $\mathsf{SCC}^{s+1}_G(u') < \mathsf{SCC}^{s+1}_G(v')$: if $s < r$ then e is an $(s+1)^{\uparrow}$-SCC bridge in S, so $\mathsf{SCC}^{s+1}_G(u') = \mathsf{SCC}^{s+1}_G(u) < \mathsf{SCC}^{s+1}_G(v) = \mathsf{SCC}^{s+1}_G(v')$. □

For the whole graph G, we define $f(G)$ as follows. Let X be the set containing, for each $\tau_{\circledast} \in \mathcal{T}_{\circledast}$, the greatest $1^{\uparrow}$-SCC S in G such that $\tau_{\circledast} \in \mathsf{Profile}^n_{G^{\circledast}}(S)$. Let $f(G)$ be the union of $f(S)$ for all $S \in X$.

From Claim 8.46 we get that:

- $|f(G)| \leq (|\mathcal{T}_{\circledast}| + 1)^{3\mathsf{MaxRank}_G}$;
- for each $\tau_{\circledast} \in \mathcal{T}_{\circledast}$, $f(G)$ contains the greatest node u in G such that $\mathsf{Type}^n_{G^{\circledast}}(u) = \tau_{\circledast}$.

Fact 8.47. For each rank r, for each $r^{\uparrow}$-SCC S in G, either $f(S) \subseteq f(G)$, or $V_S \cap f(G) = \emptyset$.

Proving properties of the induced subgraph

Let H be the subgraph of G induced by $f(G)$.

Fact 8.48. For all ranks r and all nodes u, v in H, $\mathsf{SCC}^r_H(u) = \mathsf{SCC}^r_H(v)$ iff $\mathsf{SCC}^r_G(u) = \mathsf{SCC}^r_G(v)$.

Corollary 8.49. The graph $H^{\otimes}$ is an induced subgraph of $G^{\otimes}$. The graph $H^{\circledast}$ is a subgraph of $G^{\circledast}$.

Claim 8.50. For each node u in H, $\mathsf{Type}^n_{H^{\circledast}}(u) = \mathsf{Type}^n_{G^{\circledast}}(u)$.

Proof. We use the One-Step Bisimulation Lemma for the graph $H^{\circledast}$ and the function $\mathsf{SupposedType}$ mapping a node u to $\mathsf{Type}^n_{G^{\circledast}}(u)$. Each node in H has the same label as in G, so it suffices to show that, for each node u in $H^{\circledast}$ and each edge label $a_{\circledast}$:

$$\mathsf{Subtypes}(\mathsf{Type}^n_{G^{\circledast}}(u), a_{\circledast}) = \left\{\mathsf{Type}^{n-1}_{G^{\circledast}}(v) : v \in \mathsf{Succ}_{H^{\circledast}}(u, a_{\circledast})\right\}.$$

As usual, we show two set inclusions. Let $\tau_{\circledast} = \mathsf{Type}^n_{G^{\circledast}}(u)$.

The inclusion $\left\{\mathsf{Type}^{n-1}_{G^{\circledast}}(v) : v \in \mathsf{Succ}_{H^{\circledast}}(u, a_{\circledast})\right\} \subseteq \mathsf{Subtypes}(\tau_{\circledast}, a_{\circledast})$ follows from the fact that $H^{\circledast}$ is a subgraph of $G^{\circledast}$.

It remains to show that $\mathsf{Subtypes}(\tau_{\circledast}, a_{\circledast}) \subseteq \left\{\mathsf{Type}^{n-1}_{G^{\circledast}}(v) : v \in \mathsf{Succ}_{H^{\circledast}}(u, a_{\circledast})\right\}$. Consider an edge $(u, a_{\circledast}, v)$ in $G^{\circledast}$ for some v. Let $a_{\circledast} = (a_+, s, r)$, and let S be the $s^{\uparrow}$-SCC of u and v in G. Note that, since u is a node in H, $f(S) \subseteq f(G) = V_H$.

We first consider the case of $a_+ = (\mathsf{Short}, a)$ for some a. Let v' be the greatest node in S such that $\mathsf{Type}^n_{G^{\circledast}}(v) = \mathsf{Type}^n_{G^{\circledast}}(v')$. Note that $v' \in f(S)$, so it is a node in H. We claim that $(u, a_{\circledast}, v')$ is an edge in $H^{\circledast}$. Since H is an induced subgraph of G with preserved ranked SCCs, it is enough to prove that $(u, (a, r), v')$ is an edge of SCC-rank s in G. This follows from the Saturated Edge Conditions applied to the edge $(u, (a, r), v)$, with $u' = u$ and v' defined above, using the condition $u' \leq u$, $v \leq v'$.

It remains to consider $a_+ = (\mathsf{Long}, \bot)$. We prove the claim by induction over u, in the reverse $\leq$ order.

If $s = r$, let π be an $r^{\uparrow}$-walk in S from u to v. We need to prove that there exists an $r^{\uparrow}$-walk in the subgraph of S induced by $f(S)$ from u to some v' such that $\mathsf{Type}^{n-1}_{G^{\circledast}}(v) = \mathsf{Type}^{n-1}_{G^{\circledast}}(v')$. Let v' be the greatest node in S such that $\mathsf{Type}^n_{G^{\circledast}}(v) = \mathsf{Type}^n_{G^{\circledast}}(v')$. Note that $v' \in f(S)$, so v' is a node in H. Because the subgraph of S induced by $f(S)$ is strongly connected, it contains an $r^{\uparrow}$-walk from u to v'.

In the rest of this proof we consider the case of $s < r$. Let π be an $r^{\uparrow}$-walk in S from u to v of SCC-rank s. We need to prove that there is an edge with label $((\mathsf{Long}, \bot), s, r)$ in $H^{\circledast}$ from u to some v' such that $\mathsf{Type}^{n-1}_{G^{\circledast}}(v) = \mathsf{Type}^{n-1}_{G^{\circledast}}(v')$. Let S_r be the $r^{\uparrow}$-SCC of u in G. We split the walk π into three parts:

- let π_1 be the longest prefix of π contained in S_r;

- let $e_2 = (x, (a_2, r_2), y)$ be the next edge traversed by π;
- let π_3 be the remaining suffix of π.

The following long edges are in $G^{\circledast}$:

- $(u, ((\mathsf{Long}, \bot), r, r), x)$, because π_1 is an $r^{\uparrow}$-walk within an $r^{\uparrow}$-SCC S_r;
- $(x, ((\mathsf{Long}, \bot), s_2, r), y)$, where s_2 is the SCC-rank of e_2; note that $s_2 < r$;
- $(y, ((\mathsf{Long}, \bot), s'_3, r), v)$, where s_3 is the SCC-rank of π_3 and $s'_3 = \min\{s_3, r\}$.

We have $s = \min\{r, s_2, s'_3\}$. From the third long edge we get:

$$\mathsf{Type}^{n-1}_{G^{\circledast}}(v) \in \mathsf{Subtypes}(\mathsf{Type}^{n}_{G^{\circledast}}(y), ((\mathsf{Long}, \bot), s'_3, r)).$$

The idea is to find corresponding nodes x', y' in H and use the inductive assumption for y' to get a node v' such that $\mathsf{Type}^{n-1}_{G^{\circledast}}(v) = \mathsf{Type}^{n-1}_{G^{\circledast}}(v')$. This will give corresponding three long edges in $H^{\circledast}$, and by Claim 8.15 on page 124 we will obtain the required long edge from u to v'.

Let S_{s_2} be the $(s_2)^{\uparrow}$-SCC of u in G.

- Let x' be the greatest node in S_r such that $\mathsf{Type}^{n}_{G^{\circledast}}(x) = \mathsf{Type}^{n}_{G^{\circledast}}(x')$.
- Let y' be the greatest node in S_{s_2} such that $\mathsf{Type}^{n}_{G^{\circledast}}(y) = \mathsf{Type}^{n}_{G^{\circledast}}(y')$.

We claim that x', y' are nodes in H, because $u, x' \in f(S_r)$ and $u, y' \in f(S_{s_2})$. There is an $r^{\uparrow}$-walk from u to x' in H, so there is a long edge in $H^{\circledast}$ from u to x' with label $((\mathsf{Long}, \bot), r, r)$.

We claim that $(x', (a_2, r_2), y')$ is an edge in G of SCC-rank s_2, by the Saturated Edge Conditions applied to the edge $(x, (a_2, r_2), y)$ in S_{s_2}, using the condition $\mathsf{SCC}^{s_2+1}_G(x') < \mathsf{SCC}^{s_2+1}_G(y')$, as $\mathsf{SCC}^{s_2+1}_G(x') = \mathsf{SCC}^{s_2+1}_G(x) < \mathsf{SCC}^{s_2+1}_G(y) \leq \mathsf{SCC}^{s_2+1}_G(y')$:

- $\mathsf{SCC}^{s_2+1}_G(x') = \mathsf{SCC}^{s_2+1}_G(x)$ because $s_2 < r$ and x, x' are in S_r;
- $\mathsf{SCC}^{s_2+1}_G(x) < \mathsf{SCC}^{s_2+1}_G(y)$ because the edge $(x, (a_2, r_2), y)$ has SCC-rank $s_2 < r \leq r_2$;
- $\mathsf{SCC}^{s_2+1}_G(y) \leq \mathsf{SCC}^{s_2+1}_G(y')$, because $y \leq y'$ by the definition of y'.

Thus, there is a long edge in $H^{\circledast}$ from x' to y' with label $((\mathsf{Long}, \bot), s_2, r)$.

We want to use the inductive assumption for y'. For this, we need to show that $u < y'$:

- $\mathsf{SCC}^{r}_G(u) = \mathsf{SCC}^{r}_G(x) < \mathsf{SCC}^{r}_G(y)$, so $u < y$;
- $y \leq y'$ by the definition of y'.

By the inductive assumption for y', there is a long edge in $H^{\circledast}$ from y' with label $((\mathsf{Long}, \bot), s'_3, r)$ to some v' such that $\mathsf{Type}^{n-1}_{G^{\circledast}}(v) = \mathsf{Type}^{n-1}_{G^{\circledast}}(v')$.

By applying Claim 8.15 on page 124 for the three long edges in $H^{\circledast}$ defined above, we get that there is a long edge in $H^{\circledast}$ from u to v' with label $((\mathsf{Long}, \bot), s, r)$, since $s = \min\{r, s_2, s'_3\}$. □

We get that $H^{\circledast} \models \mathcal{S}_{\circledast}$, because for every $\tau_{\circledast} \in \mathcal{T}_{\circledast}$ there is a node u in $f(G)$ such that $\mathsf{Type}^{n}_{G^{\circledast}}(u) = \tau_{\circledast}$. Thus, we proved Proposition 8.33: there exists a ranked graph H with at most $(|\mathcal{T}_{\circledast}| + 1)^{3R}$ nodes such that $H^{\circledast} \models \mathcal{S}_{\circledast}$, where R is the maximal rank of edge labels in $\mathcal{S}_{\circledast}$. This ends the proof of the decidability of the satisfiability problem for $\mathbb{G}_{\circledast}$.

8.4.7 Summary

We built upon the techniques from the previous chapter to construct a small witness to the satisfiability problem for $\mathbb{G}_{\circledast}$. The proof was much more involved, although the construction itself is not particularly complicated. Let us summarize the construction.

- We considered a ranked graph G and its universal topological order $\leq$ such that $G^{\circledast} \models \mathcal{S}_{\circledast}$.
- We constructed a $(\leq, n)$-saturated graph by adding all potential edges $(u, (a, r), v)$ such that $\mathsf{SCC}^r_G(u) \leq \mathsf{SCC}^r_G(v)$ and there is an edge $(u', (a, r), v')$ in G with appropriate types of nodes and an appropriate SCC-rank.
- We recursively defined the set of nodes $f(S)$ for each ranked SCC S: for an $s^{\uparrow}$-SCC S, for each type of node in S we considered the greatest $(s+1)^{\uparrow}$-SCC containing a node of this type; additionally, we fixed walks of minimal length between them, and defined $f(S)$ as the union of $f(S')$ for all $(s+1)^{\uparrow}$-SCCs S' visited by these walks.
- We defined H as the subgraph of G induced by $f(G)$, and proved that $H^{\circledast} \models \mathcal{S}_{\circledast}$.

The proofs of the Short Walks Claim and the Saturated Edge Conditions required new ideas and understanding of the structure of ranked SCCs, but the proof that $H^{\circledast} \models \mathcal{S}_{\circledast}$ used the same idea as the proof in the previous chapter, with the most involved part being the inductive reasoning over the reversed (universal) topological order.

8.5 Summary

In this chapter we defined *ranked graphs* with a notion of $r^{\uparrow}$*-reachability* that is adjusted to techniques developed in the previous chapters; as we will see in the following chapters, it is also suitable for solving the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox. Apart from generalizing the notions to ranked graphs and proving that they are well-behaved, the most significant technical contribution is the construction in the satisfiability result.

Chapter 9

Automata-annotated graphs

A *semiautomaton* is a graph $\mathcal{A}$ in which, for each edge label $a \in \Sigma_{\mathcal{A}}$, each node has exactly one outgoing edge with label a. The *$\mathcal{A}$-reachability annotation* of a graph G, denoted $G^{\mathcal{A}+}$, is the graph obtained from G by replacing each edge label a with (Short, a), and adding an edge from u to v with label (Long, p, q) iff there is a walk from u to v in G and a walk from p to q in $\mathcal{A}$ with equal edge label sequences. An example of a semiautomaton $\mathcal{A}$, a graph G, and the $\mathcal{A}$-reachability annotation $G^{\mathcal{A}+}$ of G is shown in Figure 9.1 on the following page.

For a semiautomaton $\mathcal{A}$, let $\mathbb{G}_{\mathcal{A}}$ be the class of graphs that use only edge labels in $\Sigma_{\mathcal{A}}$, and let $\mathbb{G}_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}_{\mathcal{A}}\}$. We require $\Sigma_G \subseteq \Sigma_{\mathcal{A}}$ for convenience; this assumption does not influence the reduction from the UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox presented in the next two chapters. In this chapter, we prove the following theorem.

Theorem 9.1. The minimal downsets problem for $\mathbb{G}_{\mathcal{A}+}$ is computable for each semiautomaton $\mathcal{A}$.

We show it by reducing the problem to the minimal downsets problem for $\mathbb{G}_{\oplus}$, solved in the previous chapter. It will be clear from the reduction that the problem is computable also when the semiautomaton is treated as a part of the input.

9.1 Definitions

Automata and semiautomata

A *nondeterministic finite automaton*, which we call simply an *automaton*, is a graph $\mathcal{A}$ with a distinguished *initial* node q_0 and a set of *final* nodes F. Automata are typically defined as machines, with an input, states, and possibly an output. To keep consistent with the standard terminology, we refer to nodes in an automaton as *states*, to edges as *transitions*, to walks as *runs*, and to $\Sigma_{\mathcal{A}}$ as the *alphabet*. A run *over a word* $\vec{a}$ is a run with edge label sequence $\vec{a}$.

The language *recognized* by an automaton $(\mathcal{A}, q_0, F)$ is the set of edge label sequences of runs in $\mathcal{A}$ that begin in q_0 and end in some state in F. The languages recognized by automata are called *regular*

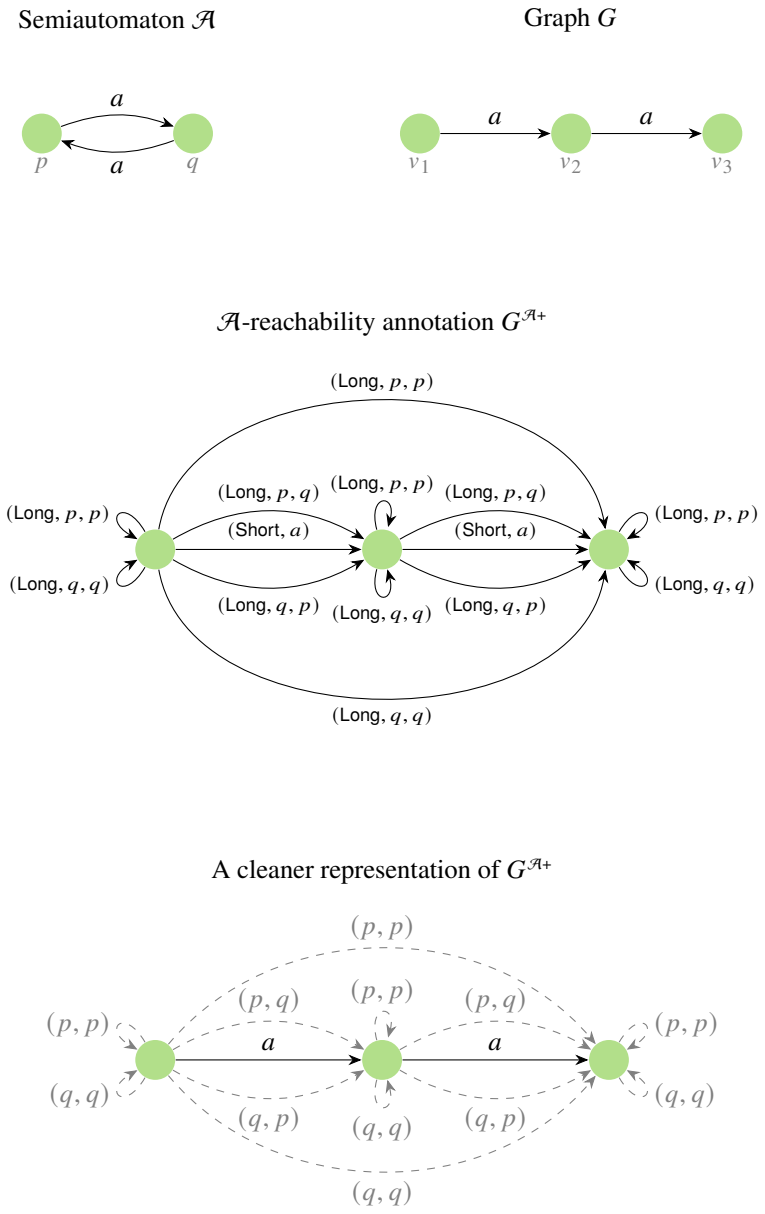


Figure 9.1: A semiautomaton $\mathcal{A}$, a graph G, and the $\mathcal{A}$-reachability annotation $G^{\mathcal{A}+}$ of G.

languages[1].

An automaton $(\mathcal{A}, q_0, F)$ is *deterministic* if, for each $a \in \Sigma_\mathcal{A}$, each state in $\mathcal{A}$ has exactly one outgoing transition with label a. In such a case, the graph $\mathcal{A}$ is called a *semiautomaton*, which coincides with the definition provided earlier. The *transition function* of a semiautomaton $\mathcal{A}$, denoted $\delta_\mathcal{A}$, is the function from $V_\mathcal{A} \times \Sigma_\mathcal{A}$ to $V_\mathcal{A}$ such that $\delta_\mathcal{A}(p, a) = q$ iff $(p, a, q) \in E_\mathcal{A}$. The transition function can be lifted to words over $\Sigma_\mathcal{A}$, with $\delta_\mathcal{A}(p, \vec{a}) = q$ iff q is the last state in the unique run from p over the word $\vec{a}$ in $\mathcal{A}$.

In this chapter, we work with semiautomata rather than automata, but we defined them for context. In Chapter 11, automata will be used to define UCRPQs.

Note that node labels do not play any role in the definition of (semi)automata. We will sometimes draw semiautomata with each node having a different label, to easily distinguish between the nodes.

$\mathcal{A}$-reachability annotations

Consider a semiautomaton $\mathcal{A}$ and a graph G.

For states p, q in $\mathcal{A}$, an $\mathcal{A}_{p,q}$*-walk* in G is a walk π in G with an edge label sequence $\vec{a}$ such that $\delta_\mathcal{A}(p, \vec{a}) = q$. As defined earlier, the $\mathcal{A}$-reachability annotation of G, denoted $G^{\mathcal{A}+}$, is the graph obtained from G by replacing each edge label a with (Short, a), and adding an edge from u to v with label (Long, p, q) iff there is an $\mathcal{A}_{p,q}$-walk from u to v in G.

As defined earlier, $\mathbb{G}_\mathcal{A}$ is the class of graphs G such that $\Sigma_G \subseteq \Sigma_\mathcal{A}$, and $\mathbb{G}_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}_\mathcal{A}\}$.

The claim below makes it possible to define $\mathcal{A}$-reachability annotations of graphs of bounded size as a trace-based graph transformation: for a graph G, it depends on walks of length at most $|V_\mathcal{A}| \cdot |V_G|$.

Claim 9.2. For nodes u, v in G and states p, q in $\mathcal{A}$, if there is an $\mathcal{A}_{p,q}$-walk from u to v in G, then there is one that visits each node at most $|V_\mathcal{A}|$ times.

Proof. By the definition of an $\mathcal{A}_{p,q}$-walk, there is a walk π from u to v in G and a run ρ from p to q in $\mathcal{A}$ such that the edge label sequences of π and ρ are equal; consider such π and ρ of minimal length. Let $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$, and let $\rho = [q_0, t_1, q_1, \ldots, t_m, q_m]$. If π visits some node more than $|V_\mathcal{A}|$ times, then there are two indices $i < j$ such that $v_i = v_j$ and $q_i = q_j$. Then, π' obtained from π by removing the infix $[v_i, \ldots, e_j]$ is an $\mathcal{A}_{p,q}$-walk from u to v in G, because ρ' obtained from ρ by removing the infix $[q_i, \ldots, t_j]$ is a run from p to q in $\mathcal{A}$, and they have equal edge label sequences. This is a contradiction, because we chose π and ρ of minimal length. □

9.2 Combining ranked graphs with semiautomata

In the context of the minimal downsets problem, we say that two classes C_1, C_2 of graphs are *equivalent* if, for every modal profile $\mathcal{S}$, the answers to the minimal downsets problem for $\mathcal{S}$ are the same for C_1 and for C_2.

[1] More precisely, regular languages over the set $\mathbb{U}$ defined in Chapter 2, because we restrict edge labels to this set.

Informally, we would like to define a class of graphs G for which we can consider $G^{\mathcal{A}+}$ and $G^{\oplus}$ at the same time, with a meaningful connection between long edges in both annotations. More formally, the main idea of the reduction is to define a class $\mathbb{G}_{\widehat{\mathcal{A}}^{\oplus}} \subseteq \mathbb{G}_{\oplus}$ of ranked graphs such that $\mathbb{G}_{\mathcal{A}+}$ (or, more precisely, an equivalent class) can be obtained from $\mathbb{G}_{\widehat{\mathcal{A}}^{\oplus}}$ by a trace-based graph transformation, which means that there is a reduction between the corresponding minimal downsets problems, by Lemma 5.14 on page 62, as long as the transformation is strongly computable. The core of the reduction is the following construction on semiautomata.

9.2.1 The permutation construction

First appearance record

For a sequence σ, its *first appearance record*, denoted $\mathsf{FAR}(\sigma)$, is the subsequence of σ consisting of the first appearance of each element. For $\sigma = (p_1, \ldots, p_n)$ and $\mathsf{FAR}(\sigma) = (q_1, \ldots, q_k)$, we have $p_i = q_i$ for indices i up to the first repeat in σ; formally, there is a unique positive integer r such that:

- for each $i \leq r$, $p_i = q_i$;
- for each $i > r$, $p_i = q_j$ for some $j < i$.

This unique number r will be treated as a rank in ranked graphs considered later.

Note that, for a set X and a sequence σ over X, $\mathsf{FAR}(\sigma)$ is a permutation of X if σ contains each element of X at least once.

The permutation semiautomaton

Consider a semiautomaton $\mathcal{A}$. Let us assume that $\mathcal{A}$ comes with an implicit linear order on its states, and let $\vec{q}_0$ be the sequence of all states in $\mathcal{A}$ in increasing order.

We lift the transition function $\delta_{\mathcal{A}}$ to sequences of states: for a sequence $\vec{q} = (q_1, \ldots, q_n)$ of states in $\mathcal{A}$ and for $a \in \Sigma_{\mathcal{A}}$, we denote by $\delta_{\mathcal{A}}(\vec{q}, a)$ the sequence $(\delta_{\mathcal{A}}(q_1, a), \ldots, \delta_{\mathcal{A}}(q_n, a))$.

The *permutation semiautomaton* of $\mathcal{A}$, denoted $\widehat{\mathcal{A}}$, is the semiautomaton in which the states are all permutations of the states in $\mathcal{A}$, and the transition function $\delta_{\widehat{\mathcal{A}}}$ is defined as follows. For a state $\vec{q}$ in $\widehat{\mathcal{A}}$ and for $a \in \Sigma_{\mathcal{A}}$, we define $\delta_{\widehat{\mathcal{A}}}(\vec{q}, a)$ as $\mathsf{FAR}(\delta_{\mathcal{A}}(\vec{q}, a) \cdot \vec{q}_0)$: the first appearance record of the concatenation of $\delta_{\mathcal{A}}(\vec{q}, a)$ and the sequence $\vec{q}_0$ of all the states in $\mathcal{A}$ in increasing order. Because $\vec{q}_0$ contains each state in $\mathcal{A}$, $\delta_{\widehat{\mathcal{A}}}(\vec{q}, a)$ is a permutation of the states in $\mathcal{A}$. An example of a semiautomaton $\mathcal{A}$ and its permutation semiautomaton is shown in Figure 9.2.

Below we discuss properties of transitions and runs in the permutation semiautomaton; refer to Figure 9.3 on page 156 for an example. Consider a transition $(\vec{p}, a, \vec{q})$ in $\widehat{\mathcal{A}}$, with $\vec{p} = (p_1, \ldots, p_n)$ and $\vec{q} = (q_1, \ldots, q_n)$. There exists a unique positive integer r such that:

- for each $i \leq r$, $\delta_{\mathcal{A}}(p_i, a) = q_i$;
- for each $i > r$, $\delta_{\mathcal{A}}(p_i, a) = q_j$ for some $j < i$.

We call this number r the *rank* of the transition $(\vec{p}, a, \vec{q})$ in $\widehat{\mathcal{A}}$. For example, in the run presented in Figure 9.3, the ranks of traversed transitions are 3, 1, 2, 1, 3, respectively, and are equal to the number of corresponding horizontal transitions of $\mathcal{A}$. We refer to indices in $\vec{p}$ and $\vec{q}$ as *positions*: if $\delta_{\mathcal{A}}(p_i, a) = q_j$, we say that the transition *maps* position i to position j; note that $j \leq i$. We also say that the transition

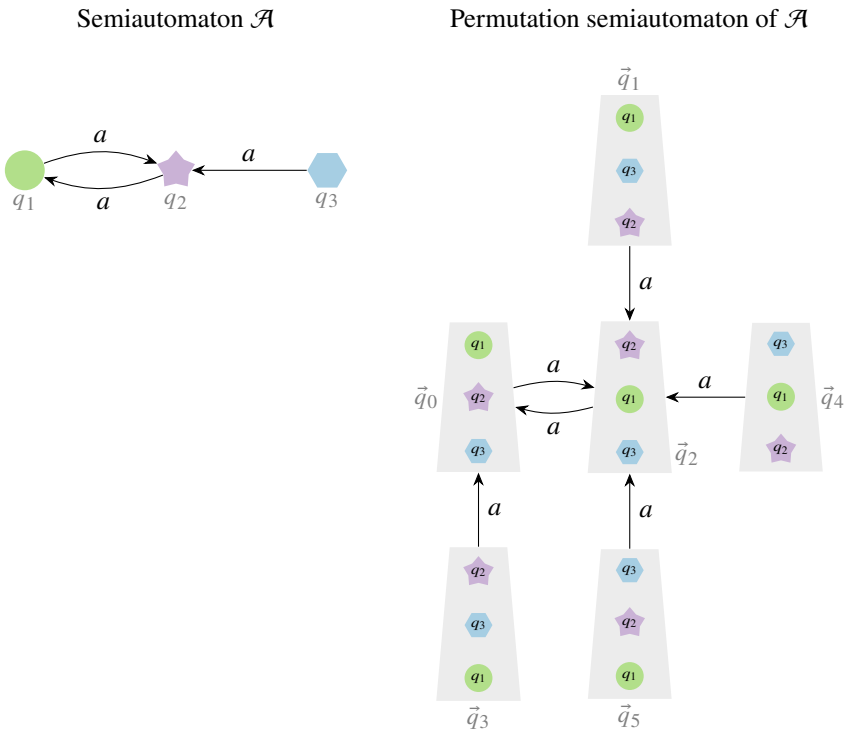


Figure 9.2: A semiautomaton $\mathcal{A}$ and its permutation semiautomaton. Each state in the permutation automaton is a sequence (permutation) of states in $\mathcal{A}$, drawn in a top-down order.

maps the state p_i to the state q_j. For example, in the run presented in Figure 9.3, the second traversed transition maps position 2 to position 1, and maps the state q_2 to itself.

Fact 9.3. A transition of rank r in $\widehat{\mathcal{A}}$ maps a position i to itself iff $i \leq r$.

Consider a run $\widehat{\pi}$ in $\widehat{\mathcal{A}}$ from $\vec{p}$ to $\vec{q}$ over a word $\vec{a}$. We lift the above notions to runs: we say that $\widehat{\pi}$ *maps* p to q, and maps position i to position j, if the run in $\mathcal{A}$ from the state p at position i in $\vec{p}$ over the word $\vec{a}$ ends in the state q at position j in $\vec{q}$. Note that $j \leq i$, as the same inequality holds for every transition in $\widehat{\mathcal{A}}$. For example, the run presented in Figure 9.3 maps position 3 to position 1, and maps the state q_3 to the state q_2.

Fact 9.4. A run in $\widehat{\mathcal{A}}$ maps a position i to itself iff it traverses only transitions of ranks at least i.

The presented construction, even though relatively simple and quite powerful, does not seem to be particularly known; as mentioned in the introduction, the earliest mention of this idea we have found is by Hesse (2003), in the context of dynamic computational complexity. The construction is also used by Bojańczyk (2009) to solve the following problem. We are given a regular language and a word, and we want to answer queries of the form: given an infix of the word, does it belong to the regular language? If the language is given by a deterministic automaton with k states, the word has length n, and the infixes are given as pairs of indices, the problem can be solved with a preprocessing in time $\text{poly}(k) \cdot n$, and query answering in time $\text{poly}(k)$. Thus, if we fix the automaton, the preprocessing is done in linear time, and query answering takes constant time. One can find interesting analogies between this result and the reductions presented in this chapter, where the use of the permutation semiautomaton

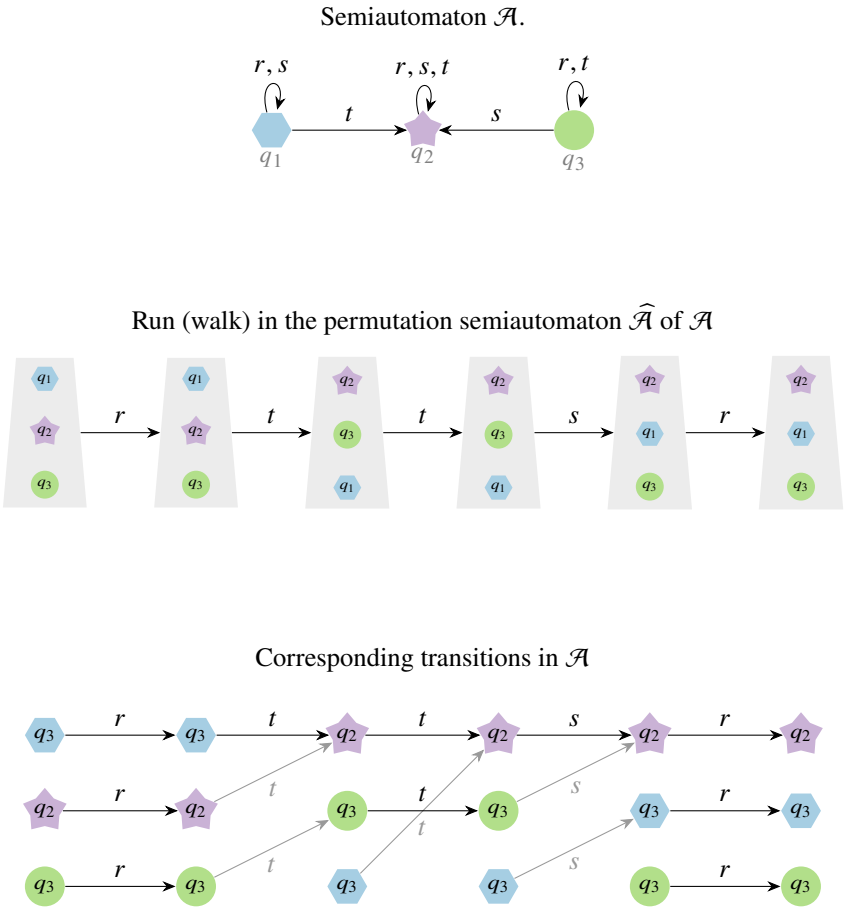


Figure 9.3: A semiautomaton $\mathcal{A}$ (with multiple loops represented by a single loop, with labels separated by commas), a run of its permutation semiautomaton, and corresponding transitions of $\mathcal{A}$. Note that all transitions of $\mathcal{A}$ in the picture are either horizontal or go "upward", to earlier indices in sequences (colored gray). This property holds for every transition (and run) in the permutation semiautomaton.

and ravelling constructions can be seen as "preprocessing", and the "constant time query answering" corresponds to a trace-based graph transformation that depends on walks of length at most $2|V_{\mathcal{A}}|$ – dependent on $\mathcal{A}$, but not on the size of the transformed graph.

9.2.2 Classes of graphs

Consider a semiautomaton $\mathcal{A}$ and its permutation semiautomaton $\widehat{\mathcal{A}}$.

Recall that $\mathbb{G}_{\mathcal{A}}$ is the class of graphs G such that $\Sigma_G \subseteq \Sigma_{\mathcal{A}}$, and $\mathbb{G}_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}_{\mathcal{A}}\}$. We want to reduce the minimal downsets problem for $\mathbb{G}_{\mathcal{A}+}$ to the same problem for $\mathbb{G}_{\oplus}$, intuitively, by defining a trace-based graph transformation between the classes; more precisely, we define a transformation between two slightly different classes of graphs:

- $\mathbb{G}'_{\mathcal{A}+} \subseteq \mathbb{G}_{\mathcal{A}+}$ such that $\mathbb{G}'_{\mathcal{A}+}$ is equivalent to $\mathbb{G}_{\mathcal{A}+}$;
- $\mathbb{G}_{\widehat{\mathcal{A}}\oplus} \subseteq \mathbb{G}_{\oplus}$ such that $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ is a restriction of $\mathbb{G}_{\oplus}$ to graphs satisfying some modal profiles.

Before we define these classes and briefly describe the reductions between the corresponding minimal downsets problems, let us show the ravelling construction that motivates the definition of $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$, and will be used to prove that $\mathbb{G}_{\mathcal{A}+}$ and $\mathbb{G}'_{\mathcal{A}+}$ are equivalent.

Converting graphs to ranked graphs

Consider a graph G in $\mathbb{G}_{\mathcal{A}}$. Let $\vec{q}_0$ be an arbitrary state in $\widehat{\mathcal{A}}$, and let $H = \mathsf{SafeRavel}_f(G)$ for f mapping a walk with edge label sequence $\vec{a}$ to $\delta_{\widehat{\mathcal{A}}}(\vec{q}_0, \vec{a})$: the last state in the run of $\widehat{\mathcal{A}}$ from $\vec{q}_0$ over the word $\vec{a}$. Recall that the nodes in H are pairs $(\vec{q}, v)$, where $\vec{q}$ is a state in $\widehat{\mathcal{A}}$ and v is a node in G, and note that for each edge $((\vec{q}_1, v_1), a, (\vec{q}_2, v_2))$, $(\vec{q}_1, a, \vec{q}_2)$ is a transition in $\widehat{\mathcal{A}}$. Thus, we can easily convert H to a ranked graph, treating the rank of this transition as the rank of the edge.

The class $\mathbb{G}'_{\mathcal{A}+} \subseteq \mathbb{G}_{\mathcal{A}+}$ will contain all graphs of the form $H^{\mathcal{A}+}$ for $G \in \mathbb{G}_{\mathcal{A}}$ and $H = \mathsf{SafeRavel}_f(G)$. As we will see, this means that $\mathbb{G}'_{\mathcal{A}+}$ is equivalent to $\mathbb{G}_{\mathcal{A}+}$, by the properties of ravelling constructions and because $\mathcal{A}$-reachability annotations of graphs of bounded size can be defined as a trace-based graph transformation.

$\widehat{\mathcal{A}}$-ranked graphs

An $\widehat{\mathcal{A}}$-*ranked graph* $G_{\widehat{\mathcal{A}}}$ is a ranked graph in which node labels are pairs $(A, \vec{q})$ such that $\vec{q}$ is a state in $\widehat{\mathcal{A}}$, and, for each edge $(v_1, (a, r), v_2)$, with v_1 labelled $(A_1, \vec{q}_1)$ and v_2 labelled $(A_2, \vec{q}_2)$, $(\vec{q}_1, a, \vec{q}_2)$ is a transition of rank r in $\widehat{\mathcal{A}}$.

Let $\mathbb{G}_{\widehat{\mathcal{A}}}$ be the class of all $\widehat{\mathcal{A}}$-ranked graphs, and let $\mathbb{G}_{\widehat{\mathcal{A}}\oplus} = \{(G_{\widehat{\mathcal{A}}})^{\oplus} : G_{\widehat{\mathcal{A}}} \in \mathbb{G}_{\widehat{\mathcal{A}}}\}$. We will show that the minimal downsets problem for $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ is the same as for $\mathbb{G}_{\oplus}$, only restricted to modal profiles of an appropriate shape.

Deannotated $\widehat{\mathcal{A}}$-ranked graphs

The *deannotation* of an $\widehat{\mathcal{A}}$-ranked graph $G_{\widehat{\mathcal{A}}}$ is the graph obtained from $G_{\widehat{\mathcal{A}}}$ by replacing each node label $(A, \vec{q})$ with A, and replacing each edge label (a, r) with a.

Let $\mathbb{G}'_{\mathcal{A}}$ be the class of deannotations of $\widehat{\mathcal{A}}$-ranked graphs, and $\mathbb{G}'_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}'_{\mathcal{A}}\}$. We will show that $\mathbb{G}'_{\mathcal{A}+}$ is equivalent to $\mathbb{G}_{\mathcal{A}+}$, because $\mathbb{G}'_{\mathcal{A}}$ contains all graphs obtained by the ravelling construction discussed earlier.

We will also show that $\mathbb{G}'_{\mathcal{A}+}$ can be obtained from $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ by a strongly computable trace-based graph transformation, which means that there is a reduction between the corresponding minimal downsets problems. Importantly, note that we want to apply the same trace-based graph transformation for all graphs in the class, so it needs to depend on walks of length at most k, where k is independent of the size of the graph; in our case, $k = 2|V_{\mathcal{A}}|$. To describe the transformation, for an $\widehat{\mathcal{A}}$-ranked graph $G_{\widehat{\mathcal{A}}}$ and its deannotation G, we will describe the relation between long edges in $G^{\mathcal{A}+}$, walks in $G_{\widehat{\mathcal{A}}}$, and walks (of length at most $2|V_{\mathcal{A}}|$) in $(G_{\widehat{\mathcal{A}}})^{\oplus}$.

9.3 Reductions

We need to show the three reductions described above:

- from $\mathbb{G}_{\mathcal{A}+}$ to $\mathbb{G}'_{\mathcal{A}+}$ (the classes are equivalent, by the discussed ravelling construction);
- from $\mathbb{G}'_{\mathcal{A}+}$ to $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ (a trace-based graph transformation);
- from $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ to $\mathbb{G}_{\oplus}$ (restricting to modal profiles of an appropriate shape).

We describe them in the reverse order.

Adjusting modal profiles

Recall that $\mathbb{G}_{\widehat{\mathcal{A}}}$ is the class of all $\widehat{\mathcal{A}}$-ranked graphs, and $\mathbb{G}_{\widehat{\mathcal{A}}\oplus} = \{G^{\oplus} : G \in \mathbb{G}_{\widehat{\mathcal{A}}}\}$. To describe the reduction, we need to define the relevant modal profiles, and show that they accurately correspond to the class $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$.

An *$\widehat{\mathcal{A}}$-ranked short-long graph* $G_{\widehat{\mathcal{A}}\oplus}$ is a ranked short-long graph in which the node labels are pairs $(A, \vec{q})$ such that $\vec{q}$ is a state in $\widehat{\mathcal{A}}$, and for each short edge $(v_1, ((\mathsf{Short}, a), r), v_2)$, where the labels of v_1 and v_2 are $(A_1, \vec{q}_1)$ and $(A_2, \vec{q}_2)$, respectively, $(\vec{q}_1, a, \vec{q}_2)$ is a transition of rank r in $\widehat{\mathcal{A}}$.

Fact 9.5. A ranked graph G is an $\widehat{\mathcal{A}}$-ranked graph iff $G^{\oplus}$ is an $\widehat{\mathcal{A}}$-ranked short-long graph.

An *$\widehat{\mathcal{A}}$-ranked short-long modal profile* is a modal profile $\mathcal{S}_{\widehat{\mathcal{A}}\oplus} = (n, \mathcal{T}_{\widehat{\mathcal{A}}\oplus})$ in which, for each $\tau \in \mathcal{T}_{\widehat{\mathcal{A}}\oplus}$, $\mathsf{Representative}(\tau)$ is an $\widehat{\mathcal{A}}$-ranked short-long graph.

Fact 9.6. For a ranked graph G and a modal profile $\mathcal{S}_{\oplus}$ of $G^{\oplus}$, G is an $\widehat{\mathcal{A}}$-ranked graph iff $\mathcal{S}_{\oplus}$ is an $\widehat{\mathcal{A}}$-ranked short-long modal profile.

Consider an instance of the minimal downsets problem for $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$: a modal profile $\mathcal{S}$. By Fact 9.6, if $\mathcal{S}$ is an $\widehat{\mathcal{A}}$-ranked short-long modal profile, the answer is the same as the answer to the minimal downsets problem for $\mathbb{G}_{\oplus}$, and otherwise the answer is the empty set. Clearly, checking if $\mathcal{S}$ is an $\widehat{\mathcal{A}}$-ranked short-long modal profile is decidable. The same strategy yields a reduction from the constructive satisfiability problem for $\mathbb{G}_{\widehat{\mathcal{A}}\oplus}$ to the constructive satisfiability problem for $\mathbb{G}_{\oplus}$.

A graph transformation

Recall that $\mathbb{G}'_{\mathcal{A}}$ is the class of deannotations of $\widehat{\mathcal{A}}$-ranked graphs, and $\mathbb{G}'_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}'_{\mathcal{A}}\}$. We want to show that $\mathbb{G}'_{\mathcal{A}+} = \{t(G_{\widehat{\mathcal{A}}_\oplus}) : G_{\widehat{\mathcal{A}}_\oplus} \in \mathbb{G}_{\widehat{\mathcal{A}}_\oplus}\}$ for some strongly computable trace-based graph transformation t.

Consider an $\widehat{\mathcal{A}}$-ranked graph $G_{\widehat{\mathcal{A}}}$ and its deannotation G. Let $G_{\widehat{\mathcal{A}}_\oplus} = (G_{\widehat{\mathcal{A}}})^\oplus$.

When constructing the permutation semiautomaton $\widehat{\mathcal{A}}$, we defined what it means for a run in $\widehat{\mathcal{A}}$ to map a state p to a state q. Below we lift this notion to walks in $G_{\widehat{\mathcal{A}}}$ and in $G_{\widehat{\mathcal{A}}_\oplus}$.

Consider a walk $\pi_{\widehat{\mathcal{A}}}$ in $G_{\widehat{\mathcal{A}}}$, with $\pi_{\widehat{\mathcal{A}}} = [v_0, e_1, v_1, \ldots, e_m, v_m]$, each $e_i = (v_{i-1}, (a_i, r_i), v_i)$, and each v_i having label $(A_i, \vec{q}_i)$. The run in $\widehat{\mathcal{A}}$ *encoded* in $\pi_{\widehat{\mathcal{A}}}$ is the run $[\vec{q}_0, t_1, \vec{q}_1, \ldots, t_m, \vec{q}_m]$, with each $t_i = (\vec{q}_{i-1}, a_i, \vec{q}_i)$. We say that $\pi_{\widehat{\mathcal{A}}}$ *maps* p to q if the encoded run in $\widehat{\mathcal{A}}$ maps p to q, and $\pi_{\widehat{\mathcal{A}}}$ maps position i to position j if the encoded run maps position i to position j.

Recall that G is the deannotation of $G_{\widehat{\mathcal{A}}}$.

Fact 9.7. The following are equivalent.
- There is a long edge $(u, (\mathsf{Long}, p, q), v)$ in $G^{\mathcal{A}+}$.
- There is a walk in $G_{\widehat{\mathcal{A}}}$ from u to v that maps p to q.

Recall that there is a long edge of rank r from v_1 to v_2 in $G_{\widehat{\mathcal{A}}_\oplus}$ iff there is an $r^\uparrow$-walk from v_1 to v_2 in $G_{\widehat{\mathcal{A}}}$: a walk traversing only edges of ranks at least r. Each $r^\uparrow$-walk in $G_{\widehat{\mathcal{A}}}$ maps each position $i \leq r$ to itself; in other words, if the label of v_1 is $(A_1, \vec{q}_1)$ and the label of v_2 is $(A_2, \vec{q}_2)$, then, for each position $i \leq r$, the encoded run in $\widehat{\mathcal{A}}$ maps the state at position i in $\vec{q}_1$ to the state at position i in $\vec{q}_2$.

Consider a walk $\pi_{\widehat{\mathcal{A}}_\oplus}$ in $G_{\widehat{\mathcal{A}}_\oplus}$, with $\pi_{\widehat{\mathcal{A}}_\oplus} = [v_0, e_1, v_1, \ldots, e_m, v_m]$, each $e_i = (v_{i-1}, a_i, v_i)$, and each v_i having label $(A_i, \vec{q}_i)$. We say that $\pi_{\widehat{\mathcal{A}}_\oplus}$ *maps* p to q if there exists a sequence of positions $(k_0, \ldots, k_m)$ such that:
- the state at position k_0 in $\vec{q}_0$ is p;
- for each edge e_i:
  - if $a_i = ((\mathsf{Short}, a), r)$, then the transition $(\vec{q}_{i-1}, a, \vec{q}_i)$ in $\widehat{\mathcal{A}}$ maps position k_{i-1} to k_i;
  - if $a_i = ((\mathsf{Long}, \bot), r)$, then $k_{i-1} = k_i \leq r$;
- the state at position k_m in $\vec{q}_m$ is q.

Note that if such a sequence $(k_0, \ldots, k_m)$ exists, it is non-increasing and unique.

Fact 9.8. The following are equivalent.
- There is a long edge $(u, (\mathsf{Long}, p, q), v)$ in $G^{\mathcal{A}+}$.
- There is a walk in $G_{\widehat{\mathcal{A}}_\oplus}$ from u to v that maps p to q.

Claim 9.9. If there is a walk in $G_{\widehat{\mathcal{A}}_\oplus}$ from u to v that maps p to q, there is one of length at most $2|V_{\mathcal{A}}|$.

Proof. Let $\pi_{\widehat{\mathcal{A}}_\oplus}$ be a walk in $G_{\widehat{\mathcal{A}}_\oplus}$ from u to v that maps p to q, with $\pi_{\widehat{\mathcal{A}}_\oplus} = [v_0, e_1, v_1, \ldots, e_m, v_m]$, each $e_i = (v_{i-1}, a_i, v_i)$, and each v_i having label $(A_i, \vec{q}_i)$. Let $(k_0, \ldots, k_m)$ be the corresponding sequence of non-increasing positions. Let $\pi'_{\widehat{\mathcal{A}}_\oplus}$ be obtained from $\pi_{\widehat{\mathcal{A}}_\oplus}$ by replacing each maximal non-trivial infix $[v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j]$ such that $k_i = k_j$ by a single long edge $(v_i, ((\mathsf{Long}, \bot), k_i), v_j)$.

Let $(k'_0, \ldots, k'_n)$ be the sequence of positions obtained from $(k_0, \ldots, k_m)$ by analogous operations, replacing each maximal infix $(k_i, \ldots, k_j)$ of length at least 2 such that $k_i = k_j$ by (k_i, k_j).

We claim that $\pi'_{\widehat{\mathcal{A}}_\oplus}$ is a walk in $G_{\widehat{\mathcal{A}}_\oplus}$ from u to v of length at most $2|V_{\mathcal{A}}|$ that maps p to q. The resulting walk has length at most $2|V_{\mathcal{A}}|$, because the sequence $(k'_0, \ldots, k'_n)$ is non-increasing and each maximal infix of equal numbers has length at most 2. For each replaced infix $[v_i, \ldots, v_j]$, let $r = k_i = k_j$; the infix traverses only transitions of ranks at least r (by Fact 9.4 on page 155), so it is an $r^{\uparrow}$-walk, which means that the long edge $(v_i, ((\mathsf{Long}, \bot), r), v_j)$ is in $G_{\widehat{\mathcal{A}}_\oplus}$ (by Fact 8.6 on page 121). This long edge trivially satisfies the condition $k_i = k_j \leq r$, required for the resulting walk to still map p to q. □

We are ready to define the required trace-based graph transformation. Let $\mathsf{RecoverReachability}_{\mathcal{A}+}(G_{\widehat{\mathcal{A}}_\oplus})$ denote the graph obtained from $G_{\widehat{\mathcal{A}}_\oplus}$ by:

- replacing each node label $(A, \vec{q})$ with A;
- dropping all long edges;
- replacing each short edge label $((\mathsf{Short}, a), r)$ with (Short, a);
- adding an edge $(u, (\mathsf{Long}, p, q), v)$ if there is a walk in $G_{\widehat{\mathcal{A}}_\oplus}$ from u to v that maps p to q.

We get that $G^{\mathcal{A}+} = \mathsf{RecoverReachability}_{\mathcal{A}+}(G_{\widehat{\mathcal{A}}_\oplus})$, from the definitions of an $\widehat{\mathcal{A}}$-ranked graph and its deannotation, and by Fact 9.8 on the previous page. The function $\mathsf{RecoverReachability}_{\mathcal{A}+}$ can be easily expressed as a trace-based graph transformation; by the claim above, it depends on walks of length at most $2|V_{\mathcal{A}}|$.

It remains to show that this transformation is strongly computable. Clearly, it is computable: the functions g_Σ (mapping traces to sets of edge labels) and g_Γ (mapping node labels to node labels) are computable. To show that it is strongly computable, it is enough to show that, for a given modal profile $\mathcal{S}_{\mathcal{A}+}$ of a graph $G_{\mathcal{A}+} \in \mathbb{G}'_{\mathcal{A}+}$, we can compute finite sets of node labels and edge labels that can be used by graphs $G_{\widehat{\mathcal{A}}_\oplus} \in \mathbb{G}_{\widehat{\mathcal{A}}_\oplus}$ such that $G_{\mathcal{A}+} = \mathsf{RecoverReachability}_{\mathcal{A}+}(G_{\widehat{\mathcal{A}}_\oplus})$. The set of edge labels used by $\widehat{\mathcal{A}}$-ranked graphs is already finite and easily computable (pairs of the form (a, r), where $a \in \Sigma_{\mathcal{A}}$, and r is a rank of a transition in $\widehat{\mathcal{A}}$), and node labels in $G_{\widehat{\mathcal{A}}_\oplus}$ must be of the form $(A, \vec{q})$, where A is a node label in $\mathcal{S}_{\mathcal{A}+}$, and $\vec{q}$ is a state in $\widehat{\mathcal{A}}$.

By the properties of trace-based graph transformations, there are Turing reductions: from the constructive satisfiability problem for $\mathbb{G}'_{\mathcal{A}+}$ to the same problem for $\mathbb{G}_{\widehat{\mathcal{A}}_\oplus}$ (by Lemma 5.13 on page 61), and from the minimal downsets problem for $\mathbb{G}'_{\mathcal{A}+}$ to the same problem for $\mathbb{G}_{\widehat{\mathcal{A}}_\oplus}$ (by Lemma 5.14).

The equivalence between $\mathbb{G}_{\mathcal{A}+}$ and $\mathbb{G}'_{\mathcal{A}+}$

Recall that $\mathbb{G}_{\mathcal{A}}$ is the class of graphs G such that $\Sigma_G \subseteq \Sigma_{\mathcal{A}}$, $\mathbb{G}_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}_{\mathcal{A}}\}$, $\mathbb{G}'_{\mathcal{A}}$ is the class of deannotations of $\widehat{\mathcal{A}}$-ranked graphs, and $\mathbb{G}'_{\mathcal{A}+} = \{G^{\mathcal{A}+} : G \in \mathbb{G}'_{\mathcal{A}}\}$. We want to show that $\mathbb{G}_{\mathcal{A}+}$ and $\mathbb{G}'_{\mathcal{A}+}$ are equivalent in the context of the minimal downsets problem.

Consider a modal profile $\mathcal{S}$ of depth n.

We need to prove that, for each graph $G_{\mathcal{A}+}$ in one class such that $G_{\mathcal{A}+} \models \mathcal{S}$, there is a graph $H_{\mathcal{A}+}$ in the other class such that $H_{\mathcal{A}+} \models \mathcal{S}$ and $\mathsf{Downset}_n(H_{\mathcal{A}+}) \subseteq \mathsf{Downset}_n(G_{\mathcal{A}+})$. We have $\mathbb{G}'_{\mathcal{A}+} \subseteq \mathbb{G}_{\mathcal{A}+}$, so in one direction this is trivially satisfied, for $G_{\mathcal{A}+} = H_{\mathcal{A}+}$. It remains to prove that for each graph $G_{\mathcal{A}+} \in \mathbb{G}_{\mathcal{A}+}$ such that $G_{\mathcal{A}+} \models \mathcal{S}$ there is a graph $H_{\mathcal{A}+} \in \mathbb{G}'_{\mathcal{A}+}$ such that $H_{\mathcal{A}+} \models \mathcal{S}$ and $\mathsf{Downset}_n(H_{\mathcal{A}+}) \subseteq \mathsf{Downset}_n(G_{\mathcal{A}+})$.

Consider a graph G in $\mathbb{G}_{\mathcal{A}}$ such that $G^{\mathcal{A}+} \models \mathcal{S}$. Recall the ravelling construction described earlier. Let $\vec{q}_0$ be an arbitrary state in $\widehat{\mathcal{A}}$, and let $H = \mathsf{SafeRavel}_f(G)$ for f mapping a walk with edge label sequence $\vec{a}$ to $\delta_{\widehat{\mathcal{A}}}(\vec{q}_0, \vec{a})$: the last state in the run of $\widehat{\mathcal{A}}$ from $\vec{q}_0$ over the word $\vec{a}$. Recall that nodes in H are pairs $(\vec{q}, v)$, where $\vec{q}$ is a state in $\widehat{\mathcal{A}}$ and v is a node in G, and note that for each edge $((\vec{q}_1, v_1), a, (\vec{q}_2, v_2))$, $(\vec{q}_1, a, \vec{q}_2)$ is a transition in $\widehat{\mathcal{A}}$. This means that H is a deannotation of an $\widehat{\mathcal{A}}$-ranked graph, so $H \in \mathbb{G}'_{\mathcal{A}}$ and $H^{\mathcal{A}+} \in \mathbb{G}'_{\mathcal{A}+}$.

By the properties of ravelling constructions, G and H have equal modal profiles of all depths, and there is a homomorphism from H to G. For fixed graphs G, H, their $\mathcal{A}$-reachability annotations can be defined using a trace-based graph transformation, so $H^{\mathcal{A}+}$ and $G^{\mathcal{A}+}$ also have equal modal profiles of all depths (by Corollary 5.12 on page 61), and there is a homomorphism from $H^{\mathcal{A}+}$ to $G^{\mathcal{A}+}$ (by Fact 5.8 on page 60), so $H^{\mathcal{A}+} \models \mathcal{S}$ and $\mathsf{Downset}_n(H^{\mathcal{A}+}) \subseteq \mathsf{Downset}_n(G^{\mathcal{A}+})$, which is what we needed to prove.

Thus, to solve the minimal downsets problem for $\mathbb{G}'_{\mathcal{A}+}$, it is enough to solve the same problem for $\mathbb{G}_{\mathcal{A}+}$. To solve the constructive satisfiability problem for $\mathbb{G}'_{\mathcal{A}+}$, it is enough to solve it for $\mathbb{G}_{\mathcal{A}+}$, and apply the $\mathsf{SafeRavel}_f$ function defined above; it is easily computable.

9.4 Summary

We showed that, for every semiautomaton $\mathcal{A}$, the minimal downsets problem for $\mathbb{G}_{\mathcal{A}+}$ is computable, and the constructive satisfiability problem for $\mathbb{G}_{\mathcal{A}+}$ is computable. We reduced the problem to the minimal downsets problem for $\mathbb{G}_{\oplus}$; the core of the reduction was the construction of a permutation semiautomaton, which allowed us to encode $\mathcal{A}_{p,q}$-walks in a graph as walks of bounded length in the ranked reachability annotation of a ranked graph. Thanks to this encoding, we could use trace-based graph transformations to reduce the corresponding minimal downsets problems, with some additional simple reductions required for technical reasons.

This was the final variant of the minimal downsets problem solved in this thesis; it remains to solve the corresponding homomorphism entailment problem, and the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox.

Chapter 10

Homomorphism entailment for automata-annotated graphs

In this chapter we prove the following proposition.

Proposition 10.1. There is a Turing reduction from the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$ to the homomorphism coverage problem for $\mathbb{G}_{\mathcal{A}+}$, for each semiautomaton $\mathcal{A}$.

We use the fact that the constructive satisfiability problem for $\mathbb{G}_{\mathcal{A}+}$ is computable – this assumption is not strictly necessary, but it makes some arguments much cleaner. Importantly, the presented reduction is valid also if the semiautomaton is treated as a part of the input.

Recall the notions of counterexamples.

- For an instance $\mathcal{S}, \mathcal{H}$ of the homomorphism coverage problem for a class C of graphs, a *counterexample* to $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ is a graph in $\mathsf{Models}_C(\mathcal{S})$ that does not admit a homomorphism from any graph in $\mathcal{H}$.
- For an instance $\tilde{\mathcal{S}}, \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for a class C of graphs, a *counterexample* to $\mathcal{G} \models_{\mathrm{hom}}^{C,\tilde{\mathcal{S}}} \mathcal{H}$ is a graph in $\mathsf{Models}_C(\tilde{\mathcal{S}})$ that admits a homomorphism from some graph in $\mathcal{G}$, but does not admit a homomorphism from any graph in $\mathcal{H}$.

The idea is the same as in Chapter 4. We use detachments to split graphs into inner and outer graphs. For an instance $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}}), \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem, we iterate over all inner graphs (of bounded size) that admit a homomorphism from some graph in $\mathcal{G}$ and contain all nominal nodes (with labels in Γ_{N}), and show that $\mathcal{S}$ and $\mathcal{H}$ "distribute" well over the inner and the outer graph, which allows us to compute all relevant modal profiles $\mathcal{S}_{\mathsf{outer}}$ and sets $\mathcal{H}_{\mathsf{outer}}$ of graphs, and solve the homomorphism coverage problem for them.

The main difference is that, for a detachment of a graph in $\mathbb{G}_{\mathcal{A}+}$, the resulting inner and outer graphs might not belong to $\mathbb{G}_{\mathcal{A}+}$, because, intuitively, they contain long edges that combine information from both parts. For this reason, for a graph D such that $D^{\mathcal{A}+} \in \mathbb{G}_{\mathcal{A}+}$, we consider a detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ of the graph D instead of $D^{\mathcal{A}+}$, and work with $(D_{\mathsf{inner}})^{\mathcal{A}+}$ and $(D_{\mathsf{outer}})^{\mathcal{A}+}$. Because they

contain only some long edges from $D^{\mathcal{A}+}$, proving that $\mathcal{S}$ and $\mathcal{H}$ distribute well over the inner and the outer graph is more challenging.

10.1 Definitions – recap

This section contains definitions that are taken verbatim from earlier chapters.

Decision problems

For a class C of graphs, a modal profile $\mathcal{S}$, and a set $\mathcal{H}$ of graphs, we write $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$ if every graph in $\mathsf{Models}_C(\mathcal{S})$ admits a homomorphism from some graph in $\mathcal{H}$.

Homomorphism coverage of a modal profile within a class C of graphs

Input: A modal profile $\mathcal{S}$ and a set $\mathcal{H}$ of graphs.
Question: Does $\mathcal{H} \twoheadrightarrow \mathsf{Models}_C(\mathcal{S})$?

A modal profile is *shallow* if it has depth 1. A *modal profile with nominals* $\tilde{\mathcal{S}}$ is a modal profile $\mathcal{S}$ with a set $\Gamma_N \subseteq \Gamma_{\mathcal{S}}$ of *nominals*. A graph is called Γ_N-*unique* if it contains exactly one node with label A for each $A \in \Gamma_N$; then, these nodes are called *nominal nodes*. A graph G *satisfies* $\tilde{\mathcal{S}}$, denoted $G \models \tilde{\mathcal{S}}$, if G satisfies $\mathcal{S}$ and is Γ_N-unique. For a class C of graphs, we define $\mathsf{Models}_C(\tilde{\mathcal{S}}) = \{G \in C : G \models \tilde{\mathcal{S}}\}$.

For a class C of graphs, a shallow modal profile with nominals $\tilde{\mathcal{S}}$, and sets $\mathcal{G}, \mathcal{H}$ of graphs, we write $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$ if, for every graph $D \in \mathsf{Models}_C(\tilde{\mathcal{S}})$, if D admits a homomorphism from some graph in $\mathcal{G}$, then D admits a homomorphism from some graph in $\mathcal{H}$.

Homomorphism entailment under a modal profile within a class C of graphs

Input: A shallow modal profile with nominals $\tilde{\mathcal{S}}$ and two sets $\mathcal{G}, \mathcal{H}$ of graphs.
Question: Does $\mathcal{G} \models^{C,\tilde{\mathcal{S}}}_{\mathrm{hom}} \mathcal{H}$?

Detachments

Consider a graph D and a subset V of its nodes. We define two graphs: D_{inner}, which is the subgraph of D induced by V; and D_{outer}, obtained from D by removing all nodes in V and modifying node labels as follows. For each node u, we change its label A to $(A, \mathsf{EIn}, \mathsf{EOut})$, where:

- EIn is the set of pairs (v, a) such that $v \in V$ and (v, a, u) is an edge in D;
- EOut is the set of pairs (a, v) such that $v \in V$ and (u, a, v) is an edge in D.

Intuitively, the sets EIn and EOut represent all edges in D between the node u and nodes in V. The pair $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$ is called the V-*detachment* of D; we refer to the graph D_{inner} as the *inner graph*, and to D_{outer} as the *outer graph*. A *detachment* of D is its V-detachment for some $V \subseteq V_D$. We tend to use the letter u to denote a node in the outer graph, v to denote a node in the inner graph, and w to denote a node in D.

Attachments

For a graph D_{inner}, an *outer node label* for D_{inner} is a node label of the form $(A, \mathsf{EIn}, \mathsf{EOut})$, where:

- EIn is a finite set of pairs of the form (v, a), where v is a node in D_{inner} and a is an edge label;
- EOut is a finite set of pairs of the form (a, v), where v is a node in D_{inner} and a is an edge label.

For finite sets $\Gamma, \Sigma \subseteq \mathbb{U}$, let $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma, \Sigma)$ be the set of all outer node labels $(A, \mathsf{EIn}, \mathsf{EOut})$ for D_{inner} such that $A \in \Gamma$ and $a \in \Sigma$ for all $(v, a) \in \mathsf{EIn}$ and all $(a, v) \in \mathsf{EOut}$. Note that the function $\mathsf{OuterLabels}$ is computable.

For disjoint graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ such that each node label in D_{outer} is an outer node label for D_{inner}, the *attachment* of $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, denoted $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, is the graph obtained from the union of D_{inner} and D_{outer} by:

- adding an edge (v, a, u) for each node u in D_{outer} with label $(A, \mathsf{EIn}, \mathsf{EOut})$ such that $(v, a) \in \mathsf{EIn}$;
- adding an edge (u, a, v) for each node u in D_{outer} with label $(A, \mathsf{EIn}, \mathsf{EOut})$ such that $(a, v) \in \mathsf{EOut}$;
- replacing the label $(A, \mathsf{EIn}, \mathsf{EOut})$ with A for each node in D_{outer}.

The construction can be easily generalized to non-disjoint graphs $D_{\mathsf{inner}}, D_{\mathsf{outer}}$ by using their disjoint union instead of union, and, instead of u and v, referring to corresponding nodes in the disjoint union. To avoid cluttering the notation, we implicitly assume that the considered graphs are always disjoint.

10.2 Overview

In Chapter 4, for a counterexample D to $\mathcal{G} \models^{\mathbb{G}, \tilde{S}}_{\mathrm{hom}} \mathcal{H}$, we considered its V-detachment $(D_{\mathsf{inner}}, D_{\mathsf{outer}})$, where V was the image of a homomorphism from some graph in $\mathcal{G}$ to D, extended with the nominal nodes in D. We showed that:

- $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}}) = D$;
- the graph D_{inner} and the shallow modal profile of D_{outer} determine the shallow modal profile of D;
- if $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$, then $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}}) \to \mathsf{AttachGraphs}(D_{\mathsf{inner}}, D_{\mathsf{outer}}) = D$;
- if $H \to D$, then there is a graph H_{outer} of size bounded in terms of H such that $H_{\mathsf{outer}} \to D_{\mathsf{outer}}$ and $H \to \mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$.

These properties allowed us to "distribute" the modal profile $\mathcal{S}$ and the graphs in $\mathcal{H}$ over the inner and the outer graph, reducing the problem to the homomorphism coverage problem.

For a semiautomaton $\mathcal{A}$, an *$\mathcal{A}$-reachability annotated graph* is a graph in which all edge labels are of the form (Short, a) for some $a \in \Sigma_{\mathcal{A}}$, or (Long, p, q) for some states p, q in $\mathcal{A}$. Because we work mostly with $\mathcal{A}$-reachability annotated graphs, in this section we do not use subscripts to denote them (e.g. $G_{\mathcal{A}^+}$), and instead denote graphs that are *not* $\mathcal{A}$-reachability annotated by a tilde (e.g. $\tilde{D}$).

We want to define a function $\mathsf{AttachGraphs}_{\mathcal{A}^+}$ satisfying properties analogous to the properties of $\mathsf{AttachGraphs}$ listed above. That is, for a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$, we would like to reason about $\tilde{D}^{\mathcal{A}^+}$ in terms of $(\tilde{D}_{\mathsf{inner}})^{\mathcal{A}^+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}^+}$. The first three properties translate directly – we will show that:

- $\mathsf{AttachGraphs}_{\mathcal{A}^+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}^+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}^+}) = \tilde{D}^{\mathcal{A}^+}$;
- the graph $\tilde{D}_{\mathsf{inner}}$ and the shallow modal profile of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}^+}$ determine the shallow modal profile of $\tilde{D}^{\mathcal{A}^+}$;
- if $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}^+}$, then $\mathsf{AttachGraphs}_{\mathcal{A}^+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}^+}, H_{\mathsf{outer}}) \to \tilde{D}^{\mathcal{A}^+}$.

However, the last condition gets slightly more complicated; one could expect the following statement:

- for a graph H, if $H \to \tilde{D}^{\mathcal{A}^+}$, then there is a graph H_{outer} of size bounded in terms of H such that

$H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, H_{\mathsf{outer}})$.

We are able to prove a similar statement, although not for $(\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}$, but for the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V (the nodes of $\tilde{D}_{\mathsf{inner}}$); we will call it D_{inner}. Note that D_{inner} might contain long edges that correspond to walks in $\tilde{D}$ that traverse both $\tilde{D}_{\mathsf{inner}}$ and $\tilde{D}_{\mathsf{outer}}$; we will show that these long edges are determined by the graph $\tilde{D}_{\mathsf{inner}}$ and the shallow modal profile of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$. An example of a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$, its detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$, and graphs $(\tilde{D})^{\mathcal{A}+}$, $(\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}$ and D_{inner} is shown in Figure 10.1.

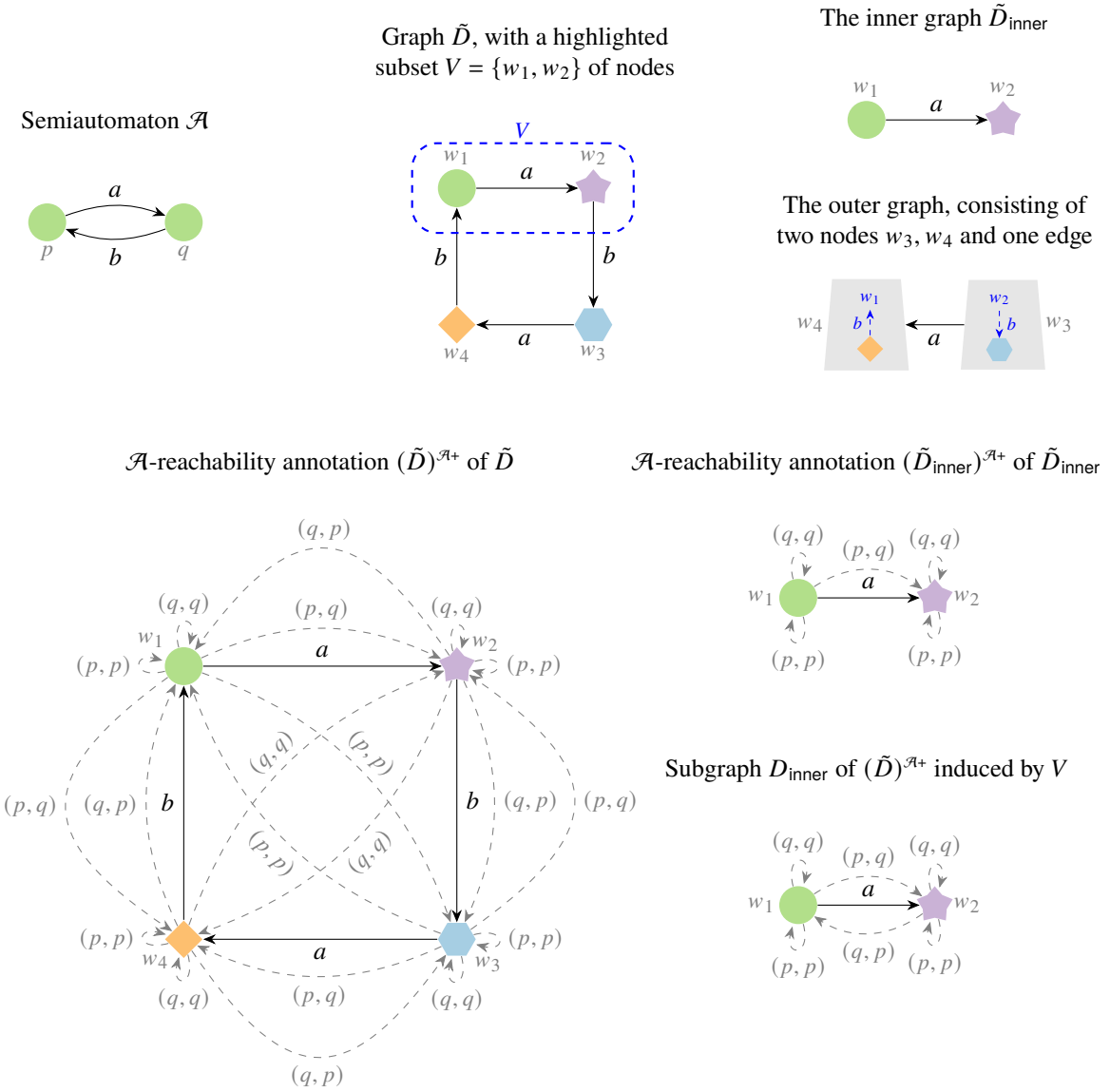


Figure 10.1: A semiautomaton $\mathcal{A}$, a graph $\tilde{D}$, its detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$, and corresponding graphs $(\tilde{D})^{\mathcal{A}+}$, $(\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}$ and D_{inner}. Note the long edge $(w_2, (\mathsf{Long}, q, p), w_1)$ in D_{inner}, which is not present in $(\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}$.

We define functions analogous to the ones from Chapter 4: $\mathsf{AttachGraphs}_{\mathcal{A}+}$ and $\mathsf{AttachProfile}_{\mathcal{A}+}$, and the three functions used directly in the reduction: $\mathsf{InnerGraphs}_{\mathcal{A}+}$, $\mathsf{OuterQueries}_{\mathcal{A}+}$, and $\mathsf{OuterProfiles}_{\mathcal{A}+}$.

We begin by considering an example that shows the main idea for handling long edges. Next, we define $\mathsf{AttachGraphs}_{\mathcal{A}+}$, prove some of its properties listed earlier, and define the function $\mathsf{AttachProfile}_{\mathcal{A}+}$. Finally, we define the remaining three functions and describe the reduction. We move some technical proofs to the end of the chapter.

Below we state the main proposition, which requires one more definition. For a semiautomaton $\mathcal{A}$, an *$\mathcal{A}$-reachability annotated modal profile* is a modal profile $\mathcal{S}$ such that $\mathsf{GlobalRepresentative}(\mathcal{S})$ is an $\mathcal{A}$-reachability annotated graph. Note that, because all graphs in $\mathbb{G}_{\mathcal{A}+}$ are $\mathcal{A}$-reachability annotated graphs, for a modal profile $\mathcal{S}$ that is *not* $\mathcal{A}$-reachability annotated, the problem trivializes: the set $\mathsf{Models}_{\mathbb{G}_{\mathcal{A}+}}(\mathcal{S})$ is empty, which means that $\mathcal{G} \models_{\mathrm{hom}}^{\mathbb{G}_{\mathcal{A}+},\tilde{\mathcal{S}}} \mathcal{H}$ holds for all $\mathcal{G}, \mathcal{H}$ and $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$ for all Γ_{N}.

Proposition 10.2. For a semiautomaton $\mathcal{A}$ and an instance $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}}), \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$, $\mathcal{G} \models_{\mathrm{hom}}^{\mathbb{G}_{\mathcal{A}+},\tilde{\mathcal{S}}} \mathcal{H}$ iff either $\mathcal{S}$ is not an $\mathcal{A}$-reachability annotated modal profile, or, for each $G \in \mathcal{G}$,

for each $D_{\mathsf{inner}} \in \mathsf{InnerGraphs}_{\mathcal{A}+}(G, \Gamma_{\mathrm{N}})$,
for each $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}_{\mathcal{A}+}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$,
$\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}_{\mathcal{A}+}}(\mathcal{S}_{\mathsf{outer}})$, where $\mathcal{H}_{\mathsf{outer}} = \mathsf{OuterQueries}_{\mathcal{A}+}(\mathcal{H}, D_{\mathsf{inner}}, \Gamma_{\mathcal{S}} \setminus \Gamma_{\mathrm{N}})$.

10.3 Attaching graphs

The example discussed below shows the main idea behind the constructions. Consider the semiautomaton $\mathcal{A}$ and graphs $\tilde{D}$ and H presented in Figure 10.2 on the following page, along with the V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$ of $\tilde{D}$. There is a homomorphism h from H to $(\tilde{D})^{\mathcal{A}+}$: the unique edge $(u, (\mathsf{Long}, p, q), v)$ in H is mapped to the long edge in $(\tilde{D})^{\mathcal{A}+}$ arising from the walk π visiting all nodes in $\tilde{D}$.

Consider the first node in π that is inside $\tilde{D}_{\mathsf{inner}}$ (v_4), and the preceding node in π (v_3); in the constructions, we will refer to them as x_{inner} and x_{outer}, respectively. Similarly, consider the last node in π that is inside $\tilde{D}_{\mathsf{inner}}$ (v_6), and the succeeding node in π (v_7); we will refer to them as y_{inner} and y_{outer}, respectively. The graph H_{outer} shown in Figure 10.2 is the subgraph of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ induced by nodes $h(u)$, x_{outer}, y_{outer}, and $h(v)$; that is, by $\{w_1, w_3, w_7, w_9\}$.

We claim that, intuitively, the graph H_{outer} contains enough information about long edges in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ to witness, when combined with D_{inner}, the existence of a homomorphism from H to $(\tilde{D})^{\mathcal{A}+}$. More precisely, we will define $\mathsf{AttachGraphs}_{\mathcal{A}+}$ in such a way that H maps homomorphically to $\mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$.

Let us first consider the graph $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, presented in Figure 10.2; to avoid introducing another notion, we treat edge labels a and (Short, a) interchangeably. While the graph itself does not admit a homomorphism from H – it does not contain a long edge from w_1 to w_9 – it contains enough information to reason that such a homomorphism exists in $(\tilde{D})^{\mathcal{A}+}$; this requires combining five edges: a long edge from w_1 to w_3, a short edge from w_3 to w_4, a long edge from w_4 to w_6, a short edge from w_6 to w_7, and a long edge from w_7 to w_9. This observation is the backbone of the constructions and claims presented below; we begin by defining a function $\mathsf{LongClosure}_{\mathcal{A}+}$, intuitively, recovering long edges such as the one from w_1 to w_9 in $\mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$.

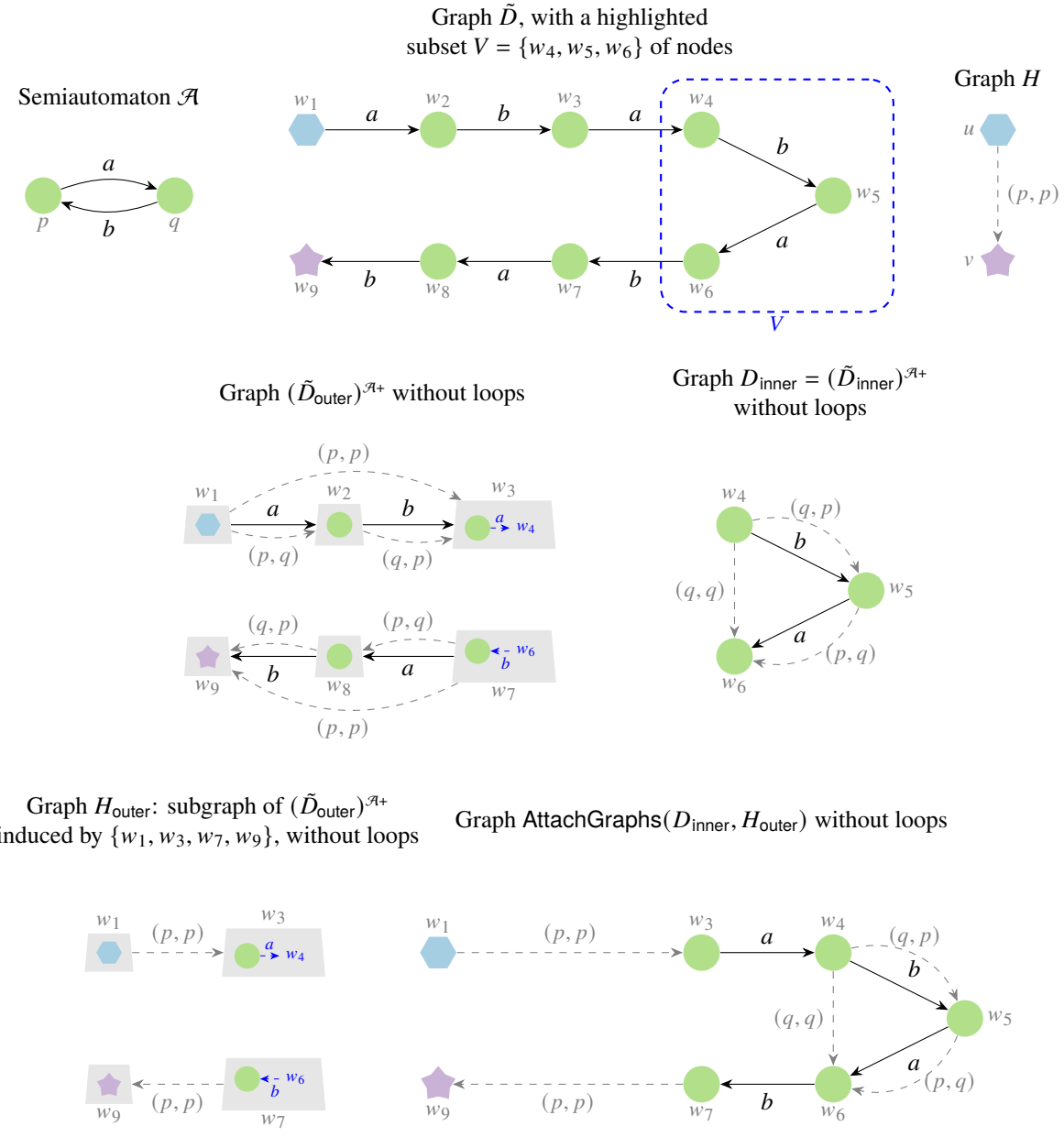


Figure 10.2: A semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ with a subset V of its nodes, and a graph H that maps homomorphically to $(\tilde{D})^{\mathcal{A}+}$, along with other graphs relevant to the discussed example.

10.3.1 Long closure

In the rest of this section we assume we work with a fixed semiautomaton $\mathcal{A}$, to avoid repeating the phrase "for every semiautomaton $\mathcal{A}$".

Consider a walk π in an $\mathcal{A}$-reachability annotated graph G. Let $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, a_i, v_i)$. For states p, q in $\mathcal{A}$, we say that π *maps* p to q if there exists a sequence $(q_0, q_1, \ldots, q_m)$ of states in $\mathcal{A}$ such that:

- $q_0 = p$;
- for each edge label a_i:
  - if $a_i = (\mathsf{Short}, a)$, then $\delta_{\mathcal{A}}(q_{i-1}, a) = q_i$;
  - otherwise, $a_i = (\mathsf{Long}, q_{i-1}, q_i)$;
- $q_m = q$.

Let $\mathsf{LongClosure}_{\mathcal{A}+}(G)$ denote the graph obtained from G by adding an edge $(u, (\mathsf{Long}, p, q), v)$ if there is a walk from u to v that maps p to q. Note that the function $\mathsf{LongClosure}_{\mathcal{A}+}$ is computable, because, if such a walk exists, there is one of length at most $|V_G| \cdot |V_{\mathcal{A}}|$; importantly, the function is computable even if we treat the semiautomaton $\mathcal{A}$ as a part of its input.

Fact 10.3. For every graph $\tilde{D}$, $\mathsf{LongClosure}_{\mathcal{A}+}(\tilde{D}^{\mathcal{A}+}) = \tilde{D}^{\mathcal{A}+}$.

Fact 10.4. For $\mathcal{A}$-reachability annotated graphs G, H:

- if $G \to H$ then $\mathsf{LongClosure}_{\mathcal{A}+}(G) \to \mathsf{LongClosure}_{\mathcal{A}+}(H)$;
- if G is a subgraph of H, then $\mathsf{LongClosure}_{\mathcal{A}+}(G)$ is a subgraph of $\mathsf{LongClosure}_{\mathcal{A}+}(H)$.

10.3.2 Attaching graphs

For two $\mathcal{A}$-reachability annotated graphs G, H, we define $\mathsf{AttachGraphs}_{\mathcal{A}+}(G, H)$ as the graph $\mathsf{LongClosure}_{\mathcal{A}+}(\mathsf{AttachGraphs}(G, H))$. Note that the function $\mathsf{AttachGraphs}_{\mathcal{A}+}$ is computable, even if $\mathcal{A}$ is treated as a part of its input.

Below we state three claims for a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$:

- $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}) = \tilde{D}^{\mathcal{A}+}$;
- if $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, then $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, H_{\mathsf{outer}}) \to \tilde{D}^{\mathcal{A}+}$;
- if $H \to \tilde{D}^{\mathcal{A}+}$, there is a graph H_{outer} with at most $|V_H| + 2|E_H|$ nodes such that $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

We prove the first two, and we move the proof of the third claim to the end of the chapter.

Claim 10.5. $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}) = \tilde{D}^{\mathcal{A}+}$.

Proof. Let $G = \mathsf{AttachGraphs}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+})$; we need to show that $\mathsf{LongClosure}_{\mathcal{A}+}(G)$ is equal to $\tilde{D}^{\mathcal{A}+}$. By the analogous property for $\mathsf{AttachGraphs}$ (Fact 4.2 on page 45), short edges in G and in $\tilde{D}^{\mathcal{A}+}$ are the same, so we focus on long edges.

Each long edge in $\mathsf{LongClosure}_{\mathcal{A}+}(G)$ is an edge in $\tilde{D}^{\mathcal{A}+}$: $\tilde{D}_{\mathsf{inner}}$ and $\tilde{D}_{\mathsf{outer}}$ are subgraphs of $\tilde{D}$, so G is a subgraph of $\tilde{D}^{\mathcal{A}+}$, by the construction of $\mathcal{A}$-reachability annotations; thus, $\mathsf{LongClosure}_{\mathcal{A}+}(G)$ is a subgraph of $\mathsf{LongClosure}_{\mathcal{A}+}(\tilde{D}^{\mathcal{A}+})$, which is equal to $\tilde{D}^{\mathcal{A}+}$ by Fact 10.3.

It remains to show that each long edge in $\tilde{D}^{\mathcal{A}+}$ is also a long edge in $\mathsf{LongClosure}_{\mathcal{A}+}(G)$. For each long edge $(u, (\mathsf{Long}, p, q), v)$ in $\tilde{D}^{\mathcal{A}+}$, there is an $\mathcal{A}_{p,q}$-walk from u to v in $\tilde{D}$, and because G and $\tilde{D}^{\mathcal{A}+}$ have the same short edges, there is a corresponding walk from u to v in G that traverses only short edges and maps p to q, so $(u, (\mathsf{Long}, p, q), v)$ is an edge in $\mathsf{LongClosure}_{\mathcal{A}+}(G)$. □

Claim 10.6. If $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ then $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, H_{\mathsf{outer}}) \to \tilde{D}^{\mathcal{A}+}$, for every graph H_{outer}.

Proof. The claim follows from the claim above, an analogous property for $\mathsf{AttachGraphs}$, and Fact 10.4 on the previous page. By the claim above, it is enough to show that $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, H_{\mathsf{outer}})$ maps homomorphically to $\mathsf{AttachGraphs}_{\mathcal{A}+}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+})$. By the analogous property for $\mathsf{AttachGraphs}$ (Fact 4.3 on page 45), $\mathsf{AttachGraphs}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, H_{\mathsf{outer}})$ maps homomorphically to $\mathsf{AttachGraphs}((\tilde{D}_{\mathsf{inner}})^{\mathcal{A}+}, (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+})$. Because $\mathsf{AttachGraphs}_{\mathcal{A}+}$ is defined as the composition of $\mathsf{LongClosure}_{\mathcal{A}+}$ and $\mathsf{AttachGraphs}$, by Fact 10.4 on the previous page we get the required homomorphism. □

Claim 10.7. For a graph H, if $H \to \tilde{D}^{\mathcal{A}+}$, then there is a graph H_{outer} with at most $|V_H| + 2|E_H|$ nodes such that $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

We move the proof to the end of this chapter. It uses the idea presented in the example discussed earlier: for each long edge in H, intuitively, we potentially add to H_{outer} two nodes, corresponding to the first time a witnessing walk enters $\tilde{D}_{\mathsf{inner}}$, and to the last time it leaves $\tilde{D}_{\mathsf{inner}}$.

10.3.3 Attaching profiles and recovering inner reachability

Below we define two functions; we prove that they are correctly defined at the end of this chapter.

Consider a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$. Let $\mathcal{S}_{\mathsf{outer}}$ be the shallow modal profile of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$. We define $\mathsf{AttachProfile}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$ as the shallow modal profile of $\tilde{D}^{\mathcal{A}+}$, and $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$ as the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

The proof that these functions are correctly defined will use the same idea as the example discussed earlier, but it is technically involved, because we work with a modal profile instead of a fixed graph, and we need to take into account walks that switch between $\tilde{D}_{\mathsf{inner}}$ and $\tilde{D}_{\mathsf{outer}}$ multiple times.

To compute these functions, it suffices to consider an arbitrary graph $\tilde{D}_{\mathsf{outer}}$ such that $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ satisfies $\mathcal{S}_{\mathsf{outer}}$, compute $\mathsf{AttachGraphs}(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, and calculate the shallow modal profile of the resulting graph, or the subgraph induced by V. Finding an appropriate graph $\tilde{D}_{\mathsf{outer}}$ is possible by solving the constructive satisfiability problem for $\mathbb{G}_{\mathcal{A}+}$. Alternatively, an algorithm computing the functions can be described based on the proof that they are correctly defined.

10.4 The reduction

It remains to define the functions $\mathsf{InnerGraphs}_{\mathcal{A}+}$, $\mathsf{OuterQueries}_{\mathcal{A}+}$ and $\mathsf{OuterProfiles}_{\mathcal{A}+}$. Recall that a graph G is *natural* if $V_G = \{1, \ldots, |V_G|\}$.

10.4.1 Inner graphs

For a semiautomaton $\mathcal{A}$, an $\mathcal{A}$-reachability annotated graph G, and a finite set $\Gamma_N \subseteq \mathbb{U}$, let $\mathsf{InnerGraphs}_{\mathcal{A}+}(G, \Gamma_N)$ be the set of all natural $\mathcal{A}$-reachability annotated graphs D_{inner} in which:

- there are at most $|V_G| + |\Gamma_N|$ nodes;
- all node labels are in $\Gamma_G \cup \Gamma_N$;

such that D_{inner} is Γ_N-unique and $G \to D_{\mathsf{inner}}$. The function $\mathsf{InnerGraphs}_{\mathcal{A}+}$ is computable, even if $\mathcal{A}$ is treated as a part of its input; note that edge labels are limited by the definition of an $\mathcal{A}$-reachability annotated graph.

Fact 10.8. For a semiautomaton $\mathcal{A}$, an $\mathcal{A}$-reachability annotated graph G, a finite set $\Gamma_N \subseteq \mathbb{U}$, a Γ_N-unique $\mathcal{A}$-reachability annotated graph D, and a homomorphism h from G to D, some graph in $\mathsf{InnerGraphs}_{\mathcal{A}+}(G, \Gamma_N)$ is isomorphic to the subgraph of D induced by the image of h extended with all nominal nodes in D.

10.4.2 Outer queries

For a semiautomaton $\mathcal{A}$, two $\mathcal{A}$-reachability annotated graphs H, D_{inner}, and a finite set $\Gamma \subseteq \mathbb{U}$, we define $\mathsf{OuterQueries}_{\mathcal{A}+}(H, D_{\mathsf{inner}}, \Gamma)$ as the set of all natural $\mathcal{A}$-reachability annotated graphs H_{outer} in which:

- there are at most $|V_H| + 2|E_H|$ nodes;
- all node labels are in $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma, \Sigma_{\mathcal{A}})$;

and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$. Note that the function $\mathsf{OuterQueries}_{\mathcal{A}+}$ is computable, even if $\mathcal{A}$ is treated as a part of its input.

Claim 10.9. For a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ such that $\Sigma_{\tilde{D}} \subseteq \Sigma_{\mathcal{A}}$, its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$, an $\mathcal{A}$-reachability annotated graph H, and a finite set $\Gamma \subseteq \mathbb{U}$ such that all node labels in $\tilde{D}_{\mathsf{outer}}$ are in $\mathsf{OuterLabels}(\tilde{D}_{\mathsf{inner}}, \Gamma, \Sigma_{\mathcal{A}})$, the following are equivalent:

- $H \to \tilde{D}^{\mathcal{A}+}$;
- $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ for some $H_{\mathsf{outer}} \in \mathsf{OuterQueries}_{\mathcal{A}+}(H, D_{\mathsf{inner}}, \Gamma)$;

where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

Proof. For the top-down implication, by Claim 10.7 on the facing page, there is a graph H_{outer} with at most $|V_H| + 2|E_H|$ nodes such that $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$. Since $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, H_{outer} has node labels in $\mathsf{OuterLabels}(D_{\mathsf{inner}}, \Gamma, \Sigma_{\mathcal{A}})$. Thus, H_{outer} is isomorphic to some graph in $\mathsf{OuterQueries}_{\mathcal{A}+}(H, D_{\mathsf{inner}}, \Gamma)$.

For the bottom-up implication, by the definition of $\mathsf{OuterQueries}_{\mathcal{A}+}(H, D_{\mathsf{inner}}, \Gamma)$ we have that $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, so by Claim 10.6 on the preceding page we get $H \to \tilde{D}^{\mathcal{A}+}$. □

We extend the function $\mathsf{OuterQueries}_{\mathcal{A}+}$ to sets of graphs. For a semiautomaton $\mathcal{A}$, an $\mathcal{A}$-reachability annotated graph D_{inner}, a set $\mathcal{H}$ of $\mathcal{A}$-reachability annotated graphs, and a finite set $\Gamma \subseteq \mathbb{U}$, we define $\mathsf{OuterQueries}_{\mathcal{A}+}(\mathcal{H}, D_{\mathsf{inner}}, \Gamma)$ as the union of $\mathsf{OuterQueries}_{\mathcal{A}+}(H, D_{\mathsf{inner}}, \Gamma)$ for all $H \in \mathcal{H}$.

Corollary 10.10. For a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ such that $\Sigma_{\tilde{D}} \subseteq \Sigma_{\mathcal{A}}$, its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$, a set $\mathcal{H}$ of $\mathcal{A}$-reachability annotated graphs, and a finite set $\Gamma \subseteq \mathbb{U}$ such that all node labels in $\tilde{D}_{\mathsf{outer}}$ are in $\mathsf{OuterLabels}(\tilde{D}_{\mathsf{inner}}, \Gamma, \Sigma_{\mathcal{A}})$, the following are equivalent:

- $H \to \tilde{D}^{\mathcal{A}+}$ for some $H \in \mathcal{H}$;
- $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ for some $H_{\mathsf{outer}} \in \mathsf{OuterQueries}_{\mathcal{A}+}(\mathcal{H}, D_{\mathsf{inner}}, \Gamma)$;

where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

10.4.3 Outer modal profiles

For an $\mathcal{A}$-reachability annotated graph D, the shortening of D is the graph $\tilde{D}$ obtained from D by removing long edges and replacing each edge label (Short, a) with a.

For a semiautomaton $\mathcal{A}$, an $\mathcal{A}$-reachability annotated graph D_{inner}, and a shallow $\mathcal{A}$-reachability annotated modal profile with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$, let $\mathsf{OuterProfiles}_{\mathcal{A}+}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$ be the set of all satisfiable shallow $\mathcal{A}$-reachability annotated modal profiles $\mathcal{S}_{\mathsf{outer}}$ such that:

- all node labels in $\mathcal{S}_{\mathsf{outer}}$ are in $\mathsf{OuterLabels}(\tilde{D}_{\mathsf{inner}}, \Gamma_{\mathcal{S}} \setminus \Gamma_{\mathrm{N}}, \Sigma_{\mathcal{A}})$;
- $D_{\mathsf{inner}} = \mathsf{RecoverInnerReachability}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$;
- $\mathcal{S} = \mathsf{AttachProfile}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$;

where $\tilde{D}_{\mathsf{inner}}$ is the shortening of D_{inner}. Note that the function $\mathsf{OuterProfiles}_{\mathcal{A}+}$ is computable, even if $\mathcal{A}$ is treated as a part of its input; we use the constructive satisfiability problem for $\mathbb{G}_{\mathcal{A}+}$ once again – this could be avoided, since including unsatisfiable modal profiles $\mathcal{S}_{\mathsf{outer}}$ would not influence the reduction, but it would also cause some technical issues with the definitions of $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}$ and $\mathsf{AttachProfile}_{\mathcal{A}+}$.

Fact 10.11. For a shallow $\mathcal{A}$-reachability annotated modal profile with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}})$, a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$ such that $\tilde{D}_{\mathsf{inner}}$ is Γ_{N}-unique, the following are equivalent:

- $\tilde{D}^{\mathcal{A}+} \models \tilde{\mathcal{S}}$;
- $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+} \models \mathcal{S}_{\mathsf{outer}}$ for some $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}_{\mathcal{A}+}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$;

where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

10.4.4 The reduction

From Fact 10.8 on the preceding page, Corollary 10.10, and Fact 10.11, we get the main proposition stated earlier.

Proposition 10.2. For a semiautomaton $\mathcal{A}$ and an instance $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\mathrm{N}}), \mathcal{G}, \mathcal{H}$ of the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$, $\mathcal{G} \models_{\mathrm{hom}}^{\mathbb{G}_{\mathcal{A}+}, \tilde{\mathcal{S}}} \mathcal{H}$ iff either $\mathcal{S}$ is not an $\mathcal{A}$-reachability annotated modal profile, or, for each $G \in \mathcal{G}$,

for each $D_{\mathsf{inner}} \in \mathsf{InnerGraphs}_{\mathcal{A}+}(G, \Gamma_{\mathrm{N}})$,
for each $\mathcal{S}_{\mathsf{outer}} \in \mathsf{OuterProfiles}_{\mathcal{A}+}(\tilde{\mathcal{S}}, D_{\mathsf{inner}})$,
$\mathcal{H}_{\mathsf{outer}} \twoheadrightarrow \mathsf{Models}_{\mathbb{G}_{\mathcal{A}+}}(\mathcal{S}_{\mathsf{outer}})$, where $\mathcal{H}_{\mathsf{outer}} = \mathsf{OuterQueries}_{\mathcal{A}+}(\mathcal{H}, D_{\mathsf{inner}}, \Gamma_{\mathcal{S}} \setminus \Gamma_{\mathrm{N}})$.

10.5 Remaining proofs

Proof of Claim 10.7

Recall the statement of the claim, for a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$.

Claim 10.7. For a graph H, if $H \to \tilde{D}^{\mathcal{A}+}$, then there is a graph H_{outer} with at most $|V_H| + 2|E_H|$ nodes such that $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

Let h be a homomorphism from H to $\tilde{D}^{\mathcal{A}+}$. For each long edge $e = (u, (\mathsf{Long}, p, q), v)$ in H, fix an $\mathcal{A}_{p,q}$-walk π_e from $h(u)$ to $h(v)$ in $\tilde{D}$. Let $\pi_e = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, a_i, v_i)$, and let $(q_0, \ldots, q_m)$ be the sequence of states in $\mathcal{A}$ such that $q_0 = p$ and $\delta_{\mathcal{A}}(q_{i-1}, a_i) = q_i$ for each $i = 1, \ldots, m$. If π_e contains some node in $\tilde{D}_{\mathsf{inner}}$, we define the following notions:

- *the first inner index* in π_e is the first index i such that v_i is in $\tilde{D}_{\mathsf{inner}}$; *the first inner node* is v_i, and *the first inner state* is q_i;
- *the last inner index* in π_e is the last index j such that v_j is in $\tilde{D}_{\mathsf{inner}}$; *the last inner node* is v_j, and *the last inner state* is q_j.

The first inner node and the last inner node correspond to nodes x_{inner} and y_{inner} in the example discussed earlier. We also need to be able to refer to the nodes corresponding to x_{outer} (preceding x_{inner}) and y_{outer} (succeeding y_{inner}). To this end, we use the words *preceding* and *succeeding*, in this context always referring to the walk π_e and the corresponding sequence $(q_0, \ldots, q_m)$ of states; for example, the state *preceding* the first inner state q_i in π_e is the state q_{i-1}, if $i > 0$.

Let V_{inner} be the set of nodes in $\tilde{D}_{\mathsf{inner}}$. Let V' be the union of V_{inner}, the image of h, and, for each long edge $e = (u, (\mathsf{Long}, p, q), v)$ in H:

- the node preceding the first inner node in π_e, if it exists;
- the node succeeding the last inner node in π_e, if it exists.

Let $V_{\mathsf{outer}} = V' \setminus V_{\mathsf{inner}}$, and let H_{outer} be the subgraph of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ induced by V_{outer}. By the construction, H_{outer} has at most $|V_H| + 2|E_H|$ nodes and $H_{\mathsf{outer}} \to (\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, because it is a subgraph of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$. These are two out of three conditions in the statement of the lemma; it remains to prove that $H \to \mathsf{AttachGraphs}_{\mathcal{A}+}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$, where D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V_{inner}. Recall that $\mathsf{AttachGraphs}_{\mathcal{A}+}$ is the composition of $\mathsf{LongClosure}_{\mathcal{A}+}$ and $\mathsf{AttachGraphs}$.

Let $D_H = \mathsf{AttachGraphs}(D_{\mathsf{inner}}, H_{\mathsf{outer}})$; note that the set of nodes in D_H is V', which means that it contains the image of h (the homomorphism from H to $\tilde{D}^{\mathcal{A}+}$). It remains to prove that $H \to \mathsf{LongClosure}_{\mathcal{A}+}(D_H)$; more precisely, we prove that h is the required homomorphism. That is, we need to show that, for each edge $e = (u, a, v)$ in H, there is an edge $(h(u), a, h(v))$ in $\mathsf{LongClosure}_{\mathcal{A}+}(D_H)$. For short edges, this follows directly from the construction, so let us focus on long edges. For $a = (\mathsf{Long}, p, q)$, we consider four cases:

- $h(u), h(v) \in V_{\mathsf{inner}}$: because h is a homomorphism from H to $\tilde{D}^{\mathcal{A}+}$, $(h(u), (\mathsf{Long}, p, q), h(v))$ is an edge in $\tilde{D}^{\mathcal{A}+}$, and because D_{inner} is the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V_{inner}, it is also an edge in D_{inner}, in D_H, and in $\mathsf{LongClosure}_{\mathcal{A}+}(D_H)$;
- $h(u), h(v) \in V_{\mathsf{outer}}$: if π_e does not contain a node in V_{inner}, then $(h(u), (\mathsf{Long}, p, q), h(v))$ is an edge

in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, so it is also an edge in its induced subgraph H_{outer}, in D_H, and in $\mathsf{LongClosure}_{\mathcal{A}+}(D_H)$; otherwise, let $x_{\mathsf{outer}}, x_{\mathsf{inner}}, y_{\mathsf{inner}}, y_{\mathsf{outer}}$ be, respectively, the node preceding the first inner node in π_e, the first inner node in π_e, the last inner node in π_e, and the node succeeding the last inner node in π_e; let $p_{\mathsf{outer}}, p_{\mathsf{inner}}, q_{\mathsf{inner}}, q_{\mathsf{outer}}$ be the corresponding states in $\mathcal{A}$; then, as witnessed by appropriate infixes of π_e, there are the following edges in D_H:
  - a long edge $(h(u), (\mathsf{Long}, p, p_{\mathsf{outer}}), x_{\mathsf{outer}})$ (in H_{outer});
  - a short edge $(x_{\mathsf{outer}}, (\mathsf{Short}, a_1), x_{\mathsf{inner}})$ for some a_1 such that $\delta_{\mathcal{A}}(p_{\mathsf{outer}}, a_1) = p_{\mathsf{inner}}$;
  - a long edge $(x_{\mathsf{inner}}, (\mathsf{Long}, p_{\mathsf{inner}}, q_{\mathsf{inner}}), y_{\mathsf{inner}})$ (in D_{inner});
  - a short edge $(y_{\mathsf{inner}}, (\mathsf{Short}, a_2), y_{\mathsf{outer}})$ for some a_2 such that $\delta_{\mathcal{A}}(q_{\mathsf{inner}}, a_2) = q_{\mathsf{outer}}$;
  - a long edge $(y_{\mathsf{outer}}, (\mathsf{Long}, q_{\mathsf{outer}}, q), h(v))$ (in H_{outer});

  the walk in D_H traversing the above edges maps p to q, so indeed $(h(u), (\mathsf{Long}, p, q), h(v))$ is a long edge in $\mathsf{LongClosure}_{\mathcal{A}+}(D_H)$;
- $h(u) \in V_{\mathsf{inner}}, h(v) \in V_{\mathsf{outer}}$: as above, replacing the first 3 edges with $(h(u), (\mathsf{Long}, p, q_{\mathsf{inner}}), y_{\mathsf{inner}})$;
- $h(u) \in V_{\mathsf{outer}}, h(v) \in V_{\mathsf{inner}}$: as above, replacing the last 3 edges with $(x_{\mathsf{inner}}, (\mathsf{Long}, p_{\mathsf{inner}}, q), h(v))$.

This finishes the proof.

Proof that $\mathsf{AttachProfile}_{\mathcal{A}+}$ and $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}$ are correctly defined

Recall the definitions. Consider a semiautomaton $\mathcal{A}$, a graph $\tilde{D}$ and its V-detachment $(\tilde{D}_{\mathsf{inner}}, \tilde{D}_{\mathsf{outer}})$. Let $\mathcal{S}_{\mathsf{outer}}$ be the shallow modal profile of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$. We defined $\mathsf{AttachProfile}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$ as the shallow modal profile of $\tilde{D}^{\mathcal{A}+}$, and $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$ as the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V.

To prove that $\mathsf{AttachProfile}_{\mathcal{A}+}$ is correctly defined, we need to characterize long edges in the shallow modal profile of $\tilde{D}^{\mathcal{A}+}$ in terms of $\tilde{D}_{\mathsf{inner}}$ and $\mathcal{S}_{\mathsf{outer}}$; short edges were already characterized this way in Chapter 4. To prove that $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}$ is correctly defined, we need to characterize long edges in the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V in terms of $\tilde{D}_{\mathsf{inner}}$ and $\mathcal{S}_{\mathsf{outer}}$; short edges are determined by $\tilde{D}_{\mathsf{inner}}$.

Consider a long edge $(u, (\mathsf{Long}, p, q), v)$ in $\tilde{D}^{\mathcal{A}+}$, and an $\mathcal{A}_{p,q}$-walk π from u to v in $\tilde{D}$. We can assume that π visits each node at most $|V_{\mathcal{A}}|$ times (by Claim 9.2 on page 153). The main idea is that π can be split into maximal infixes contained in $\tilde{D}_{\mathsf{inner}}$, maximal infixes contained in $\tilde{D}_{\mathsf{outer}}$, and single edges between them; we can reason about the infixes in $\tilde{D}_{\mathsf{inner}}$ directly, the maximal infixes contained in $\tilde{D}_{\mathsf{outer}}$ are represented by long edges in $\mathcal{S}_{\mathsf{outer}}$, and single edges between them are represented in the components $\mathsf{EIn}, \mathsf{EOut}$ in node labels in $\mathcal{S}_{\mathsf{outer}}$.

Let $\pi = [v_0, e_1, v_1, \ldots, e_m, v_m]$, with each $e_i = (v_{i-1}, a_i, v_i)$. Let $q_0, \ldots, q_m$ be the sequence of states in the corresponding run of $\mathcal{A}$; that is, $q_0 = p$, and $q_i = \delta_{\mathcal{A}}(q_{i-1}, a_i)$ for each $i = 1, \ldots, m$. Note that $q_m = q$, because π is an $\mathcal{A}_{p,q}$-walk.

Consider the sequence π' obtained from π by replacing each maximal infix $[v_i, \ldots, v_j]$ contained in $\tilde{D}_{\mathsf{outer}}$ (including trivial ones) with a single long edge with label $(\mathsf{Long}, q_i, q_j)$; note that it is an edge in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$. We treat edge labels a and (Short, a) as equal; then, π' is a walk in $\tilde{D}^{\mathcal{A}+}$. Let $\pi' = [v'_0, e'_1, v'_1, \ldots, e'_k, v'_k]$, with each $e'_i = (v'_{i-1}, a'_i, v'_i)$. Let $\vec{q}' = (q'_0, \ldots, q'_k)$ be the corresponding sequence of states in $\mathcal{A}$, such that:

- for each long edge e'_i, we have $a'_i = (\mathsf{Long}, q'_{i-1}, q'_i)$;
- for each short edge e'_i, we have $q'_i = \delta_{\mathcal{A}}(q'_{i-1}, a'_i)$.

Note that $v_0 = v'_0 = u$, $v_m = v'_k = v$, and $q_0 = q'_0 = p$, $q_m = q'_k = q$. Moreover, $k \leq 3 \cdot |V| \cdot |V_{\mathcal{A}}| + 2$, because each node in π is either in $\tilde{D}_{\mathsf{inner}}$ or in $\tilde{D}_{\mathsf{outer}}$, each node in $\tilde{D}_{\mathsf{inner}}$ is visited at most $|V_{\mathcal{A}}|$ times by π, so π consists of at most $|V| \cdot |V_{\mathcal{A}}|$ nodes in $\tilde{D}_{\mathsf{inner}}$ and at most $|V| \cdot |V_{\mathcal{A}}| + 1$ infixes in $\tilde{D}_{\mathsf{outer}}$, and π' is obtained from π by replacing each maximal infix contained in $\tilde{D}_{\mathsf{outer}}$ with a single edge (two nodes).

Below we list some properties of π' and $\vec{q}'$; later, to calculate $\mathsf{AttachProfile}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$, we consider all pairs of sequences that satisfy very similar properties. Recall that V is the set of nodes in $\tilde{D}_{\mathsf{inner}}$. For each v'_i, if $v'_i \notin V$, let τ_i be the 1-bisimilarity type of v'_i in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$, and let $(A_i, \mathsf{EIn}_i, \mathsf{EOut}_i)$ be the label of v'_i in $\tilde{D}_{\mathsf{outer}}$. Then, for each $i = 1, \ldots, k$:

- if $v'_{i-1} \in V$ and $v'_i \in V$, then e'_i is an edge in $\tilde{D}_{\mathsf{inner}}$, and $\delta_{\mathcal{A}}(q'_{i-1}, a'_i) = q'_i$;
- if $v'_{i-1} \in V$ and $v'_i \notin V$, then $(v'_{i-1}, a'_i) \in \mathsf{EIn}_i$, and $\delta_{\mathcal{A}}(q'_{i-1}, a'_i) = q'_i$;
- if $v'_{i-1} \notin V$ and $v'_i \in V$, then $(a'_i, v'_i) \in \mathsf{EOut}_{i-1}$, and $\delta_{\mathcal{A}}(q'_{i-1}, a'_i) = q'_i$;
- if $v'_{i-1} \notin V$ and $v'_i \notin V$, then e'_i is the result of replacing a maximal infix of π contained in $\tilde{D}_{\mathsf{outer}}$, so e'_i is an edge in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ and has label $(\mathsf{Long}, q'_{i-1}, q'_i)$, so $\tau_i|_0 \in \mathsf{Subtypes}(\tau_{i-1}, (\mathsf{Long}, q'_{i-1}, q'_i))$.

Moreover, the nodes outside V always occur in pairs: if $v'_i \notin V$, then exactly one of v'_{i-1}, v'_{i+1} is also not in V. This includes the first and the last node: if $v'_0 \notin V$, then $v'_1 \notin V$, and if $v'_k \notin V$, then $v'_{k-1} \notin V$.

Recall that $\mathcal{S}_{\mathsf{outer}}$ is a shallow modal profile of $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$; let $\mathcal{S}_{\mathsf{outer}} = (1, \mathcal{T})$. To obtain an algorithm calculating $\mathsf{AttachProfile}_{\mathcal{A}+}(\tilde{D}_{\mathsf{inner}}, \mathcal{S}_{\mathsf{outer}})$, we consider sequences similar to π' and $\vec{q}'$, intuitively, obtained by replacing each edge e'_i with its label a'_i, and replacing each node v'_i in $\tilde{D}_{\mathsf{outer}}$ with its 1-bisimilarity type τ_i. Formally, *witnessing sequences* are any sequences $(w_0, a_1, w_1, \ldots, a_m, w_m)$ and $(q_0, \ldots, q_m)$ such that $m \leq 3 \cdot |V| \cdot |V_{\mathcal{A}}| + 2$ and, for each $i = 1, \ldots, m$, either:

- $w_{i-1} \in V$, $w_i \in V$, and (w_{i-1}, a_i, w_i) is an edge in $\tilde{D}_{\mathsf{inner}}$;
- $w_{i-1} \in V$, $w_i \in \mathcal{T}$, and $(w_{i-1}, a_i) \in \mathsf{EIn}$, where $\mathsf{RootLabel}(w_i) = (A, \mathsf{EIn}, \mathsf{EOut})$;
- $w_{i-1} \in \mathcal{T}$, $w_i \in V$, and $(a_i, w_i) \in \mathsf{EOut}$, where $\mathsf{RootLabel}(w_{i-1}) = (A, \mathsf{EIn}, \mathsf{EOut})$;
- $w_{i-1} \in \mathcal{T}$, $w_i \in \mathcal{T}$, and $w_i|_0 \in \mathsf{Subtypes}(w_{i-1}, a_i)$, $a_i = (\mathsf{Long}, q_{i-1}, q_i)$.

In the first three cases, we additionally require that $\delta_{\mathcal{A}}(q_{i-1}, a_i) = q_i$. Moreover, the elements of $\mathcal{T}$ occur in pairs: if $w_i \in \mathcal{T}$, then exactly one of w_{i-1}, w_{i+1} is in $\mathcal{T}$.

We claim that, for a node v_0 in $\tilde{D}^{\mathcal{A}+}$, there is a long edge with label (Long, p, q) outgoing from v_0 to a node with label A iff there are witnessing sequences $(w_0, a_1, w_1, \ldots, a_m, w_m)$ and $(q_0, \ldots, q_m)$ such that $q_0 = p$, $q_m = q$, and:

- either $v_0 \in V$ and $w_0 = v_0$, or $v_0 \notin V$ and w_0 is the 1-bisimilarity type of v_0 in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$;
- either $w_m \in V$ and A is the label of w_m in $\tilde{D}_{\mathsf{inner}}$, or $w_m \in \mathcal{T}$ and $\mathsf{RootLabel}(w_m) = (A, \mathsf{EIn}, \mathsf{EOut})$ for some $\mathsf{EIn}, \mathsf{EOut}$.

The left-to-right implication is witnessed by the sequences π' and $\vec{q}'$ from the previous construction. For the right-to-left implication, beginning from the node v_0, for each $i = 1, \ldots, m$ we will iteratively define a node v_i and an $\mathcal{A}_{q_{i-1}, q_i}$-walk π_i from v_{i-1} to v_i in $\tilde{D}$; let us first briefly describe it. For each $w_i \in V$, we will set $v_i = w_i$. For a pair $w_{i-1}, w_i \in \mathcal{T}$, we will set v_{i-1} as some node in $\tilde{D}_{\mathsf{outer}}$ whose 1-bisimilarity type is w_{i-1}, and v_i as some node whose 0-bisimilarity type is $w_i|_0$ such that there is an appropriate walk between them. Formally, we define v_i and π_i iteratively as follows:

- if $w_{i-1} \in V$, $w_i \in V$, let $v_i = w_i$ and let $\pi_i = [v_{i-1}, (v_{i-1}, a_i, v_i), v_i]$;

- if $w_{i-1} \in V$, $w_i \in \mathcal{T}$, let v_i be an arbitrary node in $(\tilde{D}_{\mathsf{outer}})^{\mathcal{A}+}$ whose 1-bisimilarity type is w_i, and let $\pi_i = [v_{i-1}, (v_{i-1}, a_i, v_i), v_i]$;
- if $w_{i-1} \in \mathcal{T}$, $w_i \in V$, let $v_i = w_i$ and let $\pi_i = [v_{i-1}, (v_{i-1}, a_i, v_i), v_i]$;
- if $w_{i-1} \in \mathcal{T}$, $w_i \in \mathcal{T}$, let π_i be an arbitrary $\mathcal{A}_{q_{i-1},q_i}$-walk in $\tilde{D}_{\mathsf{outer}}$ from v_{i-1} to some node v_i whose 0-bisimilarity type is $w_i|_0$; such a walk exists, because, by the definition of witnessing sequences, $w_i|_0 \in \mathsf{Subtypes}(w_{i-1}, a_i)$, $a_i = (\mathsf{Long}, q_{i-1}, q_i)$.

Then, to obtain an $\mathcal{A}_{p,q}$-walk from u to a node with label A in $\tilde{D}$, it is enough to concatenate the walks $\pi_1, \ldots, \pi_m$.

This finishes the proof: the correspondence above shows that the modal profile of $\tilde{D}^{\mathcal{A}+}$ is determined by $\tilde{D}_{\mathsf{inner}}$ and $\mathcal{S}_{\mathsf{outer}}$, so the function $\mathsf{AttachProfile}_{\mathcal{A}+}$ is correctly defined. Similarly, we can construct the subgraph of $\tilde{D}^{\mathcal{A}+}$ induced by V by considering all long edges arising from witnessing sequences with $w_0, w_m \in V$, so $\mathsf{RecoverInnerReachability}_{\mathcal{A}+}$ is correctly defined as well.

10.6 Summary

We have shown that the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$ is decidable for every semiautomaton $\mathcal{A}$, even if $\mathcal{A}$ is treated as a part of the input. In the next chapter we use this result to prove the main result of the thesis.

Chapter 11

UCRPQ containment under modal constraints

In this chapter we prove the main result of the thesis.

Theorem 1.1. The problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox is decidable.

We achieve this by reducing the problem to the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$. Additionally, we summarize the whole algorithm and discuss its time complexity, including some optimizations.

11.1 Definitions

Some subsections below are marked as "recaps" – they contain definitions that are taken verbatim from earlier chapters.

11.1.1 Interpretations – recap

Recall that at the beginning of Chapter 2 we defined an infinite, computable set $\mathbb{U}$, which we use in the definitions to ensure that the corresponding objects can be encoded as inputs of decision problems.

Let $\mathsf{ConceptNames}$, $\mathsf{RoleNames}$, $\mathsf{IndividualNames}$ and $\mathsf{Variables}$ be disjoint subsets of $\mathbb{U}$ that are infinite and computable. In terms of first-order interpretations, concept names correspond to unary predicates, role names correspond to binary predicates, and individual names correspond to constants – functional predicates of arity zero.

An *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ consists of a non-empty set Δ, called the *domain*, and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps each concept name to a subset of Δ, each role name to a relation on Δ, and each individual name to an element of Δ. An interpretation is *finite* if Δ is finite.

We define the notions for potentially infinite interpretations, as this is standard in the literature, but we actually work exclusively with finite ones. For this reason, and to simplify the translation from interpretations to graphs, let us assume that domains of interpretations are subsets of $\mathbb{U}$. We acknowledge

that this assumption is non-standard – in particular, it precludes uncountable interpretations – but it does not affect the problems we consider.

11.1.2 Translating between interpretations and graphs – recap

By a *finite signature* we mean a triple $X = (X_{\mathrm{C}}, X_{\mathrm{R}}, X_{\mathrm{I}})$ of finite sets $X_{\mathrm{C}} \subseteq \mathsf{ConceptNames}$, $X_{\mathrm{R}} \subseteq \mathsf{RoleNames}$, and $X_{\mathrm{I}} \subseteq \mathsf{IndividualNames}$. An *$X$-graph* is a non-empty graph in which node labels are subsets of $X_{\mathrm{C}} \cup X_{\mathrm{I}}$ and edge labels are in X_{R}. An X-graph G *respects individual names* if, for each individual name $a \in X_{\mathrm{I}}$, there is exactly one node v in G such that $a \in \mathsf{Label}_G(v)$.

Interpretations to graphs

For a finite signature X and a finite interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, we define $\mathsf{InterpretationToGraph}_X(\mathcal{I})$ as the X-graph G with nodes Δ such that:

- for each role name $r \in X_{\mathrm{R}}$, there is an edge (u, r, v) in G iff $(u, v) \in r^{\mathcal{I}}$;
- for each concept name $A \in X_{\mathrm{C}}$, $A \in \mathsf{Label}_G(v)$ iff $v \in A^{\mathcal{I}}$;
- for each individual name $a \in X_{\mathrm{I}}$, $a \in \mathsf{Label}_G(v)$ iff $v = a^{\mathcal{I}}$.

Note that G respects individual names.

Graphs to interpretations

For a finite signature X and an X-graph G that respects individual names, let $\mathsf{GraphToInterpretation}_X(G)$ be an arbitrary fixed interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ such that $\Delta = V_G$ and:

- $r^{\mathcal{I}} = \{(u, v) : (u, r, v) \in E_G\}$ for each role name $r \in X_{\mathrm{R}}$;
- $A^{\mathcal{I}} = \{v \in V_G : A \in \mathsf{Label}_G(v)\}$ for each concept name $A \in X_{\mathrm{C}}$;
- $a^{\mathcal{I}}$ is the unique node v in G such that $a \in \mathsf{Label}_G(v)$ for each individual name $a \in X_{\mathrm{I}}$.

The interpretation of concept names, role names and individual names that do not occur in X can be arbitrary.

11.1.3 Automata

In this chapter, we implicitly assume that the alphabet of an automaton is a subset of $\mathsf{RoleNames}$.

Definitions – recap

A *nondeterministic finite automaton*, which we call simply an *automaton*, is a graph $\mathcal{A}$ with a distinguished *initial* node q_0 and a set of *final* nodes F. Automata are typically defined as machines, with an input, states, and possibly an output. To keep consistent with the standard terminology, we refer to nodes in an automaton as *states*, to edges as *transitions*, to walks as *runs*, and to $\Sigma_{\mathcal{A}}$ as the *alphabet*. A run *over a word* $\vec{a}$ is a run with edge label sequence $\vec{a}$.

The language *recognized* by an automaton $(\mathcal{A}, q_0, F)$ is the set of edge label sequences of runs in $\mathcal{A}$ that begin in q_0 and end in some state in F. The languages recognized by automata are called *regular languages*[1].

[1] In this chapter, this means regular languages over $\mathsf{RoleNames}$.

An automaton $(\mathcal{A}, q_0, F)$ is *deterministic* if, for each $a \in \Sigma_\mathcal{A}$, each state in $\mathcal{A}$ has exactly one outgoing transition with label a. In such a case, the graph $\mathcal{A}$ is called a *semiautomaton*. The *transition function* of a semiautomaton $\mathcal{A}$, denoted $\delta_\mathcal{A}$, is the function from $V_\mathcal{A} \times \Sigma_\mathcal{A}$ to $V_\mathcal{A}$ such that $\delta_\mathcal{A}(p, a) = q$ iff $(p, a, q) \in E_\mathcal{A}$. The transition function can be lifted to words over $\Sigma_\mathcal{A}$, with $\delta_\mathcal{A}(p, \vec{a}) = q$ iff q is the last state in the unique run from p over the word $\vec{a}$ in $\mathcal{A}$.

Automata over a finite signature

For a finite signature X, in an automaton $(\mathcal{A}, q_0, F)$ *over* X, the alphabet $\Sigma_\mathcal{A}$ is equal to X_R – the set of role names in the signature. The fact that we require $\Sigma_\mathcal{A} = X_\mathrm{R}$ instead of $\Sigma_\mathcal{A} \subseteq X_\mathrm{R}$ is related to the definition of $\mathbb{G}_{\mathcal{A}+}$, where we assumed that $\Sigma_G \subseteq \Sigma_\mathcal{A}$ for each $G^{\mathcal{A}+} \in \mathbb{G}_{\mathcal{A}+}$. As mentioned earlier, this assumption is made for convenience; without it, we would need to consider special cases for edge labels not in $\Sigma_\mathcal{A}$, or modify the automaton.

Interpretations of automata

For an interpretation $\mathcal{I} = (\Delta, \cdot^\mathcal{I})$, we extend the interpretation function $\cdot^\mathcal{I}$ to automata as follows. For an automaton M, let $M^\mathcal{I}$ denote the set of pairs $(x, y) \in \Delta \times \Delta$ such that there exist:

- a sequence of elements $x_0, x_1, \ldots, x_m$ in Δ, with $x_0 = x$ and $x_m = y$, and
- a sequence of role names $a_1, a_2, \ldots, a_m \in \mathsf{RoleNames}$,

such that:

- $(x_{i-1}, x_i) \in (a_i)^\mathcal{I}$ for each $i = 1, \ldots, m$;
- the word $(a_1, \ldots, a_m)$ belongs to the language recognized by M.

We state the below fact for deterministic automata, because we defined the graph $G^{\mathcal{A}+}$ for a semiautomaton $\mathcal{A}$, but it can be easily generalized to arbitrary automata.

Fact 11.1. For a finite signature X, a deterministic automaton $M = (\mathcal{A}, q_0, F)$ over X, a finite interpretation $\mathcal{I}$, and the graph $G = \mathsf{InterpretationToGraph}_X(\mathcal{I})$, the following conditions are equivalent:

- $(u, v) \in M^\mathcal{I}$;
- there is an edge $(u, (\mathsf{Long}, q_0, q), v)$ in $G^{\mathcal{A}+}$ for some $q \in F$.

Unifying semiautomata

For a set $\mathcal{M}$ of deterministic automata, we say that the automata in $\mathcal{M}$ *use a common semiautomaton* $\mathcal{A}$ if, for each $M \in \mathcal{M}$, $M = (\mathcal{A}, q_0, F)$ for some q_0, F.

Fact 11.2. There is a computable function mapping a set $\mathcal{M}$ of automata to a set $\mathcal{M}'$ of deterministic automata that use a common semiautomaton, with the alphabet being the union of alphabets of all automata in $\mathcal{M}$, such that for each automaton in $\mathcal{M}$ there is an automaton in $\mathcal{M}'$ that recognizes the same language.

11.1.4 Conjunctive regular path queries

A *term* is a variable or an individual name. A *concept atom* is a formula of the form $A(x)$, where A is a concept name and x is a term. A *regular path query* (RPQ) is a formula of the form $M(x, y)$, where M

is an automaton, and x, y are terms. An *equality atom* is a formula of the form $x = y$, where x, y are terms. An *atom* is a concept atom, a regular path query, or an equality atom.

A *conjunctive regular path query* (CRPQ) q consists of:

- a set Φ of atoms;
- a finite sequence $\vec{x}$ of variables, called *answer variables*.

For a CRPQ $q = (\vec{x}, \Phi)$, we denote by $\mathsf{Terms}(q)$ the set of all terms that occur in atoms in Φ, and by $\mathsf{Var}(q)$ all variables among them. The length of $\vec{x}$ is called the *arity* of q.

Remark. We do not allow role atoms $r(x, y)$ in CRPQs, because they can be expressed as RPQs. Equality atoms could also be expressed as RPQs, but we include them to improve readability of the reduction to Boolean queries. We also note that RPQs are typically defined using *regular expressions* rather than automata, but it is well known that they can be (computably) translated to each other, preserving the recognized language.

For an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ and a CRPQ $q = (\vec{x}, \Phi)$, a *variable assignment* is a function μ from $\mathsf{Var}(q)$ to Δ. Let us extend μ to individual names, with $\mu(a) = a^{\mathcal{I}}$ for each $a \in \mathsf{IndividualNames}$. The CRPQ q *evaluates to "true"* in $\mathcal{I}$ under μ if:

- for each concept atom $A(x)$ in Φ, $\mu(x) \in A^{\mathcal{I}}$;
- for each RPQ $M(x, y)$ in Φ, $(\mu(x), \mu(y)) \in M^{\mathcal{I}}$;
- for each equality atom $x = y$ in Φ, $\mu(x) = \mu(y)$.

Then, for $\vec{x} = (x_1, \ldots, x_k)$, the sequence $(\mu(x_1), \ldots, \mu(x_k))$ is called an *answer* to q in $\mathcal{I}$. We denote by $q(\mathcal{I})$ the set of all answers to q in $\mathcal{I}$.

A CRPQ is *Boolean* if it has arity 0 ($\vec{x}$ is the empty sequence). A Boolean CRPQ q is *satisfied* in an interpretation $\mathcal{I}$ if q evaluates to "true" in $\mathcal{I}$ under some variable assignment μ; equivalently, if $q(\mathcal{I})$ is the singleton of the empty sequence.

For a finite signature X, a CRPQ *over* X is a CRPQ in which all concept atoms of the form $A(x)$ for $A \in X_{\mathrm{C}}$, the alphabet of each automaton in each RPQ is X_{R}, and all terms that are individual names are in X_{I}. A CRPQ q *uses a common semiautomaton* $\mathcal{A}$ if all automata in RPQs in q are deterministic and use $\mathcal{A}$ as a common semiautomaton. By Fact 11.2 on the preceding page, every CRPQ can be rewritten into an equivalent CRPQ that uses a common semiautomaton.

Unions of CRPQs

A *union of CRPQs* (UCRPQ) is a set Q of CRPQs of equal arity k; we refer to k as the arity of Q. For an interpretation $\mathcal{I}$, the set of answers to Q in $\mathcal{I}$, denoted $Q(\mathcal{I})$, is the union of $q(\mathcal{I})$ for all $q \in Q$. A UCRPQ Q is *Boolean* if its arity is 0. A Boolean UCRPQ Q is *satisfied* in an interpretation $\mathcal{I}$ if $Q(\mathcal{I})$ is non-empty; that is, some $q \in Q$ is satisfied in $\mathcal{I}$.

For a finite signature X, a UCRPQ *over* X contains only CRPQs over X. For a set of UCRPQs, we say that they *use a common semiautomaton* $\mathcal{A}$ if, for each UCRPQ Q in the set, each CRPQ in Q uses $\mathcal{A}$ as a common semiautomaton. By Fact 11.2 on the previous page, every UCRPQ can be rewritten into an equivalent UCRPQ that uses a common semiautomaton.

Translating between Boolean (U)CRPQs and graphs

For a finite signature X and a semiautomaton $\mathcal{A}$ over X, an *$\mathcal{A}$-reachability annotated X-graph* is a graph in which node labels are subsets of $X_C \cup X_I$, and edge labels are of the form (Short, a) for some $a \in \Sigma_{\mathcal{A}}$, or (Long, p, q) for some states p, q in $\mathcal{A}$.

A graph G is a *quotient* of a CRPQ q if the set of nodes in G forms a partition of $\mathsf{Terms}(q)$; that is, the nodes in G are non-empty subsets of $\mathsf{Terms}(q)$ such that each term t is contained in exactly one subset, denoted by $[t]_G$.

For a finite signature X and a non-empty Boolean CRPQ q over X that uses a common semiautomaton $\mathcal{A}$, we define $\mathsf{CRPQToGraphs}_X(q)$ as the set of all $\mathcal{A}$-reachability annotated X-graphs G that are quotients of q such that:

- for each RPQ $M(x, y)$ in q, where $M = (\mathcal{A}, q_0, F)$, there is an edge $([x]_G, (\mathsf{Long}, q_0, q), [y]_G)$ in G for some $q \in F$;
- for each concept atom $A(x)$ in q, $A \in \mathsf{Label}_G([x]_G)$;
- for each individual name a in q, $a \in \mathsf{Label}_G([a]_G)$;
- for each equality atom $x = y$ in q, $[x]_G = [y]_G$.

Note that the function $\mathsf{CRPQToGraphs}_X$ is computable, as the set of partitions of $\mathsf{Terms}(q)$ is easily computable, and the graphs use only labels specified by the finite signature X and the semiautomaton $\mathcal{A}$.

For the empty CRPQ q, we define $\mathsf{CRPQToGraphs}_X(q)$ as the singleton of the empty graph.

Fact 11.3. For a finite signature X, a Boolean CRPQ q over X that uses a common semiautomaton $\mathcal{A}$, a finite interpretation $\mathcal{I}$ and an X-graph G that respects individual names:

- if q is satisfied in $\mathcal{I}$, $G_q \to (\mathsf{InterpretationToGraph}_X(\mathcal{I}))^{\mathcal{A}+}$ for some $G_q \in \mathsf{CRPQToGraphs}_X(q)$;
- if $G_q \to G^{\mathcal{A}+}$ for some $G_q \in \mathsf{CRPQToGraphs}_X(q)$, q is satisfied in $\mathsf{GraphToInterpretation}_X(G)$.

The above result naturally lifts to UCRPQs. For a finite signature X and a Boolean UCRPQ Q over X with a common semiautomaton $\mathcal{A}$, we define $\mathsf{UCRPQToGraphs}_X(Q)$ as the union of $\mathsf{CRPQToGraphs}_X(q)$ for all $q \in Q$.

Corollary 11.4. For a finite signature X, a Boolean UCRPQ Q over X that uses a common semiautomaton $\mathcal{A}$, a finite interpretation $\mathcal{I}$ and an X-graph G that respects individual names:

- if Q is satisfied in $\mathcal{I}$, $G_Q \to (\mathsf{InterpretationToGraph}_X(\mathcal{I}))^{\mathcal{A}+}$ for some $G_Q \in \mathsf{UCRPQToGraphs}_X(Q)$;
- if $G_Q \to G^{\mathcal{A}+}$ for some $G_Q \in \mathsf{UCRPQToGraphs}_X(Q)$, Q is satisfied in $\mathsf{GraphToInterpretation}_X(G)$.

11.1.5 The description logic $\mathcal{ALCO}_{\mathrm{reg}}$

Recall that the set of *concept descriptions* in description logics is defined as the smallest set that contains some *basic concept descriptions* and all formulas constructed inductively from other concept descriptions using a given set of *constructors*. In the description logic $\mathcal{ALCO}_{\mathrm{reg}}$, the basic concept descriptions and their interpretations are the same as in $\mathcal{ALCO}$, and are listed in Table 11.1, and the constructors are listed in Table 11.2.

Remark. We do not include the constructors $\exists r.D$ and $\forall r.D$ that are present in $\mathcal{ALC}$, because they can be expressed using the constructors $\exists M.D$ and $\forall M.D$.

Basic concept description C	**Interpretation** $C^{\mathcal{I}}$
$\bot$	$\emptyset$
$\top$	Δ
$\{a\}$	$\{a^{\mathcal{I}}\}$
A	$A^{\mathcal{I}}$

Table 11.1: Basic concept descriptions in $\mathcal{ALCO}_{\text{reg}}$ and their interpretations in $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, for every concept name A and individual name a.

Constructor of a concept description C	**Interpretation** $C^{\mathcal{I}}$
$\neg D$	$\Delta \setminus D^{\mathcal{I}}$
$D \sqcap E$	$D^{\mathcal{I}} \cap E^{\mathcal{I}}$
$D \sqcup E$	$D^{\mathcal{I}} \cup E^{\mathcal{I}}$
$\exists M.D$	$\{x \in \Delta : \exists_{y \in \Delta} (x, y) \in M^{\mathcal{I}} \ \wedge \ y \in D^{\mathcal{I}}\}$
$\forall M.D$	$\{x \in \Delta : \forall_{y \in \Delta} (x, y) \in M^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}}\}$

Table 11.2: Constructors in $\mathcal{ALCO}_{\text{reg}}$ and the interpretations of the resulting concept descriptions in $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, for all concept descriptions D, E and automata M.

Redundant constructions

Exactly as in the case of $\mathcal{ALC}$, some basic concept descriptions and constructors in $\mathcal{ALCO}_{\text{reg}}$ can be replaced with others, resulting in the same interpretations:

- $\top$ has the same interpretation as $\neg\bot$;
- $D \sqcap E$ has the same interpretation as $\neg((\neg D) \sqcup (\neg E))$;
- $\forall M.D$ has the same interpretation as $\neg(\exists M.(\neg D))$.

Thus, we can rewrite each concept description in $\mathcal{ALCO}_{\text{reg}}$ to use only basic concept descriptions $\bot, \{a\}, A$ and constructors $\neg D$, $D \sqcup E$ and $\exists M.D$, while preserving its interpretation. In the rest of this chapter we assume that all concept descriptions are of this form.

Terminological boxes – recap

A *terminological box* (TBox) in a description logic is a finite set of *general concept inclusions*: formulas of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ *satisfies* a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$.

For convenience, we also allow a TBox to contain *general concept equivalences*: formulas of the form $C \equiv D$, which are interpreted as two general concept inclusions, $C \sqsubseteq D$ and $D \sqsubseteq C$.

Shallow TBoxes

We call an $\mathcal{ALCO}_{\text{reg}}$ TBox *shallow* if it contains only:

- general concept inclusions of the form $A \sqsubseteq B$ for concept names A, B;
- general concept equivalences of the form $A \equiv C$ for a concept name A and a concept description C that uses at most one constructor; that is, C is of the form $\bot, \{a\}, B, \neg B, B_1 \sqcup B_2$ or $\exists M.B$, where

B, B_1, B_2 are concept names, a is an individual name, and M is an automaton.

For a concept description C, let $\mathsf{sub}(C)$ be the set of subformulas of C; that is, the set of concept descriptions defined recursively as follows:

- if C is a basic concept description, $\mathsf{sub}(C) = \{C\}$;
- if C is of the form $\neg D$ or $\exists M.D$, $\mathsf{sub}(C) = \{C\} \cup \mathsf{sub}(D)$;
- if C is of the form $D \sqcup E$, $\mathsf{sub}(C) = \{C\} \cup \mathsf{sub}(D) \cup \mathsf{sub}(E)$.

For an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$, let $\mathsf{sub}(\mathcal{T})$ denote the union of $\mathsf{sub}(C)$ for all concept descriptions appearing in $\mathcal{T}$.

Consider a finite signature X and an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$ *over* X: using only concept names and individual names in X, and only automata over X. Below we construct a shallow $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}'$ that, intuitively, imposes the same constraints as $\mathcal{T}$ on interpretations of concept names in X_{C}.

For each $C \in \mathsf{sub}(\mathcal{T})$, let A_C be an arbitrary concept name in $\mathsf{ConceptNames} \setminus X_{\text{C}}$, such that $A_C \neq A_D$ whenever $C \neq D$. Let $\mathcal{T}'$ be the shallow $\mathcal{ALCO}_{\text{reg}}$ TBox containing the following general concept inclusions and general concept equivalences:

- $A_C \sqsubseteq A_D$ for each general concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$;
- $A_C \equiv A_D$ for each general concept equivalence $C \equiv D$ in $\mathcal{T}$;
- for each concept description C in $\mathsf{sub}(\mathcal{T})$:
  - $A_C \equiv C$ if C is a basic concept description;
  - $A_C \equiv \neg A_D$ if C is of the form $\neg D$;
  - $A_C \equiv A_D \sqcup A_E$ if C is of the form $D \sqcup E$;
  - $A_C \equiv \exists M.A_D$ if C is of the form $\exists M.D$.

Two interpretations $\mathcal{I}, \mathcal{J}$ *agree on* X if they have equal domains and equal interpretations of all symbols (concept names, role names and individual names) in X.

Fact 11.5. For every interpretation $\mathcal{I}$, the following are equivalent:

- $\mathcal{I} \models \mathcal{T}$;
- there is an interpretation $\mathcal{I}'$ that satisfies $\mathcal{T}'$ such that $\mathcal{I}$ and $\mathcal{I}'$ agree on X.

TBoxes to modal profiles with nominals

An $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$ *uses a common semiautomaton* $\mathcal{A}$ if all automata appearing in concept descriptions in $\mathcal{T}$ are deterministic and use $\mathcal{A}$ as a common semiautomaton. By Fact 11.2 on page 179, each $\mathcal{ALCO}_{\text{reg}}$ TBox can be rewritten to use a common semiautomaton.

For a finite signature X and an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$ over X that uses a common semiautomaton $\mathcal{A}$, we define $\mathsf{TBoxToProfiles}_X(\mathcal{T})$ as the set of all shallow modal profiles with nominals $\tilde{\mathcal{S}} = (\mathcal{S}, \Gamma_{\text{N}})$ such that, for $\mathcal{S} = (1, \Theta)$ and $\hat{A}_\tau = \mathsf{RootLabel}(\tau)$ for each $\tau \in \Theta$:

- Θ is non-empty;
- for each $\tau \in \Theta$, $\mathsf{Representative}(\tau)$ is an $\mathcal{A}$-reachability annotated X-graph;
- for each individual name $a \in X_{\text{I}}$ there is exactly one $\tau \in \Theta$ with $a \in \hat{A}_\tau$;
- Γ_{N} is the set of node labels $\hat{A}_\tau$ for all $\tau \in \Theta$ such that $\hat{A}_\tau \cap X_{\text{I}} \neq \emptyset$;
- for each $\tau \in \Theta$, for each $A_C \sqsubseteq A_D$ in $\mathcal{T}$, if $A_C \in \hat{A}_\tau$ then $A_D \in \hat{A}_\tau$;

- for each $\tau \in \Theta$, for each $A_C \equiv C$ in $\mathcal{T}$:
  - if $C = \bot$, then $A_C \notin \hat{A}_\tau$;
  - if $C \in \mathsf{ConceptNames}$, then $A_C \in \hat{A}_\tau$ iff $C \in \hat{A}_\tau$;
  - if C is of the form $\{a\}$, then $A_C \in \hat{A}_\tau$ iff $a \in \hat{A}_\tau$;
  - if C is of the form $\neg A_D$, then $A_C \in \hat{A}_\tau$ iff $A_D \notin \hat{A}_\tau$;
  - if C is of the form $A_D \sqcup A_E$, then $A_C \in \hat{A}_\tau$ iff $A_D \in \hat{A}_\tau$ or $A_E \in \hat{A}_\tau$;
  - if C is of the form $\exists M.A_D$ for an automaton $M = (\mathcal{A}, q_0, F)$, then $A_C \in \hat{A}_\tau$ iff there is some $\tau' \in \mathsf{Subtypes}(\tau, (\mathsf{Long}, q_0, q))$ for some $q \in F$ such that $A_D \in \mathsf{RootLabel}(\tau')$.

Note that the function $\mathsf{TBoxToProfiles}$ is computable, because we can compute the set of all shallow modal profiles $\mathcal{S}$ that use only labels determined by the finite signature X and the semiautomaton $\mathcal{A}$ (by Fact 2.8 on page 30), consider all $\Gamma_N \subseteq \Gamma_{\mathcal{S}}$, and verify the conditions listed above.

Fact 11.6. For a finite signature X, an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$ over X that uses a common semiautomaton $\mathcal{A}$, a finite interpretation $\mathcal{I}$ and a graph G:

- if $\mathcal{I} \models \mathcal{T}$, then $(\mathsf{InterpretationToGraph}_X(\mathcal{I}))^{\mathcal{A}+} \models \tilde{\mathcal{S}}$ for some $\tilde{\mathcal{S}} \in \mathsf{TBoxToProfiles}_X(\mathcal{T})$;
- if $G^{\mathcal{A}+} \models \tilde{\mathcal{S}}$ for some $\tilde{\mathcal{S}} \in \mathsf{TBoxToProfiles}_X(\mathcal{T})$, then G is an X-graph that respects individual names and $\mathsf{GraphToInterpretation}_X(G) \models \mathcal{T}$.

11.1.6 UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox

For UCRPQs Q_1, Q_2 and an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$, we say that Q_1 is contained in Q_2 *modulo* $\mathcal{T}$, denoted $Q_1 \subseteq^{\mathcal{T}} Q_2$, if $Q_1(\mathcal{I}) \subseteq Q_2(\mathcal{I})$ for every finite interpretation $\mathcal{I}$ that satisfies $\mathcal{T}$.

UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox

Input: An $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$ and two UCRPQs Q_1, Q_2 of equal arity.

Question: Does $Q_1 \subseteq^{\mathcal{T}} Q_2$?

Without loss of generality, we can assume that $\mathcal{T}$ is a shallow $\mathcal{ALCO}_{\text{reg}}$ TBox (by Fact 11.5 on the preceding page), and that Q_1, Q_2 and $\mathcal{T}$ use a common semiautomaton $\mathcal{A}$ (by Fact 11.2 on page 179). Below we show that we can additionally assume that Q_1 and Q_2 are Boolean UCRPQs.

Reduction to Boolean UCRPQs

Consider an instance of the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox: two UCRPQs Q_1, Q_2 of equal arity r, and an $\mathcal{ALCO}_{\text{reg}}$ TBox $\mathcal{T}$. The reduction is exactly the same as in Chapter 3: we replace answer variables with individual names.

Let X_{I} be the set of all individual names that occur in at least one of Q_1, Q_2 and $\mathcal{T}$. Let $\vec{a} = (a_1, \ldots, a_r)$ be an arbitrary sequence of length r of different individual names in $\mathsf{IndividualNames} \setminus X_{\text{I}}$.

For each CRPQ $q = ((x_1, \ldots, x_r), \Phi)$ in Q_1 or in Q_2, intuitively, we want to replace each answer variable x_i by the individual name a_i. Formally, because a variable might occur multiple times as an answer variable, we need to be more careful. Let $f(i)$ denote the smallest index j such that x_j is the same variable as x_i. We define q' as the Boolean CRPQ obtained from q by:

- replacing each occurrence of x_i in atoms in Φ with $a_{f(i)}$, for each $i \in \{1, \ldots, r\}$;

- adding to Φ an atom $a_i = a_j$ for each $i, j \in \{1, \dots, r\}$ such that x_i is the same variable as x_j;
- replacing the sequence of answer variables with the empty sequence.

Let Q_1' denote the Boolean UCRPQ consisting of all q' obtained as described above for $q \in Q_1$, and let Q_2' denote the Boolean UCRPQ consisting of all q' for $q \in Q_2$.

Fact 11.7. The answer to the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox is the same for $\mathcal{T}, Q_1, Q_2$ and for $\mathcal{T}, Q_1', Q_2'$.

It is easy to compute such queries Q_1', Q_2', given Q_1, Q_2 and $\mathcal{T}$. Thus, when solving the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox, we can assume that the queries are Boolean, without loss of generality.

11.2 The reduction to homomorphism entailment

Recall that our goal is to prove the following theorem.

Theorem 1.1. The problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox is decidable.

We reduce the problem to the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$, for an appropriate semiautomaton $\mathcal{A}$.

Consider an instance $\mathcal{T}, Q_1, Q_2$ of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox. Without loss of generality we can assume that $\mathcal{T}$ is a shallow TBox, Q_1, Q_2 are Boolean UCRPQs, and all Q_1, Q_2 and $\mathcal{T}$ use a common semiautomaton $\mathcal{A}$.

Let X be the finite signature consisting of all concept names, role names and individual names that occur in at least one of Q_1, Q_2 or $\mathcal{T}$. Note that, since they use a common semiautomaton $\mathcal{A}$, $X_R = \Sigma_{\mathcal{A}}$.

From Corollary 11.4 on page 181 and Fact 11.6 on the preceding page we get the following.

Proposition 11.8. The following are equivalent:
- $Q_1 \subseteq^{\mathcal{T}} Q_2$;
- $\mathsf{UCRPQToGraphs}_X(Q_1) \models_{\mathrm{hom}}^{\mathbb{G}_{\mathcal{A}+}, \tilde{\mathcal{S}}} \mathsf{UCRPQToGraphs}_X(Q_2)$ for each $\tilde{\mathcal{S}} \in \mathsf{TBoxToProfiles}_X(\mathcal{T})$.

This finishes the proof of the main result of this thesis.

11.3 Time complexity

In this section we discuss the time complexity of the whole algorithm solving the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox. We begin by summarizing the algorithm; for each reduction we point out some constructions or properties relevant to this section.

A summary of the algorithm

1. The reduction of the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox to the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$ (this chapter).

- We convert nondeterministic finite automata to deterministic ones.
- We iterate over all shallow modal profiles for given sets of node and edge labels.

2. The reduction of the homomorphism entailment problem for $\mathbb{G}_{\mathcal{A}+}$ to the homomorphism coverage problem for $\mathbb{G}_{\mathcal{A}+}$ (Chapter 10), based on detachments of graphs.
   - The crucial property is that the shallow modal profile of the whole graph ($\mathcal{S}$) is determined by the inner graph (D_{inner}) and the shallow modal profile of the outer graph ($\mathcal{S}_{\mathsf{outer}}$).
   - We iterate over all graphs D_{inner} and profiles $\mathcal{S}_{\mathsf{outer}}$, and determine the resulting profile $\mathcal{S}$.
3. The reduction of the homomorphism coverage problem to the minimal downsets problem (Chapter 5).
   - We iterate over all modal profiles of a given depth, for given sets of node and edge labels.
4. The reduction of the minimal downsets problem for $\mathbb{G}_{\mathcal{A}+}$ to the same problem for $\mathbb{G}_{\oplus}$ (Chapter 9).
   - We define a trace-based graph transformation, based on the permutation semiautomaton.
   - The length of the considered traces is twice the number of states of the original semiautomaton.
5. The solution of the minimal downsets problem for $\mathbb{G}_{\oplus}$ (Chapter 8).
   - We consider all doubly-ranked annotations of the modal profile.
   - We prove that the downsets of the ranked reachability-annotated global representatives of modal profiles are minimal.
   - We solve the satisfiability problem for $\mathbb{G}_{\circledast}$.

The time complexity of the presented algorithm is non-elementary, because of the number of modal profiles of a given depth for fixed sets of node labels and edge labels. In (Gutiérrez-Basulto et al. 2022, 2024) we described algorithms solving the problem of UCRPQ containment modulo $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ TBoxes with an optimal – doubly exponential – time complexity. In this section we discuss how the techniques presented in this thesis can be optimized, to correspond more closely to the approach used in these papers. The plan to optimize the algorithm is as follows.

- Replace modal profiles with *homomorphism profiles* to achieve an elementary time complexity.
- Use a more succinct representation of (sets of) profiles, and avoid iterating over all profiles by describing constructions tailored to these representations.
- Avoid an exponential blow-up caused by the trace-based graph transformation with trace lengths dependent on the number of states of the semiautomaton.

11.3.1 Homomorphism profiles

The definition

The following definition is crucial for optimizing the algorithm. For a positive integer n, the *n-node homomorphism type* of a node u in a graph G, denoted $\mathsf{HomType}^n_G(u)$, is the set of all trees with at most n nodes that admit a homomorphism to G mapping the root to u. The *homomorphism profile* of a graph consists of a positive integer n, called its *node-size*, and the set of n-node homomorphism types of all nodes in G.

Properties

We claim that:

- shallow modal profiles can be represented using homomorphism profiles;
- the number of different n-node homomorphism types is an elementary (doubly exponential) function of n and the number of node labels and edge labels;
- intuitively, to compute the minimal downsets of size at most n, it suffices to consider homomorphism profiles of node-size $2n$ instead of modal profiles of depth n.

Let us discuss these claims in slightly more detail.

The first claim, intuitively, follows from the fact that the bisimilarity type of a tree of depth 1 is uniquely determined by its homomorphism type. The second claim follows from the fact that, even up to isomorphism type, there are exponentially many trees with at most n nodes for a given n and given sets of node labels and edge labels. Proving the third claim would require going through the details of the proofs, so here we just mention a high-level explanation and the reason for the node-size $2n$ instead of n; later we also state an analogue of the One-Step Bisimulation Lemma. As for the high-level explanation, one could argue that it is even more natural to consider homomorphism profiles instead of modal profiles, because downsets are defined using homomorphism types of subgraphs. The number $2n$ comes from the construction that uses the lowest common ancestors in a tree (Claim 8.27 on page 130): we use the fact that, for a tree and a set V of its nodes, the set that extends V with the lowest common ancestors of all pairs of nodes in V has size at most $2|V| - 1$.

A characterization through rooted subtrees

To be able to talk about n-node homomorphism types on examples, and to relate them to n-bisimilarity types, we present an informal alternative characterization; we formalize it later.

A *rooted homomorphism* from a tree T_1 to a tree T_2 is a homomorphism from T_1 to T_2 that maps the root of T_1 to the root of T_2. A *rooted subtree* of a tree T is a connected subgraph of T that contains the root. For a positive integer n, we are interested in the following preorder on trees: $T_1 \preccurlyeq_n T_2$ iff every rooted subtree of T_1 with at most n nodes admits a rooted homomorphism to T_2 (equivalently: to some rooted subtree of T_2 with at most n nodes).

Recall that, for a positive integer n and a node u in a graph G, $\mathsf{HomType}^n_G(u)$ is the set of all trees with at most n nodes that admit a homomorphism to G mapping the root to u. It is easy to see that $T_1 \preccurlyeq_n T_2$ iff $\mathsf{HomType}^n_{T_1}(\mathsf{Root}(T_1)) \subseteq \mathsf{HomType}^n_{T_2}(\mathsf{Root}(T_2))$. Consequently, the root of T_1 has the same n-node homomorphism type as the root of T_2 iff $T_1 \preccurlyeq_n T_2$ and $T_2 \preccurlyeq_n T_1$. Informally, one can think of $\mathsf{HomType}^n_G(u)$ as the downward closure of the set of rooted subtrees of $\mathsf{Unravel}(n, G, u)$ with at most n nodes.

An example

Below we list the rooted subtrees of trees T and T' in Figure 11.1, by providing the set of nodes contained in each subtree. We list the sets of nodes inducing rooted subtrees:

- with 1 node: $\{u_0\}$ in T and $\{v_0\}$ in T';
- with 2 nodes: $\{u_0, u_1\}$ in T, $\{v_0, v_1\}$ and $\{v_0, v_3\}$ in T';
- with 3 nodes: $\{u_0, u_1, u_2\}$ and $\{u_0, u_1, u_3\}$ in T; $\{v_0, v_1, v_2\}$, $\{v_0, v_3, v_4\}$ and $\{v_0, v_1, v_3\}$ in T';
- with 4 nodes: the whole tree T; $\{v_0, v_1, v_2, v_3\}$ and $\{v_0, v_1, v_3, v_4\}$ in T'.

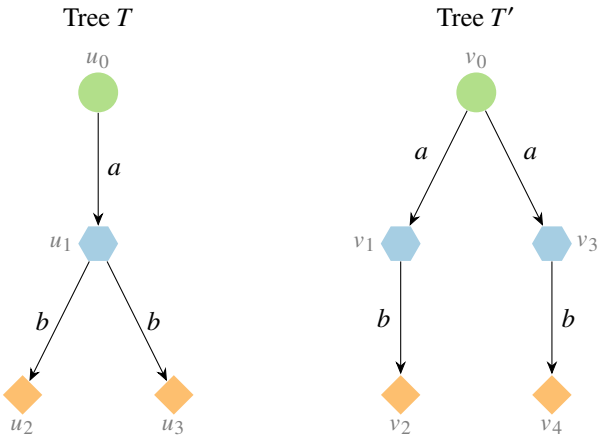


Figure 11.1: Examples of trees.

Each rooted subtree with at most 3 nodes in one of the trees admits a rooted homomorphism to the other tree. This does not hold for rooted subtrees with 4 nodes: the whole tree T does not admit a rooted homomorphism to T', so $\mathsf{HomType}^3_T(u_0) = \mathsf{HomType}^3_{T'}(v_0)$, but $\mathsf{HomType}^4_T(u_0) \neq \mathsf{HomType}^4_{T'}(v_0)$.

A recursive characterization

Note that, because n-node homomorphism types are defined as sets of trees, we can consider their union. It can be shown that $\mathsf{HomType}^n_G(u)$ determines and is uniquely determined by:

- the label of u in G, and
- if $n \geq 2$, the function mapping each edge label a to the union of $(n-1)$-node homomorphism types of all successors of u connected by an edge with label a.

We note that the proof of the above relation is not trivial; intuitively, the function described above contains only information about *single-branch* rooted subtrees with at most n nodes, in which the root has at most one child, and the claim is that it determines (and is determined by) the appropriate information about all rooted subtrees with at most n nodes.

This gives a recursive characterization of n-node homomorphism types, similar to the one for n-bisimilarity types, and an analogue of the One-Step Bisimulation Lemma. Below we provide the statement of this lemma, without proof, and assuming that notions such as $\mathsf{RootLabel}(\tau)$, $\mathsf{Subtypes}(\tau, a)$ and $\tau|_k$ are defined for an n-node homomorphism type τ, an edge label a, and a positive integer $k < n$.

Lemma 11.9. For every homomorphism profile $\mathcal{S} = (n, \mathcal{T})$, every graph G, and every function $\mathsf{SupposedType} : V_G \to \mathcal{T}$, the following conditions are equivalent.

- For each node u in G, $\mathsf{HomType}^n_G(u) = \mathsf{SupposedType}(u)$.
- For each node u in G, with $\tau = \mathsf{SupposedType}(u)$:
  - $\mathsf{Label}_G(u) = \mathsf{RootLabel}(\tau)$, and
  - for each edge label a, $\mathsf{Subtypes}(\tau, a) = \bigcup_{v \in \mathsf{Succ}_G(u,a)} \mathsf{SupposedType}(v)|_{n-1}$.

A connection to isomorphism types and bisimilarity types

For rigorous, inductive proofs of the properties of homomorphism profiles, we would use a different definition of n-node homomorphism types, more similar to the definition of n-bisimilarity types. Recall the informal characterizations of isomorphism types, bisimilarity types, and homomorphism types of trees from Chapter 2.

For a tree T, we define $\mathsf{IsoType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the **multiset** $\wr \mathsf{IsoType}(T') : T' \in \mathsf{Subtrees}(T, a) \wr$.

Fact 11.10. Two trees T_1, T_2 are isomorphic iff $\mathsf{IsoType}(T_1) = \mathsf{IsoType}(T_2)$.

For a tree T, we define $\mathsf{BisimType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the **set** $\{\mathsf{BisimType}(T') : T' \in \mathsf{Subtrees}(T, a)\}$.

Fact 11.11. Two trees T_1, T_2 are bisimilar iff $\mathsf{BisimType}(T_1) = \mathsf{BisimType}(T_2)$.

For a tree T, we define $\mathsf{RootedHomType}(T)$ recursively as the pair (A, f), where:

- $A = \mathsf{RootLabel}(T)$;
- for each edge label a, $f(a)$ is the ***rooted downward closure*** of the set

$$\{\mathsf{RootedHomType}(T') : T' \in \mathsf{Subtrees}(T, a)\}.$$

The term *rooted downward closure* is used above in the context of the partial order $\preccurlyeq$ on the image of $\mathsf{RootedHomType}$ based on rooted homomorphisms: $\mathsf{RootedHomType}(T_1) \preccurlyeq \mathsf{RootedHomType}(T_2)$ iff there is a rooted homomorphism from T_1 to T_2.

Fact 11.12. For trees T_1, T_2, $\mathsf{RootedHomType}(T_1) = \mathsf{RootedHomType}(T_2)$ iff T_1 and T_2 are homomorphically equivalent.

A different definition of the n-node homomorphism type of a node u in a graph G would be the rooted downward closure of the set consisting of $\mathsf{RootedHomType}(T)$ for all rooted subtrees T of $\mathsf{Unravel}(n, G, u)$ with at most n nodes.

Replacing modal profiles with homomorphism profiles

Replacing modal profiles with homomorphism profiles allows us to obtain an elementary time complexity of the overall algorithm, but it is still very high: quadruply exponential, while the lower bound obtained in (Gogacz et al. 2020) is doubly exponential, for the finite query entailment problem for $\mathcal{ALC}$ and CQs with transitive closure. In (Gutiérrez-Basulto et al. 2022, 2024) we achieved the optimal, doubly exponential time complexity for the problem of UCRPQ containment modulo $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ TBoxes. Below we discuss the two main optimizations used in these papers.

11.3.2 Succinct representations of sets of profiles

Modal profiles and homomorphism profiles are very verbose, as they specify exactly the set of types of nodes in a graph. In several cases we iterate over all relevant profiles, which impacts the time

complexity of the algorithm: the number of "shallow" homomorphism profiles (of node-size 2) for given sets of node labels and edge labels is triply exponential, so to achieve a doubly exponential time complexity we need to avoid iterating through all of them.

Essentially, the idea is to treat a single profile as a representative of all its "subprofiles" (profiles consisting of a subset of its types of nodes). Then, all considered problems need to be adjusted to treat the profile given in the input as an "upper bound" on all relevant profiles; this is similar to the $\preccurlyeq$-bounded satisfiability problem we considered in Section 7.6.2 on page 106. The fact that solving such problems is enough to solve the original problem relies on the details of the algorithm: ultimately, we check if some graph maps homomorphically to the ranked reachability-annotated global representative of a profile, so, intuitively, considering "smaller" profiles is better.

Below we mention some additional technical details.

- Strictly speaking, it is not enough to consider only "upper bounds" on the profiles; in the reduction that involves detachments of graphs, we need to ensure that the outer graph contains nodes of some types, so we also need to consider some "lower bounds", corresponding to an *assertion box* (ABox) in description logics.
- We need to consider the doubly-ranked annotations of types of nodes in early stages of the algorithm.
- Writing (and reading) rigorous proofs would be much more tedious in the case of the optimized algorithm, because for each instance where we iterate over all relevant modal profiles in this thesis, we would need to construct appropriate "upper bounds" on the relevant profiles, prove that they are correct, and that there are not too many of them (at most doubly exponentially many).

11.3.3 Optimizing the trace-based graph transformation

In the translation from the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\mathrm{reg}}$ TBox to the homomorphism entailment problem, we convert nondeterministic finite automata to deterministic ones, which might cause an exponential blow-up in the number of states. This has a significant impact on the time complexity of the algorithm, because the trace-based graph transformation used in Chapter 9 works with traces of length dependent on the number of states of the semiautomaton (Claim 9.9 on page 159), which in turn influences the depth (or node-size) of the profiles considered in the minimal downsets problem for ranked graphs.

A natural solution to this problem would be to work with nondeterministic automata; however, in the construction of the permutation semiautomaton, we rely on the determinism of the (semi)automata, and it is not clear if this can be avoided. In Chapter 9 we discussed some connections between our approach and the complexity results obtained by Bojańczyk (2009); it might be worth investigating the result obtained by Bojańczyk and Parys (2010), which improves the result in (Bojańczyk 2009) by considering nondeterministic automata instead of deterministic ones.

In the published papers we managed to optimize the algorithm differently: intuitively, we use the fact that we are ultimately interested in tree-shaped queries, and while the considered trace-based graph transformation makes the depth of the relevant tree-shaped queries exponential, their branching is very limited – there are polynomially many branching nodes. We use this fact to optimize the algorithm, but the resulting constructions are quite complicated. Moreover, it is not clear if this optimization can be

implemented while preserving the high-level reduction from the minimal downsets problem for $\mathbb{G}_{\mathcal{A}+}$ to the minimal downsets problem for $\mathbb{G}_{\oplus}$; in the papers we work with UCRPQs throughout the whole algorithm, without this reduction.

11.3.4 Other optimizations

To use the above ideas to achieve a doubly exponential time complexity, a lot of additional details need to be taken care of; below we discuss two most important ones.

In the reduction that involves detachments of graphs, in (Gutiérrez-Basulto et al. 2024) we make additional assumptions about the shape of the whole graph, and use tree automata techniques to achieve a doubly exponential time complexity. It is worth noting that these additional assumptions do not hold when the constraints are expressed using the description logic $\mathcal{ALCQI}$.

We also need to optimize the algorithm solving the satisfiability problem; in (Gutiérrez-Basulto et al. 2022) we use a procedure based on calculating the greatest fixed point, very similar to the one discussed in Section 7.6.3 on page 113.

Chapter 12

Conclusions

We proved the decidability of the problem of UCRPQ containment modulo an $\mathcal{ALCO}_{\text{reg}}$ TBox, which appears to be the first decidability result for full UCRPQs in this context. The result has significance in database theory and in the field of knowledge representation and reasoning.

We combined and generalized several techniques known in these fields, and introduced some new ones. Most notably, we generalized the SCC-based techniques to ranked graphs, and introduced the permutation semiautomaton, trace-based graph transformations, and ravelling constructions along with the Last Edge Appearance Record.

The most interesting areas of future research involve extending the decidability results to the description logic $\mathcal{ALCQI}$, which is particularly well-suited for data modelling. We already made some progress in this direction in (Gutiérrez-Basulto et al. 2024) – we end this thesis by discussing this progress and its limitations in more detail at the end of this chapter – but the next step seemingly requires a significant generalization of the techniques used in this thesis, to handle the two-directionality of the modal constraints. In this context, techniques developed by Pratt-Hartmann (2009) and by Otto (2012) are worth exploring.

Extending the query language with inverses (UC2RPQs) is likely related to handling inverses in the description logic, as it also requires handling two-directionality. However, the fact that the permutation semiautomaton construction, crucial for our approach, strongly depends on the determinism of the (semi)automaton, might also prove troublesome for handling UC2RPQs. To circumvent this, it might be worth studying the results by Bojańczyk and Parys (2010).

One of the goals of the chosen presentation of the results and proofs in this thesis was to make it easier to find connections to other constructions and techniques, possibly in different fields. We have already mentioned possible connections to some constructions in (Danielski and Kieroński 2019) and in (Canavoi and Otto 2017). The first paper studies fragments of classical logics (the unary negation fragment of first-order logic, extended with transitivity), while the second paper studies a different aspect of knowledge representation ("multi-agent epistemic modal logic with common knowledge modalities for groups of agents"), and uses techniques based on group theory. Studying these connections might result in a fruitful transfer of knowledge between the fields.

12.1 Extensions

In the last section of this thesis, we discuss extending the description logic with either counting or inverses. We explain how we achieved this in (Gutiérrez-Basulto et al. 2024), and how the techniques presented in this thesis can possibly be extended.

12.1.1 Counting

In (Gutiérrez-Basulto et al. 2024) we show that the problem of UCRPQ containment modulo an $\mathcal{ALCQ}$ TBox is decidable and 2ExpTime-complete; the description logic $\mathcal{ALCQ}$ extends $\mathcal{ALC}$ with counting restrictions, such as $C \sqsubseteq \exists_{\leq 5} r.D$ meaning "each element in concept C has at most 5 r-successors in concept D".

Importantly, when the description logic is non-local – e.g. $\mathcal{ALCQ}_{\text{reg}}$, with regular expressions – there are several ways to define counting restrictions; this distinction is also visible in real graph databases. The most common way in description logics is to allow counting only over "simple roles", not regular expressions. Counting over regular expressions gives rise to the question: what exactly do we count? A restriction of the form $\exists_{\leq 3} r^*.D$ might mean "at most 3 elements in concept D reachable by r^*", or "at most 3 walks over r^* that end in an element in concept D". In real graph databases, sometimes paths (walks without repeated nodes) are counted, to guarantee their finite number.

The former way of counting over regular expressions – counting the number of elements reachable by a regular expression – is known to easily lead to undecidability; see (Horrocks, Sattler, and Tobies 1999; Kazakov, Sattler, and Zolin 2007). For this reason, we discuss only two scenarios: counting over simple roles, and counting over regular expressions with the latter interpretation, i.e. counting walks.

For a given set of constraints that involve counting, there is a maximal number M that appears in the counting restrictions; this number plays an important role in handling such restrictions. Counting walks can seemingly be described in a natural way in our setting, by defining *isomorphism profiles*, consisting of *n-node isomorphism types* of nodes in a graph for $n \geq M$, defined analogously to n-node homomorphism types discussed in the previous chapter. Below we discuss changes that would need to be made to selected constructions to handle isomorphism profiles, and how we handle it in (Gutiérrez-Basulto et al. 2024).

Ravelling constructions

We call a ravelling function f *left-foldable* if, for all walks $\pi = \pi' \cdot e$, $f(\pi)$ is determined by $f(\pi')$ and e. For a left-foldable ravelling function f, intuitively, Ravel_f preserves isomorphism types of unravellings. It is straightforward to verify that all ravelling functions used in the proofs are left-foldable.

Detached graphs

In the reduction that involves detachments of graphs, the most natural way of handling counting restrictions is to handle constants in the outer graph – that is, to work with profiles with nominals throughout the whole algorithm. This way we can ensure that nodes in the inner graph have an appropriate number of successors in the outer graph. It is not guaranteed that all components of the

algorithm can be adjusted to profiles with nominals, but for all combinatorial constructions there is a natural modification to consider: at the end of the construction, simply identify all nodes with the same constant label.

In (Gutiérrez-Basulto et al. 2024) we take a different approach. Recall that the reduction attempts to construct a counterexample for the homomorphism entailment problem. We assume a particular shape of the counterexample: intuitively, we satisfy all counting restrictions for inner nodes by adding edges to fresh copies of the whole counterexample. This is possible both in the case of $\mathcal{ALCQ}$ and $\mathcal{ALCI}$.

We note that, when combining counting with inverses in the description logic $\mathcal{ALCQI}$, such a shape of a counterexample cannot be assumed. As for working with profiles with nominals, while there is seemingly no immediate reason for the approach to fail in the case of $\mathcal{ALCQI}$, the undecidability result for the finite entailment problem for $\mathcal{ALCOIF}$ and UCRPQs by Rudolph (2016) casts a doubt on obtaining decidability results using this approach; perhaps it is more likely that the described reduction might help to obtain undecidability results for the problem of query containment under constraints.

Satisfiability

In (Gutiérrez-Basulto et al. 2024) we solve the problem for constraints expressed in the description logic $\mathcal{ALCQ}$, which cannot use regular expressions – in particular, counting is allowed only over simple roles, and the logic cannot express even reachability. Under these constraints, it is straightforward to adjust the algorithm solving the $\preccurlyeq$-bounded satisfiability problem we described in Section 7.6.2 on page 106. In the construction, we considered the last node of each type in a node-level topological order, and redirected edges between them appropriately; to take into account counting up to some number M, it is enough to consider the last M nodes of each type, and redirect edges between them.

However, if we allow the constraints to express reachability, even with counting restricted to simple roles, the situation becomes more complicated. In the corresponding construction (Section 7.6.3 on page 109), we considered, for each node type, the last SCC with a node of this type, and within each SCC we identified all nodes of the same type. With counting up to some number M, a natural idea is to consider the last M SCCs with nodes of a given type; however, it is not clear what to do within each SCC. We want to end up with an SCC of bounded size, and enough nodes of each type to satisfy all counting restrictions. The author is not aware of any relevant combinatorial constructions that preserve strong connectivity of the graph.

Note that, when solving the satisfiability problem for ranked graphs (Section 8.4 on page 136), we defined a quite complex combinatorial construction that bounds the size of the resulting graph and preserves strong connectivity by fixing some walks between nodes and taking a subgraph that contains all these walks. However, the fact that we managed to bound the size of the resulting graph crucially depended on the ability to "saturate" the graph with edges and showing that this results in short walks between each pair of nodes (Claim 8.44 on page 144). This saturation does not seem to be possible in the presence of counting restrictions – intuitively, we cannot add edges, we can only redirect them.

12.1.2 Inverses

In (Gutiérrez-Basulto et al. 2024) we solve the problem for $\mathcal{ALCI}$ and UCRPQs. This is done by a high-level reduction that relies on a limited interaction between constraints regarding "forward" roles and constraints regarding inverses. We only give a short overview of the approach, because it is not likely to extend to more expressive description logics.

We show that, if a counterexample exists, there is one that consists of multiple components, each corresponding either to "forward" constraints or "inverse" constraints, but not both. All edges are either inside the components, or between components of different types, with a fixed direction (from the "inverse" component to the "forward" component). Because in RPQs inverses cannot be used inside regular expressions, a walk corresponding to a single RPQ can pass through at most two such components, which means that we can easily distribute queries over the components. Inside the "forward" components, the constraints and the queries use only forward roles, so we eliminated the inverses from the problem. Because regular expressions are closed under reversing, inside "inverse" components we can consider "reversed" RPQs, which means that both the constraints and the queries use only inverses; in this case, intuitively, we can treat all edges in the graph as reversed, and solve the problem without inverses.

To adjust the techniques presented in this thesis to the case of inverses, it seems natural to consider two-way unravellings, where edges can be directed from a child to its parent, or, alternatively, have labels of the form a^-, representing a reversed edge. However, it is not clear if ravelling constructions and the Last Edge Appearance Record can be generalized to this case; if not, it is worth investigating the existing constructions for the two-way modal logic (Otto 2004) and for the two-variable guarded fragment of first-order logic with counting (Pratt-Hartmann 2009). There is also a more general form of the first construction: Otto (2012) constructs "finite groups whose Cayley graphs have large girth even w.r.t. a discounted distance measure that contracts arbitrarily long sequences of edges from the same colour class (subgroup), and only counts transitions between colour classes (cosets)"; this result is used in (Canavoi and Otto 2017), which is particularly promising, because the paper mentions reachability and transitive closures.

Bibliography

Abicht, Konrad (2023). *OWL Reasoners still useable in 2023*. Version v1. cs.AI: 2309.06888 (arXiv). URL: https://arxiv.org/abs/2309.06888 (cit. on p. 12).

Amarilli, Antoine and Michael Benedikt (2015). "Finite Open-World Query Answering with Number Restrictions". In: *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. LICS '15. USA: IEEE Computer Society, pp. 305–316 (cit. on p. 14).

Andréka, Hajnal, István Németi, and Johan van Benthem (1998). "Modal Languages and Bounded Fragments of Predicate Logic". In: *Journal of Philosophical Logic* 27, pp. 217–274 (cit. on p. 9).

Angles, Renzo, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč (2017). "Foundations of modern query languages for graph databases". In: *ACM Computing Surveys (CSUR)* 50.5, pp. 1–40 (cit. on p. 14).

Angles, Renzo, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savković, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoč, Mingxi Wu, and Dusan Živković (2023). "PG-Schemas: Schemas for Property Graphs". English. In: *Proceedings of the ACM on Management of Data* 1.2, pp. 1–25 (cit. on p. 15).

Baader, Franz, Bartosz Bednarczyk, and Sebastian Rudolph (2019). "Satisfiability Checking and Conjunctive Query Answering in Description Logics with Global and Local Cardinality Constraints." In: *Description Logics* (cit. on p. 115).

Baier, Christel and Joost-Pieter Katoen (2008). *Principles of Model Checking*. MIT Press (cit. on p. 28).

Bárány, Vince, Georg Gottlob, and Martin Otto (2010). "Querying the Guarded Fragment". In: *2010 25th Annual IEEE Symposium on Logic in Computer Science*. IEEE, pp. 1–10 (cit. on p. 13).

Bárány, Vince, Balder ten Cate, and Luc Segoufin (2015). "Guarded negation". In: *Journal of the ACM (JACM)* 62.3, pp. 1–26 (cit. on p. 9).

Bednarczyk, Bartosz and Emanuel Kieroński (2022). "Finite Entailment of Local Queries in the Z Family of Description Logics". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5, pp. 5487–5494 (cit. on p. 16).

Berardi, Daniela, Diego Calvanese, and Giuseppe De Giacomo (2005). "Reasoning on UML class diagrams". In: *Artificial intelligence* 168.1-2, pp. 70–118 (cit. on p. 12).

Bienvenu, Meghyn, Carsten Lutz, and Frank Wolter (2012). "Query containment in description logics reconsidered". In: *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*. KR'12. Rome, Italy: AAAI Press, pp. 221–231 (cit. on p. 9).

Blackburn, Patrick, Maarten de Rijke, and Yde Venema (2001). *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (cit. on pp. 29, 74).

Bojańczyk, Mikołaj (2009). "Factorization Forests". In: *Proceedings of the 13th International Conference on Developments in Language Theory*. DLT '09. Stuttgart, Germany: Springer-Verlag, pp. 1–17 (cit. on pp. iii, v, 17, 155, 190).

Bojańczyk, Mikołaj and Paweł Parys (2010). "Efficient Evaluation of Nondeterministic Automata Using Factorization Forests". In: *Automata, Languages and Programming*. Ed. by Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 515–526 (cit. on pp. 190, 193).

Boneva, Iovka, Benoît Groz, Jan Hidders, Filip Murlak, and Sławek Staworko (2023). "Static Analysis of Graph Database Transformations". In: *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS '23. Seattle, WA, USA: Association for Computing Machinery, pp. 251–261 (cit. on pp. 13, 16).

Calvanese, Diego, Thomas Eiter, and Magdalena Ortiz (2009). "Regular Path Queries in Expressive Description Logics with Nominals". In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. IJCAI'09. Pasadena, California, USA: Morgan Kaufmann Publishers Inc., pp. 714–720 (cit. on p. 8).

Calvanese, Diego, Thomas Eiter, and Magdalena Ortiz (2014). "Answering regular path queries in expressive Description Logics via alternating tree-automata". In: *Information and Computation* 237, pp. 12–55 (cit. on p. 8).

Calvanese, Diego, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi (2000). "Containment of conjunctive regular path queries with inverse". In: *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*. KR'00. Breckenridge, Colorado, USA: Morgan Kaufmann Publishers Inc., pp. 176–185 (cit. on p. 15).

Calvanese, Diego, Maurizio Lenzerini, and Daniele Nardi (1998). "Description Logics for Conceptual Data Modeling". In: *Logics for Databases and Information Systems*. Ed. by Jan Chomicki and Gunter Saake. Boston, MA: Springer US, pp. 229–263 (cit. on p. 12).

Canavoi, Felix and Martin Otto (2017). "Common knowledge and multi-scale locality analysis in Cayley structures". In: *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '17. Reykjavík, Iceland: IEEE Press (cit. on pp. 19, 120, 193, 196).

Chandra, Ashok K. and Philip M. Merlin (1977). "Optimal implementation of conjunctive queries in relational data bases". In: *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*. STOC '77. Boulder, Colorado, USA: Association for Computing Machinery, pp. 77–90 (cit. on pp. 12, 15, 35).

Cosmadakis, Stavros S., Paris C. Kanellakis, and Moshe Y. Vardi (1990). "Polynomial-Time Implication Problems for Unary Inclusion Dependencies". In: *J. ACM* 37.1, pp. 15–46 (cit. on pp. 12, 14).

Courcelle, Bruno (1990). "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs". In: *Information and Computation* 85.1, pp. 12–75 (cit. on p. 9).

Danielski, Daniel and Emanuel Kieroński (2019). "Finite Satisfiability of Unary Negation Fragment with Transitivity". In: *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Ed. by Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen. Vol. 138.

Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 17:1–17:15 (cit. on pp. 14, 16, 19, 193).

Deutsch, Alin and Val Tannen (2002). "Optimization Properties for Classes of Conjunctive Regular Path Queries". In: *Database Programming Languages: 8th International Workshop, DBPL 2001, Frascati, Italy, September 8-10, 2001. Revised Papers*. Vol. 2397. Springer Science & Business Media, pp. 21–39 (cit. on p. 15).

Figueira, Diego, Adwait Godbole, S Krishna, Wim Martens, Matthias Niewerth, and Tina Trautner (2020). "Containment of Simple Conjunctive Regular Path Queries". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 17. 1, pp. 371–380 (cit. on p. 15).

Florescu, Daniela, Alon Levy, and Dan Suciu (1998). "Query containment for conjunctive queries with regular expressions". In: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '98. Seattle, Washington, USA: Association for Computing Machinery, pp. 139–148 (cit. on p. 15).

Garson, James (2024). "Modal Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Spring 2024. Metaphysics Research Lab, Stanford University (cit. on p. 5).

Gheerbrant, Amélie, Leonid Libkin, Liat Peterfreund, and Alexandra Rogova (2025). "Database Theory in Action: Cypher, GQL, and Regular Path Queries". In: *28th International Conference on Database Theory (ICDT 2025)*. Ed. by Sudeepa Roy and Ahmet Kara. Vol. 328. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 36:1–36:5 (cit. on p. 14).

Gogacz, Tomasz, Víctor Gutiérrez-Basulto, Albert Gutowski, Yazmín Ibáñez-García, and Filip Murlak (2020). "On Finite Entailment of Non-Local Queries in Description Logics". In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 424–433 (cit. on pp. 16, 113, 189).

Gogacz, Tomasz, Víctor Gutiérrez-Basulto, Yazmín Ibáñez-García, Jean Christoph Jung, and Filip Murlak (2019). "On Finite and Unrestricted Query Entailment beyond SQ with Number Restrictions on Transitive Roles". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 1719–1725 (cit. on p. 16).

Gogacz, Tomasz, Yazmín Ibañez-García, and Filip Murlak (2018). "Finite Query Answering in Expressive Description Logics with Transitive Roles". In: *International Conference on Principles of Knowledge Representation and Reasoning* (cit. on pp. iii, v, 16 sq., 74, 114).

Gogacz, Tomasz and Jerzy Marcinkowski (2013). "On the BDD/FC conjecture". In: *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS '13. New York, New York, USA: Association for Computing Machinery, pp. 127–138 (cit. on pp. 16, 74).

Grädel, Erich (1999). "On the restraining power of guards". In: *The Journal of Symbolic Logic* 64.4, pp. 1719–1742 (cit. on p. 15).

Grädel, Erich, Martin Otto, and Eric Rosen (1999). "Undecidability results on two-variable logics". In: *Archive for Mathematical Logic* 38.4, pp. 313–354 (cit. on p. 15).

Grädel, Erich and Igor Walukiewicz (1999). “Guarded Fixed Point Logic”. In: *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*. LICS ’99. USA: IEEE Computer Society, p. 45 (cit. on p. 9).

Gutiérrez-Basulto, Víctor, Albert Gutowski, Yazmín Ibáñez-García, and Filip Murlak (2022). “Finite Entailment of UCRPQs over ALC Ontologies”. In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 184–194 (cit. on pp. 17, 19, 106, 113, 186, 189, 191).

Gutiérrez-Basulto, Víctor, Albert Gutowski, Yazmín Ibáñez-García, and Filip Murlak (2024). “Containment of Graph Queries Modulo Schema”. In: *Proceedings of the ACM on Management of Data* 2.2 (cit. on pp. 16 sq., 19, 113, 186, 189, 191, 193–196).

Hesse, William M. (2003). “Dynamic computational complexity”. PhD thesis. University of Massachusetts Amherst (cit. on pp. iii, v, 17, 155).

Horrocks, Ian, Ulrike Sattler, and Stephan Tobies (1999). “Practical Reasoning for Expressive Description Logics”. In: *Logic for Programming and Automated Reasoning*. Ed. by Harald Ganzinger, David McAllester, and Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 161–180 (cit. on p. 194).

Ibañez-García, Yazmín, Carsten Lutz, and Thomas Schneider (2014). “Finite model reasoning in horn description logics”. In: *Proc. KR* 14 (cit. on p. 13).

Ioannidis, Yannis E and Raghu Ramakrishnan (1995). “Containment of conjunctive queries: Beyond relations as sets”. In: *ACM Transactions on Database Systems (TODS)* 20.3, pp. 288–324 (cit. on p. 9).

Janin, David and Igor Walukiewicz (1996). “On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic”. In: *CONCUR ’96: Concurrency Theory*. Ed. by Ugo Montanari and Vladimiro Sassone. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 263–277 (cit. on p. 11).

Johnson, D. S. and A. Klug (1982). “Testing containment of conjunctive queries under functional and inclusion dependencies”. In: *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*. PODS ’82. Los Angeles, California: Association for Computing Machinery, pp. 164–169 (cit. on p. 12).

Jung, Jean Christoph, Carsten Lutz, Mauricio Martel, and Thomas Schneider (2018). “Querying the Unary Negation Fragment with Regular Path Expressions”. In: *21st International Conference on Database Theory (ICDT 2018)*. Ed. by Benny Kimelfeld and Yael Amsterdamer. Vol. 98. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 15:1–15:18 (cit. on p. 16).

Kazakov, Yevgeny, Ulrike Sattler, and Evgeny Zolin (2007). “How Many Legs Do I Have? Non-Simple Roles in Number Restrictions Revisited”. In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Nachum Dershowitz and Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 303–317 (cit. on pp. 16, 194).

Kozen, Dexter (1982). “Results on the propositional μ-calculus”. In: *Automata, Languages and Programming*. Ed. by Mogens Nielsen and Erik Meineche Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 348–359 (cit. on p. 9).

Leinberger, Martin, Philipp Seifer, Tjitze Rienstra, Ralf Lämmel, and Steffen Staab (2020). "Deciding SHACL shape containment through description logics reasoning". In: *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I 19*. Springer, pp. 366–383 (cit. on p. 15).

Maier, David, Alberto O. Mendelzon, and Yehoshua Sagiv (1979). "Testing implications of data dependencies". In: *ACM Trans. Database Syst.* 4.4, pp. 455–469 (cit. on p. 12).

Matentzoglu, Nicolas, Jared Leo, Valentino Hudhra, Uli Sattler, and Bijan Parsia (2015). "A Survey of Current, Stand-alone OWL Reasoners". English. In: *Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015) co-located with the 28th International Workshop on Description Logics (DL 2015), Athens, Greece, June 6, 2015*. Pp. 68–79 (cit. on p. 12).

Oracle (2025). *Documentation for Oracle Database*. URL: `https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/advanced-query-rewrite-materialized-views.html` (visited on May 2, 2025) (cit. on p. 11).

Otto, Martin (2004). "Modal and guarded characterisation theorems over finite transition systems". In: *Annals of Pure and Applied Logic* 130.1. Papers presented at the 2002 IEEE Symposium on Logic in Computer Science (LICS), pp. 173–205 (cit. on pp. 13 sq., 196).

Otto, Martin (2012). "Highly acyclic groups, hypergraph covers, and the guarded fragment". In: *Journal of the ACM (JACM)* 59.1, pp. 1–40 (cit. on pp. 13, 193, 196).

Pareti, Paolo, George Konstantinidis, Fabio Mogavero, and Timothy J. Norman (2020). "SHACL Satisfiability and Containment". In: *The Semantic Web – ISWC 2020*. Ed. by Jeff Z. Pan, Valentina Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal. Cham: Springer International Publishing, pp. 474–493 (cit. on p. 16).

Pratt-Hartmann, Ian (2009). "Data-complexity of the two-variable fragment with counting quantifiers". In: *Information and Computation* 207.8, pp. 867–888 (cit. on pp. 13, 193, 196).

Pratt-Hartmann, Ian, Wiesław Szwast, and Lidia Tendera (2019). "The fluted fragment revisited". In: *The Journal of Symbolic Logic* 84.3, pp. 1020–1048 (cit. on p. 9).

Pratt-Hartmann, Ian and Lidia Tendera (2019). "The Fluted Fragment with Transitivity". English. In: 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019 ; Conference date: 26-08-2019 Through 30-08-2019, 18:1–18:15 (cit. on p. 15).

Quine, Willard V. (1969). "On the limits of decision". In: *Akten des XIV. Internationalen Kongresses für Philosophie* 3, pp. 57–62 (cit. on p. 9).

Reutter, Juan L, Miguel Romero, and Moshe Y. Vardi (2017). "Regular queries on graph databases". In: *Theory of computing Systems* 61, pp. 31–83 (cit. on p. 14).

Rosati, Riccardo (2006). "On the Decidability and Finite Controllability of Query Processing in Databases with Incomplete Information". In: *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '06. Chicago, IL, USA: Association for Computing Machinery, pp. 356–365 (cit. on pp. iii, v, 13 sq., 18, 74).

Rosati, Riccardo (2008). "Finite model reasoning in DL-Lite". In: *European Semantic Web Conference*. Springer, pp. 215–229 (cit. on p. 13).

Rosen, Eric (1997). "Modal logic over finite structures". In: *Journal of Logic, Language and Information* 6, pp. 427–439 (cit. on p. 13).

Rudolph, Sebastian (2016). “Undecidability Results for Database-Inspired Reasoning Problems in Very Expressive Description Logics”. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*. Ed. by Chitta Baral, James P. Delgrande, and Frank Wolter. AAAI Press, pp. 247–257 (cit. on pp. iii, v, 15 sqq., 195).

Rudolph, Sebastian and Birte Glimm (2010). “Nominals, inverses, counting, and conjunctive queries or: why infinity is your friend!” In: *J. Artif. Int. Res.* 39.1, pp. 429–481 (cit. on p. 13).

Sanderson, Grant (2025). *There's more to those colliding blocks that compute pi*. URL: `https://youtu.be/6dTyOl1fmDo?t=1607` (visited on May 18, 2025) (cit. on p. 18).

Schild, Klaus (1991). “A correspondence theory for terminological logics: preliminary report”. In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'91. Sydney, New South Wales, Australia: Morgan Kaufmann Publishers Inc., pp. 466–471 (cit. on pp. 11, 36).

Scott, Dana (1962). “A decision method for validity of sentences in two variables”. In: *Journal of Symbolic Logic* 27, p. 377 (cit. on p. 9).

Scott, Dana and Jaco W. de Bakker (1969). “A theory of programs”. In: *Unpublished manuscript, IBM, Vienna* (cit. on p. 9).

Simon, Imre (1990). “Factorization forests of finite height”. In: *Theoretical Computer Science* 72.1, pp. 65–94 (cit. on p. 17).

Ten Cate, Balder and Luc Segoufin (2013). “Unary negation”. In: *Logical Methods in Computer Science* 9.3 (cit. on pp. 9, 13 sq.).

Tendera, Lidia (2005). “Counting in the Two Variable Guarded Logic with Transitivity”. In: *STACS 2005*. Ed. by Volker Diekert and Bruno Durand. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 83–96 (cit. on p. 15).

Van Benthem, Johan (1977). “Modal correspondence theory”. PhD thesis. University of Amsterdam (cit. on p. 11).

Van Harmelen, Frank, Vladimir Lifschitz, and Bruce Porter (2007). *Handbook of Knowledge Representation*. San Diego, CA, USA: Elsevier Science (cit. on p. 36).

Xiao, Guohui, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev (2018). “Ontology-Based Data Access: A Survey”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 5511–5519 (cit. on p. 15).

Index