International Journal of Approximate Reasoning 51 (2010) 346–362

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic[☆]

Barbara Dunin-Kęplicz [a,b], Linh Anh Nguyen [a], Andrzej Szałas [a,c,*]

[a] Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
[b] Institute of Computer Science, Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland
[c] Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden

A R T I C L E I N F O

Article history:
Received 27 May 2009
Received in revised form 10 November 2009
Accepted 11 November 2009
Available online 18 November 2009

Keywords:
Rough sets
Similarity relation
Dynamic logic
Knowledge fusion
Approximate database

A B S T R A C T

In this paper we investigate a technique for fusing approximate knowledge obtained from distributed, heterogeneous information sources. This issue is substantial, e.g., in modeling multiagent systems, where a group of loosely coupled heterogeneous agents cooperate in achieving a common goal. Information exchange, leading ultimately to knowledge fusion, is a natural and vital ingredient of this process. We use a generalization of rough sets and relations [30], which depends on allowing arbitrary similarity relations.

The starting point of this research is [6], where a framework for knowledge fusion in multiagent systems is introduced. Agents' individual perceptual capabilities are represented by similarity relations, further aggregated to express joint capabilities of teams. This aggregation, expressing a shift from individual to social level of agents' activity, has been formalized by means of dynamic logic. The approach of Doherty et al. (2007) [6] uses the full propositional dynamic logic, which does not guarantee tractability of reasoning. Our idea is to adapt the techniques of Nguyen [26–28] to provide an engine for tractable approximate database querying restricted to a Horn fragment of serial dynamic logic. We also show that the obtained formalism is quite powerful in applications.

© 2009 Elsevier Inc. All rights reserved.

1. Similarities and approximate reasoning

In this paper we investigate a technique for fusing approximate knowledge obtained from distributed information sources. We use a generalization of rough sets and relations [30], which depends on allowing arbitrary similarity relations, while in [30] only equivalence relations are considered. In order to approximate relations one uses here a covering of the underlying domain by similarity-based neighborhoods. Such approximate relations have been shown to be useful in many application areas requiring the use of approximate knowledge structures [7].

There are many choices of constraints to be placed on the similarity relation used to define upper and lower approximations. For example, one might not want the relation be transitive since similar objects do not naturally chain in a transitive manner. Many of these issues have been discussed in the context of rough sets (see, e.g., [3,5,7,9,12,18,20–22,29,31–36,38,39]). The basic requirement regarding approximations is that the lower approximation of any set/relation is included in its upper approximation. This is equivalent to the seriality of similarity relations (see [10]). We accept this property as the only requirement.

[☆] Supported by the Polish MNiSW Grant No. N206 399334 and by grants from the Swedish Foundation for Strategic Research (SSF), Strategic Research Center MOVIII and the Swedish Research Council (VR) Linnaeus Center CADICS.

[*] Corresponding author. Address: Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
E-mail addresses: keplicz@mimuw.edu.pl (B. Dunin-Kęplicz), nguyen@mimuw.edu.pl (L.A. Nguyen), andsz@mimuw.edu.pl (A. Szałas).

0888-613X/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2009.11.002

The focus of this paper is approximate knowledge fusion based on the idea of approximations. Our starting point is [6], where a framework for knowledge fusion in multiagent systems is introduced. Agents' individual perceptual capabilities are represented by similarity relations, further aggregated to express joint capabilities of teams. The aggregation expressing a shift from individual to social level of agents' activity has been formalized by means of propositional dynamic logic PDL. The approach of [6], as using the full propositional dynamic logic, does not guarantee tractability of reasoning [16]. As advocated before, we work with PDL with seriality requirement, denoted by SPDL. To achieve tractable approximate database querying, we select a Horn fragment of SPDL, denoted by HSPDL and adapt the techniques of [26–28] to provide an engine for computing queries expressed in HSPDL.

The computational engine distinguishes between extensional and intensional databases. To make this distinction clear we use the traditional terminology of description logics [1]:

- ABox (*assertion box*) stands for the extensional database (containing facts).
- TBox (*terminological box*) stands for the intensional database (containing rules).

The method of computing queries is based on an algorithm, which for a TBox $\mathcal{P}$ consisting of an HSPDL logic program and an ABox $\mathcal{A}$, constructs a least SPDL model $\mathcal{M}$ of $\mathcal{P}$ and $\mathcal{A}$. This model has the property that for every positive formula φ and for every individual a, $\varphi(a)$ is a logical consequence of $\mathcal{P}, \mathcal{A}$ in SPDL (denoted by $\mathcal{P}, \mathcal{A} \models_s \varphi(a)$) iff $\varphi(a)$ is true in $\mathcal{M}$ (i.e. $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$). The role of the constructed least model is that it is used to compute answers to queries. The construction of $\mathcal{M}$ is done in time polynomial in the size of $\mathcal{A}$ (and has a polynomial size in the size of $\mathcal{A}$). As a consequence, the problem of checking whether $\mathcal{P}, \mathcal{A} \models_s \varphi(a)$ has PTIME data complexity (measured in the size of $\mathcal{A}$).

The paper is structured as follows. In Section 2 we recall Propositional Dynamic Logic, show its relationship to approximate reasoning and approximate databases, and justify the requirement of seriality. Section 3 is devoted to showing the PTIME data complexity of the selected Horn fragment HSPDL. Section 4 illustrates its potential in an exemplary real-world application. Section 5 shows how to use the introduced formalism for epistemic reasoning in multiagent systems. Finally, Section 6 concludes the paper.

2. Serial propositional dynamic logic

2.1. Language and semantics of SPDL

Let us define *serial propositional dynamic logic* (SPDL). The key idea is to provide calculus on similarity relations rather than on programs. This somehow unusual move allows us to reason about similarities using the whole apparatus of dynamic logic, where "programs" are replaced by similarity relations.

Let $\mathcal{MOD}$ denote the set of *similarity relation symbols*, and $\mathcal{PROP}$ the set of *propositions*. We use letters like σ to indicate elements of $\mathcal{MOD}$, and letters like p, q to indicate elements of $\mathcal{PROP}$.

Definition 2.1. *Formulas* and *similarity expressions* are respectively defined by the two following BNF grammar rules:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi$$
$$\alpha ::= \sigma \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

Operator ; is called the *composition*, $\cup$ the *union*, * the *iteration* and $\varphi?$ the *test operator*.

We use letters like α, β to denote similarity expressions; φ, ψ to denote formulas; and a, b, c to denote *individuals*. Intuitively,

- $\alpha_1 ; \alpha_2$ stands for a set-theoretical composition of relations α_1 and α_2.
- $\alpha_1 \cup \alpha_2$ stands for set-theoretical union of relations α_1 and α_2.
- α^* stands for the reflexive and transitive closure of α.
- $\varphi?$ stands for the test operator.

Operators $\langle\alpha\rangle$ and $[\alpha]$ are modal operators of the dynamic logic with the following intended meaning:

- $\langle\alpha\rangle\varphi$: "there is an object similar w.r.t. α to a given object and satisfying formula φ".
- $[\alpha]\varphi$: "all objects similar w.r.t. α to a given object satisfy φ".

The following definitions naturally capture these intuitions. Observe, however, that rather than possible worlds or states, objects are used as elements of domains of Kripke structures.

Definition 2.2. A Kripke structure is a pair $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$, where $\Delta^{\mathcal{M}}$ is a set of *objects*, and $\cdot^{\mathcal{M}}$ is an interpretation function that maps each individual a to an element $a^{\mathcal{M}}$ of $\Delta^{\mathcal{M}}$, each proposition p to a subset $p^{\mathcal{M}}$ of $\Delta^{\mathcal{M}}$, and each similarity relation symbol σ to a binary relation $\sigma^{\mathcal{M}}$ on $\Delta^{\mathcal{M}}$.

The interpretation function is extended for all formulas and similarity expressions as follows:

$$\begin{aligned}
&\top^{\mathcal{M}} = \Delta^{\mathcal{M}}\\
&(\neg\varphi)^{\mathcal{M}} = \Delta^{\mathcal{M}} \setminus \varphi^{\mathcal{M}}\\
&(\varphi \wedge \psi)^{\mathcal{M}} = \varphi^{\mathcal{M}} \cap \psi^{\mathcal{M}}\\
&(\varphi \vee \psi)^{\mathcal{M}} = \varphi^{\mathcal{M}} \cup \psi^{\mathcal{M}}\\
&(\varphi \to \psi)^{\mathcal{M}} = (\neg\varphi \vee \psi)^{\mathcal{M}}\\
&(\langle\alpha\rangle\varphi)^{\mathcal{M}} = \{x \in \Delta^{\mathcal{M}} \mid \exists y\ [\alpha^{\mathcal{M}}(x,y) \wedge \varphi^{\mathcal{M}}(y)]\}\\
&([\alpha]\varphi)^{\mathcal{M}} = \{x \in \Delta^{\mathcal{M}} \mid \forall y\ [\alpha^{\mathcal{M}}(x,y) \to \varphi^{\mathcal{M}}(y)]\}\\
&(\alpha\ ;\ \beta)^{\mathcal{M}} = \alpha^{\mathcal{M}} \circ \beta^{\mathcal{M}} = \{(x,y) \mid \exists z\ [\alpha^{\mathcal{M}}(x,z) \wedge \beta^{\mathcal{M}}(z,y)]\}\\
&(\alpha \cup \beta)^{\mathcal{M}} = \alpha^{\mathcal{M}} \cup \beta^{\mathcal{M}}\\
&(\alpha^*)^{\mathcal{M}} = (\alpha^{\mathcal{M}})^*\\
&(\varphi?)^{\mathcal{M}} = \{(x,x) \mid \varphi^{\mathcal{M}}(x)\}
\end{aligned}$$

We sometimes write $\mathcal{M},\ x \models \varphi$ to denote $x \in \varphi^{\mathcal{M}}$. For a set Γ of formulas, we write $\mathcal{M},\ x \models \Gamma$ to denote that $\mathcal{M},\ x \models \varphi$ for all $\varphi \in \Gamma$. If $\mathcal{M},\ x \models \Gamma$ for all $x \in \Delta^{\mathcal{M}}$ then we call $\mathcal{M}$ a *model of* Γ. If $\varphi^{\mathcal{M}} = \Delta^{\mathcal{M}}$ then we say that φ is valid in $\mathcal{M}$.

When dealing with the data complexity of the instance checking problem, without loss of generality we can assume that both the sets $\mathcal{MOD}$ and $\mathcal{PROP}$ are finite and fixed. Under this assumption, the *size of a Kripke structure* $\mathcal{M}$ is defined to be the number of elements of the set $\Delta^{\mathcal{M}}$.

Lemma 2.3. *Given a Kripke structure $\mathcal{M}$ with size n and a formula φ with length m, the set $\varphi^{\mathcal{M}}$ can be computed in $O(m \times n^3)$ steps.*

Proof. Just notice that the complexity of computing the transitive closure of a binary relation is $O(n^3)$ (see, e.g., [4]). □

For every $\sigma \in \mathcal{MOD}$, we adopt the axioms

$$[\sigma]\varphi \to \langle\sigma\rangle\varphi \tag{1}$$

(or $\langle\sigma\rangle\top$, equivalently). It is well known (see, e.g., [10,37]) that (1) corresponds to the *seriality property*:

$$\forall x \exists y\ \sigma^{\mathcal{M}}(x,y). \tag{2}$$

Therefore we have the following definition.

Definition 2.4. By an *admissible interpretation for* SPDL we understand any Kripke structure $\mathcal{M}$ with all similarities $\sigma \in \mathcal{MOD}$ satisfying (2). We call such Kripke structures *serial*.

Note that we do not require a serial Kripke structure to satisfy the seriality condition $\forall x \exists y\, \alpha^{\mathcal{M}}(x,y)$ for every similarity expression α. This condition holds when α does not contain the test operator, but does not hold, e.g., for $\alpha = ((\neg\top)?)$.

2.2. SPDL as a query language in approximate databases

Let us now explain how SPDL is used as a query language. First observe that interpretations assign sets of objects to formulas. Thus it is natural to consider each formula as the query selecting all objects satisfying the formula.

Example 2.5. Let, in a given interpretation $\mathcal{M}$:

- $\Delta^{\mathcal{M}} = \{o_1, o_2, o_3, o_4, o_5\}$.
- $red^{\mathcal{M}} = \{o_1, o_3, o_4\}$.
- $small^{\mathcal{M}} = \{o_1, o_2, o_4, o_5\}$.

Then $(red \wedge small)^{\mathcal{M}} = \{o_1, o_4\}$, thus the query $(red \wedge small)$ returns the set $\{o_1, o_4\}$. Similarly, the query $(red \to small)$ returns $\{o_1, o_2, o_4, o_5\}$.

In order to explain the role of similarities and modal operators, let us first recall the notion of approximations.

Definition 2.6. Let Δ be a set of objects and α be a similarity expression representing a serial binary relation on Δ. For $a \in \Delta$, by the *neighborhood of a w.r.t.* α, we understand the set of elements similar to a: $n^{\alpha} \overset{\text{def}}{=} \{b \in \Delta \mid \alpha(a,b)\}$.

For $A \subseteq \Delta$, the *lower and upper approximations of A w.r.t.* α, denoted respectively by A^{+}_{α} and $A^{\oplus}_{\alpha}$, are defined by

$$\begin{aligned}
&A^{+}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \subseteq A\}\\
&A^{\oplus}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \cap A \neq \emptyset\}
\end{aligned}$$

The meaning of those approximations is illustrated in Fig. 1. Intuitively, assuming that the perception of an agent is modeled by similarity expression α,

- $a \in A^{+}_{\alpha}$ means that all objects indiscernible from a are in A.
- $a \in A^{\oplus}_{\alpha}$ means that there are objects indiscernible from a which are in A.

Note that seriality guarantees that the lower approximation of a set is included in the upper approximation of the set. This is the weakest requirement one places on approximations. It is often desirable to have the property that

$$A^{+}_{\alpha} \subseteq A \subseteq A^{\oplus}_{\alpha}, \tag{3}$$

as, in fact, shown in Fig. 1. This property corresponds to the reflexivity of the similarity relation expressed by α (see, e.g., [10,39,37]) and guarantees that

- $a \in A^{+}_{\alpha}$ means that, from the point of view of the agent, a surely is in A, since all objects indiscernible from a are in A.
- $a \in A^{\oplus}_{\alpha}$ means that, from the point of view of the agent, a possibly is in A, since there are objects indiscernible from a which are in A.

Unfortunately, in some applications the set A is only given via its approximations, so constraints (3) cannot be checked automatically. This, in particular, happens when one deals with vague concepts that do not have precise definitions or whose precise definitions are unacceptable in applications. Also, machine learned concepts are often approximated, as, e.g., in version spaces (see [11]).

We have the following proposition which is an immediate consequence of Definition 2.6.

Proposition 2.7. *Let $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$ be a Kripke structure and α be a similarity expression. Then, for any SPDL formula φ and $x \in \Delta^{\mathcal{M}}$:*

$$x \in (\varphi^{\mathcal{M}})^{+}_{\alpha} \text{ iff for all } y \in \Delta^{\mathcal{M}}, \text{ if } \alpha^{\mathcal{M}}(x,y) \text{ holds then } y \in \varphi^{\mathcal{M}}$$
$$x \in (\varphi^{\mathcal{M}})^{\oplus}_{\alpha} \text{ iff there is } y \in \Delta^{\mathcal{M}} \text{ such that } \alpha^{\mathcal{M}}(x,y) \text{ holds and } y \in \varphi^{\mathcal{M}}$$

By Proposition 2.7 we have that:

$$[\alpha]A \text{ expresses the lower approximation of } A \text{ w.r.t. } \alpha, \text{ i.e., } A^{+}_{\alpha} \tag{4}$$
$$\langle\alpha\rangle A \text{ expresses the upper approximation of } A \text{ w.r.t. } \alpha, \text{ i.e., } A^{\oplus}_{\alpha} \tag{5}$$

Remark 2.8. In the view of (4) and (5), axiom (1) expresses the property that the lower approximation of a set A w.r.t. any similarity expression α is included in the upper approximation of A w.r.t. α. As noted before, axiom (1) is equivalent to seriality expressed by (2). This justifies our seriality assumption as reflecting the basic requirement on approximations.

Example 2.9. Let $\mathcal{M}$ be the interpretation considered in Example 2.5. Let σ be the reflexive closure of relation $\{\langle o_1, o_2\rangle, \langle o_2, o_1\rangle, \langle o_3, o_4\rangle\}$. Then, for example, $red^{+}_{\sigma} = \{o_3, o_4\}$, $red^{\oplus}_{\sigma} = \{o_1, o_2, o_3, o_4\}$.

2.3. The Horn fragment HSPDL

In order to express tractable queries we restrict the query language to the Horn fragment HSPDL, defined below.

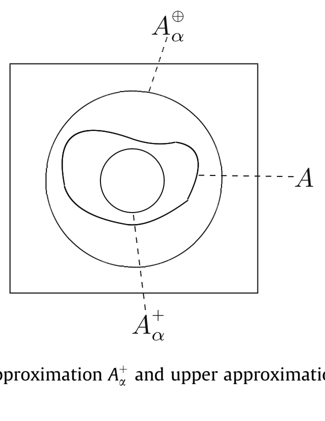


Fig. 1. Lower approximation A^{+}_{α} and upper approximation $A^{\oplus}_{\alpha}$ of a set A.

Definition 2.10. *Positive formulas*, φ_{pos}, are defined by the following BNF grammar:

$$\begin{aligned}
\varphi_{pos} &::= \top \mid p \mid \varphi_{pos} \wedge \varphi_{pos} \mid \varphi_{pos} \vee \varphi_{pos} \mid \langle \alpha_{pos_\Diamond} \rangle \varphi_{pos} \mid [\alpha_{pos_\Box}]\varphi_{pos} \\
\alpha_{pos_\Diamond} &::= \sigma \mid \alpha_{pos_\Diamond} \, ; \, \alpha_{pos_\Diamond} \mid \alpha_{pos_\Diamond} \cup \alpha_{pos_\Diamond} \mid \alpha^*_{pos_\Diamond} \mid \varphi_{pos}? \\
\alpha_{pos_\Box} &::= \sigma \mid \alpha_{pos_\Box} \, ; \, \alpha_{pos_\Box} \mid \alpha_{pos_\Box} \cup \alpha_{pos_\Box} \mid \alpha^*_{pos_\Box} \mid (\neg \varphi_{pos})?
\end{aligned}$$

*HSP*DL *program clauses*, φ_{prog}, are defined by the following BNF grammar:[1]

$$\begin{aligned}
\varphi_{prog} &::= \top \mid p \mid \varphi_{pos} \to \varphi_{prog} \mid \varphi_{prog} \wedge \varphi_{prog} \mid \langle \alpha_{prog_\Diamond} \rangle \varphi_{prog} \mid [\alpha_{prog_\Box}]\varphi_{prog} \\
\alpha_{prog_\Diamond} &::= \sigma \mid \alpha_{prog_\Diamond} \, ; \, \alpha_{prog_\Diamond} \mid \varphi_{prog}? \\
\alpha_{prog_\Box} &::= \sigma \mid \alpha_{prog_\Box} \, ; \, \alpha_{prog_\Box} \mid \alpha_{prog_\Box} \cup \alpha_{prog_\Box} \mid \alpha^*_{prog_\Box} \mid \varphi_{pos}?
\end{aligned}$$

An *HSP*DL *logic program* is a finite set of HSP DL program clauses. The *Horn fragment* HSPDL for the problem of checking whether $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$ consists of HSPDL logic programs for $\mathscr{P}$ and positive formulas for φ.

Example 2.11. Observe that HSPDL is quite expressive. For example, it allows one to express a variant of default rules (discussed, e.g., in [7]). Namely, a typical default rule can be expressed as $A^+_\sigma, \; B^\oplus_\sigma \vdash C^+_\sigma$, with intuitive meaning "if A is surely true and B might be true then accept C as surely true".

Let us now formally link SPDL with databases.

Definition 2.12. An *individual assertion* is an expression of the form $p(a)$, where p is a proposition and a is an individual. A *similarity assertion* is an expression of the form $\sigma(a, b)$, where σ is a similarity relation symbol and a, b are individuals. An *ABox* is a finite set of individual assertions and similarity assertions.

Comparing to description logics, individual assertions correspond to concept assertions, and similarity assertions correspond to role assertions. An ABox provides an extensional database (in [17], such an ABox is said to be *extensionally reduced*).

Definition 2.13. Given a Kripke structure $\mathscr{M}$ and an ABox $\mathscr{A}$, we say that $\mathscr{M}$ is a *model of* $\mathscr{A}$, denoted by $\mathscr{M} \models \mathscr{A}$, if $a^{\mathscr{M}} \in p^{\mathscr{M}}$ for every individual assertion $p(a) \in \mathscr{A}$ and $(a^{\mathscr{M}}, b^{\mathscr{M}}) \in \sigma^{\mathscr{M}}$ for every similarity assertion $\sigma(a, b) \in \mathscr{A}$.

Definition 2.14. Given an HSPDL logic program $\mathscr{P}$, an ABox $\mathscr{A}$, a positive formula φ and an individual a, we say that *a has the property φ w.r.t. $\mathscr{P}$ and $\mathscr{A}$ in SP*DL (or $\varphi(a)$ is a logical consequence of $\mathscr{P}, \mathscr{A}$ in SPDL), denoted by $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$, if for every serial Kripke structure $\mathscr{M}$, if $\mathscr{M}$ is a model of $\mathscr{P}$ and $\mathscr{A}$ then $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$.

Recall that the pair $\mathscr{P}, \mathscr{A}$ is treated as a database.

Definition 2.15. By the *instance checking* problem for HSPDL we mean the problem of checking whether $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$. The *data complexity* of this problem is measured when $\mathscr{P}$, φ and a are fixed (and compose a query), while $\mathscr{A}$ varies as input data.

3. Computational aspects

3.1. Ordering Kripke structures

Definition 3.1. A Kripke structure $\mathscr{M} = \langle \Delta^{\mathscr{M}}, \cdot^{\mathscr{M}} \rangle$ is said to be *less than or equal to* $\mathscr{M}' = \langle \Delta^{\mathscr{M}'}, \cdot^{\mathscr{M}'} \rangle$, denoted by $\mathscr{M} \leqslant \mathscr{M}'$, if for every positive formula φ and every individual a, $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$ implies $a^{\mathscr{M}'} \in \varphi^{\mathscr{M}'}$.

Definition 3.2. Given Kripke structures $\mathscr{M} = \langle \Delta^{\mathscr{M}}, \cdot^{\mathscr{M}} \rangle$ and $\mathscr{M}' = \langle \Delta^{\mathscr{M}'}, \cdot^{\mathscr{M}'} \rangle$ and a binary relation $r \subseteq \Delta^{\mathscr{M}} \times \Delta^{\mathscr{M}'}$, we say that $\mathscr{M}$ *is less than or equal to $\mathscr{M}'$ w.r.t. r*, denoted by $\mathscr{M} \leqslant_r \mathscr{M}'$, if the following conditions hold for every individual a, every similarity relation symbol σ, and every proposition p:

1. $r(a^{\mathscr{M}}, a^{\mathscr{M}'})$.
2. $\forall x, x', y \left[[\sigma^{\mathscr{M}}(x, y) \wedge r(x, x')] \to \exists y' [\sigma^{\mathscr{M}'}(x', y') \wedge r(y, y')] \right]$.
3. $\forall x, x', y' \left[[\sigma^{\mathscr{M}'}(x', y') \wedge r(x, x')] \to \exists y [\sigma^{\mathscr{M}}(x, y) \wedge r(y, y')] \right]$.
4. $\forall x, x' [r(x, x') \to (x \in p^{\mathscr{M}} \to x' \in p^{\mathscr{M}'})]$.

In Definition 3.2, the first three conditions state that r is a kind of bisimulation between the *frames* of $\mathscr{M}$ and $\mathscr{M}'$. Intuitively, $r(x, x')$ states that x has fewer positive properties than x'.

[1] Notice the two occurrences of φ_{pos} in the grammar. We do not allow formulas of the form $\langle \alpha \cup \beta \rangle \varphi$ or $\langle \alpha^* \rangle \varphi$ to be HSPDL program clauses because they cause non-determinism.

Consider a similarity expression α. In Lemma 3.4 formulated below we will use an inductive argument based on selecting from α a "path" γ of atomic expressions (similarity relation symbols and tests) occurring in α, with the property that in a given model $\mathcal{M}$, $\alpha^{\mathcal{M}}(x,y)$ iff $\gamma^{\mathcal{M}}(x,y)$. Intuitively, such a path reflects a run of regular program expressed by α, consisting of atomic programs and tests. Formally, the argument requires the following definitions.

Definition 3.3. The *alphabet* $\Sigma(\alpha)$ *of a similarity expression* α is defined as follows:

$$\begin{aligned}&\Sigma(\sigma) = \{\sigma\}\\&\Sigma(\varphi?) = \{\varphi?\}\\&\Sigma(\beta_1;\beta_2) = \Sigma(\beta_1) \cup \Sigma(\beta_2)\\&\Sigma(\beta_1 \cup \beta_2) = \Sigma(\beta_1) \cup \Sigma(\beta_2)\\&\Sigma(\beta^*) = \Sigma(\beta)\end{aligned}$$

Note that, according to Definition 3.3, $\Sigma(\alpha)$ contains not only similarity relation symbols but also expressions of the form $\varphi?$.

A similarity expression α is a regular expression over its alphabet $\Sigma(\alpha)$. The regular language $\mathcal{L}(\alpha)$ generated by α is defined as follows:

$$\begin{aligned}&\mathcal{L}(\sigma) = \{\sigma\}\\&\mathcal{L}(\varphi?) = \{\varphi?\}\\&\mathcal{L}(\beta \cup \beta') = \mathcal{L}(\beta) \cup \mathcal{L}(\beta')\\&\mathcal{L}(\beta\ ;\ \beta') = \mathcal{L}(\beta) \cdot \mathcal{L}(\beta')\\&\mathcal{L}(\beta^*) = (\mathcal{L}(\beta))^*\end{aligned}$$

where if L and M are sets of words then $L \cdot M = \{\alpha\beta \mid \alpha \in L, \beta \in M\}$ and $L^* = \bigcup_{n \geqslant 0} L^n$ with $L^0 = \{\varepsilon\}$ and $L^{n+1} = L \cdot L^n$ (ε stands for the empty word).

We treat words of $\mathcal{L}(\alpha)$ also as similarity expressions, e.g. $\sigma_1\sigma_2$ denotes $(\sigma_1\ ;\ \sigma_2)$.

Lemma 3.4. Let $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$ and $\mathcal{M}' = \langle \Delta^{\mathcal{M}'}, \cdot^{\mathcal{M}'} \rangle$ be Kripke structures, and $r \subseteq \Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}'}$ be a relation that satisfies Conditions 2–4 of Definition 3.2. If $r(x,x')$ holds then, for every positive formula φ, $x \in \varphi^{\mathcal{M}}$ implies $x' \in \varphi^{\mathcal{M}'}$.

Proof. We prove this lemma by an induction on the structure of φ. Assume that $r(x,x')$ holds and $x \in \varphi^{\mathcal{M}}$.

- The cases when $\varphi = p$ or $\varphi = \psi \wedge \xi$ or $\varphi = \psi \vee \xi$ are trivial.
- Case $\varphi = \langle\alpha\rangle\psi$:
  Since $x \in \varphi^{\mathcal{M}}$, there exists $y \in \Delta^{\mathcal{M}}$ such that $\alpha^{\mathcal{M}}(x,y)$ holds and $y \in \psi^{\mathcal{M}}$. There exists

$$\gamma = \sigma_1 \cdots \sigma_{i_1}(\xi_1?)\sigma_{i_1+1} \cdots \sigma_{i_2}(\xi_2?) \cdots \sigma_{i_k} \in \mathcal{L}(\alpha)$$

such that $\gamma^{\mathcal{M}}(x,y)$ holds. Hence, there are elements

$$x_0 = x, x_1, \ldots, x_{i_k-1}, x_{i_k} = y$$

of $\Delta^{\mathcal{M}}$ such that $\sigma_j^{\mathcal{M}}(x_{j-1},x_j)$ holds for $1 \leqslant j \leqslant i_k$ and $x_{i_h} \in \xi_h^{\mathcal{M}}$ for $1 \leqslant h \leqslant k-1$. Let $x'_0 = x'$. By Condition 2 of Definition 3.2, for every $1 \leqslant j \leqslant i_k$ there exists $x'_j \in \Delta^{\mathcal{M}'}$ such that $\sigma_j^{\mathcal{M}'}(x'_{j-1},x'_j)$ and $r(x_j,x'_j)$ hold. Since $r(x_{i_h},x'_{i_h})$ holds, $x_{i_h} \in \xi_h^{\mathcal{M}}$ and ξ_h is a positive formula, by the inductive assumption, for $1 \leqslant h \leqslant k-1$ we have that $x'_{i_h} \in \xi_h^{\mathcal{M}'}$. Hence, $\gamma^{\mathcal{M}'}(x',y')$ and $r(y,y')$ hold for $y' = x'_{i_k}$. Thus, we also have that $\alpha^{\mathcal{M}'}(x',y')$ holds. Since $r(y,y')$ holds and $y \in \psi^{\mathcal{M}}$, by the inductive assumption, $y' \in \psi^{\mathcal{M}'}$. Hence $x' \in (\langle\alpha\rangle\psi)^{\mathcal{M}'}$.
- Case $\varphi = [\alpha]\psi$:
  Let y' be an arbitrary element of $\Delta^{\mathcal{M}'}$ such that $\alpha^{\mathcal{M}'}(x',y')$ holds. (If such a y' does not exist then $x' \in \varphi^{\mathcal{M}'}$ clearly holds.) There exists

$$\gamma = \sigma_1 \cdots \sigma_{i_1}(\neg\xi_1?)\sigma_{i_1+1} \cdots \sigma_{i_2}(\neg\xi_2?) \cdots \sigma_{i_k} \in \mathcal{L}(\alpha)$$

such that $\gamma^{\mathcal{M}'}(x',y')$ holds. Hence, there are elements

$$x'_0 = x', x'_1, \ldots, x'_{i_k-1}, x'_{i_k} = y'$$

of $\Delta^{\mathcal{M}'}$ such that $\sigma_j^{\mathcal{M}'}(x'_{j-1},x'_j)$ holds for $1 \leqslant j \leqslant i_k$ and $x'_{i_h} \in (\neg\xi_h)^{\mathcal{M}'}$ holds for $1 \leqslant h \leqslant k-1$. Let $x_0 = x$. By Condition 3 of Definition 3.2, for every $1 \leqslant j \leqslant i_k$ there exists $x_j \in \Delta^{\mathcal{M}}$ such that $\sigma_j^{\mathcal{M}}(x_{j-1},x_j)$ and $r(x_j,x'_j)$ hold. For $1 \leqslant h \leqslant k-1$, since $r(x_{i_h},x'_{i_h})$ holds and $x'_{i_h} \in (\neg\xi_h)^{\mathcal{M}'}$ and ξ_h is a positive formula, by the inductive assumption (via contrapositive), we have that $x_{i_h} \in (\neg\xi_h)^{\mathcal{M}}$. Hence, $\gamma^{\mathcal{M}}(x,y)$ and $r(y,y')$ hold for $y = x_{i_k}$. Thus, we also have that $\alpha^{\mathcal{M}}(x,y)$ holds. It follows that $y \in \psi^{\mathcal{M}}$. Since $r(y,y')$ holds and $y \in \psi^{\mathcal{M}}$, by the inductive assumption, $y' \in \psi^{\mathcal{M}'}$. Hence $x' \in ([\alpha]\psi)^{\mathcal{M}'}$. □

Corollary 3.5. *Let $\mathcal{M}$ and $\mathcal{M}'$ be Kripke structures such that $\mathcal{M} \leqslant_r \mathcal{M}'$ for some r. Then $\mathcal{M} \leqslant \mathcal{M}'$.*

Definition 3.6. Let $\mathscr{P}$ be an HSPDL logic program and $\mathscr{A}$ be an ABox. We say that a Kripke structure $\mathscr{M}$ is a *least SPDL model of $\mathscr{P}$ and $\mathscr{A}$* if $\mathscr{M}$ is an SPDL model of $\mathscr{P}$ and $\mathscr{A}$ and for any other SPDL model $\mathscr{M}'$ of $\mathscr{P}$ and $\mathscr{A}$ we have that $\mathscr{M} \leqslant \mathscr{M}'$.

3.2. The algorithm

In this section, we present an algorithm that, given an HSPDL logic program $\mathscr{P}$ and an ABox $\mathscr{A}$, constructs a finite least SPDL model of $\mathscr{P}$ and $\mathscr{A}$. During execution, the algorithm constructs the following data structures:

- Δ is a set of objects. We distinguish the subset Δ_0 of Δ that consists of all the individuals occurring in the ABox $\mathscr{A}$. In the case $\mathscr{A}$ is empty, let $\Delta_0 = \{\tau\}$ for some element τ.
- H is a mapping that maps every $x \in \Delta$ to a set of formulas, which are the properties that should hold for x. When the elements of Δ are treated as states, $H(x)$ denotes the contents of the state x.
- *Next* is a mapping such that, for $x \in \Delta$ and $\langle\sigma\rangle\varphi \in H(x)$, we have $Next(x, \langle\sigma\rangle\varphi) \in \Delta$. The meaning of $Next(x, \langle\sigma\rangle\varphi) = y$ is that:
  - $\langle\sigma\rangle\varphi \in H(x)$ and $\varphi \in H(y)$,
  - the "requirement" $\langle\sigma\rangle\varphi$ is realized for x by going to y via a σ-transition.

We call the tuple $\langle \Delta, H, Next\rangle$ a *model graph.*
Using the above data structures, we define a Kripke structure $\mathscr{M}$ such that:

- $\Delta^{\mathscr{M}} = \Delta$,
- $a^{\mathscr{M}} = a$ for every individual a occurring in $\mathscr{A}$,
- $p^{\mathscr{M}} = \{x \in \Delta \mid p \in H(x)\}$ for every $p \in \mathscr{PROP}$,
- $\sigma^{\mathscr{M}} = \{(a,b) \mid \sigma(a,b) \in \mathscr{A}\} \cup \{(x,y) \mid \ Next(x, \langle\sigma\rangle\varphi) = y$ for some $\varphi\}$ for every $\sigma \in \mathscr{MOD}$.

Definition 3.7. For $x, y \in \Delta$, we say that y is *reachable from x* if there exists a word $\sigma_1 \cdots \sigma_k$ such that $(\sigma_1 \cdots \sigma_k)^{\mathscr{M}}(x,y)$ holds. We say that y is *reachable from Δ_0* if it is reachable from some $x \in \Delta_0$.

Definition 3.8. The *saturation* of a set Γ of formulas, denoted by $\mathsf{Sat}(\Gamma)$, is defined to be the smallest superset of Γ such that:

- $\top \in \mathsf{Sat}(\Gamma)$ and $\langle\sigma\rangle\top \in \mathsf{Sat}(\Gamma)$ for all $\sigma \in \mathscr{MOD}$,
- if $\varphi \wedge \psi \in \mathsf{Sat}(\Gamma)$ or $\langle\varphi?\rangle\psi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $\psi \in \mathsf{Sat}(\Gamma)$,
- if $\langle\alpha\,;\,\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$ then $\langle\alpha\rangle\langle\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha\,;\,\beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha][\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha \cup \beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha]\varphi \in \mathsf{Sat}(\Gamma)$ and $[\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $[\alpha][\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\varphi?]\psi \in \mathsf{Sat}(\Gamma)$ then $(\varphi \to \psi) \in \mathsf{Sat}(\Gamma)$.

Observe that $\mathsf{Sat}(\Gamma)$ is finite when Γ is finite. Define the size of a set of formulas to be the sum of the lengths of its formulas. It can be shown that the size of $\mathsf{Sat}(\Gamma)$ is quadratic in the size of Γ (cf. Lemma 6.3 in [16]).

Definition 3.9. The *transfer of Γ through σ* is defined by:

$$\mathsf{Trans}(\Gamma, \sigma) \overset{\text{def}}{=} \mathsf{Sat}(\{\varphi \mid [\sigma]\varphi \in \Gamma\}).$$

We use procedure $\mathsf{Find}(\Gamma)$ defined as:

if there exists $x \in \Delta \setminus \Delta_0$ with $H(x) = \Gamma$ then return x,
else add a new object x to Δ with $H(x) = \Gamma$ and return x.

The algorithm shown in Fig. 2 constructs a least SPDL model for an HSPDL logic program $\mathscr{P}$ and an ABox $\mathscr{A}$ as follows. At the beginning, Δ starts from Δ_0, which consists of all the individuals occurring in $\mathscr{A}$ or some τ if $\mathscr{A}$ is empty, with $H(x)$, for $x \in \Delta_0$, being the saturation of $\mathscr{P} \cup \{p \mid p(x) \in \mathscr{A}\}$. Then for each $x \in \Delta$ reachable from Δ_0 and for each formula $\varphi \in H(x)$ that does not hold for x, the algorithm makes a change to satisfy φ for x.

There are three forms to be considered for φ[2]:

1. φ is of the form $\langle\sigma\rangle\psi$:
   to satisfy φ for x, we connect x via a σ-transition to an object $y \in \Delta \setminus \Delta_0$ with

$$H(y) = \mathsf{Sat}(\{\psi\} \cup \{\xi \mid [\sigma]\xi \in H(x)\} \cup \mathscr{P})$$

by setting $Next(x, \langle\sigma\rangle\psi) := y$

[2] The other possible forms of φ are dealt with by the saturation operator Sat.

```
Input: An HSPDL logic program $\mathcal{P}$ and an ABox $\mathcal{A}$.
Output: A least SPDL model $\mathcal{M}$ of $\mathcal{P}$ and $\mathcal{A}$.

  1. let $\Delta_0$ be the set of all individuals occurring in $\mathcal{A}$;
     if $\Delta_0 = \emptyset$ then $\Delta_0 := \{\tau\}$;
     set $\Delta := \Delta_0$, $\mathcal{P}' := \mathsf{Sat}(\mathcal{P})$;
     for $x \in \Delta$, set $H(x) := \mathcal{P}' \cup \{p \mid p(x) \in \mathcal{A}\}$;
  2. for every $x \in \Delta$ reachable from $\Delta_0$ and for every formula
     $\varphi \in H(x)$
      (a) case $\varphi = \langle\sigma\rangle\psi$ : if $Next(x, \langle\sigma\rangle\psi)$ is not defined then
              $Next(x, \langle\sigma\rangle\psi) := \mathsf{Find}(\mathsf{Sat}(\{\psi\}) \cup \mathsf{Trans}(H(x), \sigma) \cup \mathcal{P}')$;
      (b) case $\varphi = [\sigma]\psi$ :
            i. for every $y \in \Delta_0$ such that $\sigma^{\mathcal{M}}(x, y)$ holds and $\psi \notin H(y)$
                 $H(y) := H(y) \cup \mathsf{Sat}(\{\psi\})$;
           ii. for every $y \in \Delta \setminus \Delta_0$ such that $\sigma^{\mathcal{M}}(x, y)$ holds and
               $\psi \notin H(y)$
                A. $y_* := \mathsf{Find}(H(y) \cup \mathsf{Sat}(\{\psi\}))$;
                B. for every $\xi$ such that $Next(x, \langle\sigma\rangle\xi) = y$
                       $Next(x, \langle\sigma\rangle\xi) := y_*$;
      (c) case $\varphi = (\psi \to \xi)$ : if $x \in \psi^{\mathcal{M}}$ and $Next(y, \langle\sigma\rangle\top)$ is defined
          for every $y$ reachable from $x$ and every $\sigma \in \mathcal{MOD}$ then
            i. if $x \in \Delta_0$ then $H(x) := H(x) \cup \mathsf{Sat}(\{\xi\})$
           ii. else
                A. $x_* := \mathsf{Find}(H(x) \cup \mathsf{Sat}(\{\xi\}))$;
                B. for every $y$, $\sigma$, $\zeta$ such that $Next(y, \langle\sigma\rangle\zeta) = x$
                       $Next(y, \langle\sigma\rangle\zeta) := x_*$;
  3. while some change occurred, go to Step 2;
  4. delete from $\Delta$ every $x$ unreachable from $\Delta_0$ and delete from $H$
     and $Next$ all elements related with such an $x$.
```

Fig. 2. Algorithm constructing a least SPDL model for an HSPDL logic program and an ABox.

2. φ is of the form $[\sigma]\psi$:
   we would like to add ψ to $H(y)$ for every y such that $\sigma^{\mathcal{M}}(x, y)$. We do this for the case when $y \in \Delta_0$. For $y \in \Delta \setminus \Delta_0$, however, modifying $H(y)$ has two drawbacks:
   - first, other objects connected to y will be affected (e.g., if p is added to $H(y)$ and $\sigma_2^{\mathcal{M}}(z, y)$ holds, then $\langle\sigma_2\rangle p$ becomes satisfied for z, while x and z may be independent),
   - second, modifying $H(y)$ may cause $H(y) = H(y')$ for some $y' \in \Delta \setminus \Delta_0$ different from y, which we try to avoid.

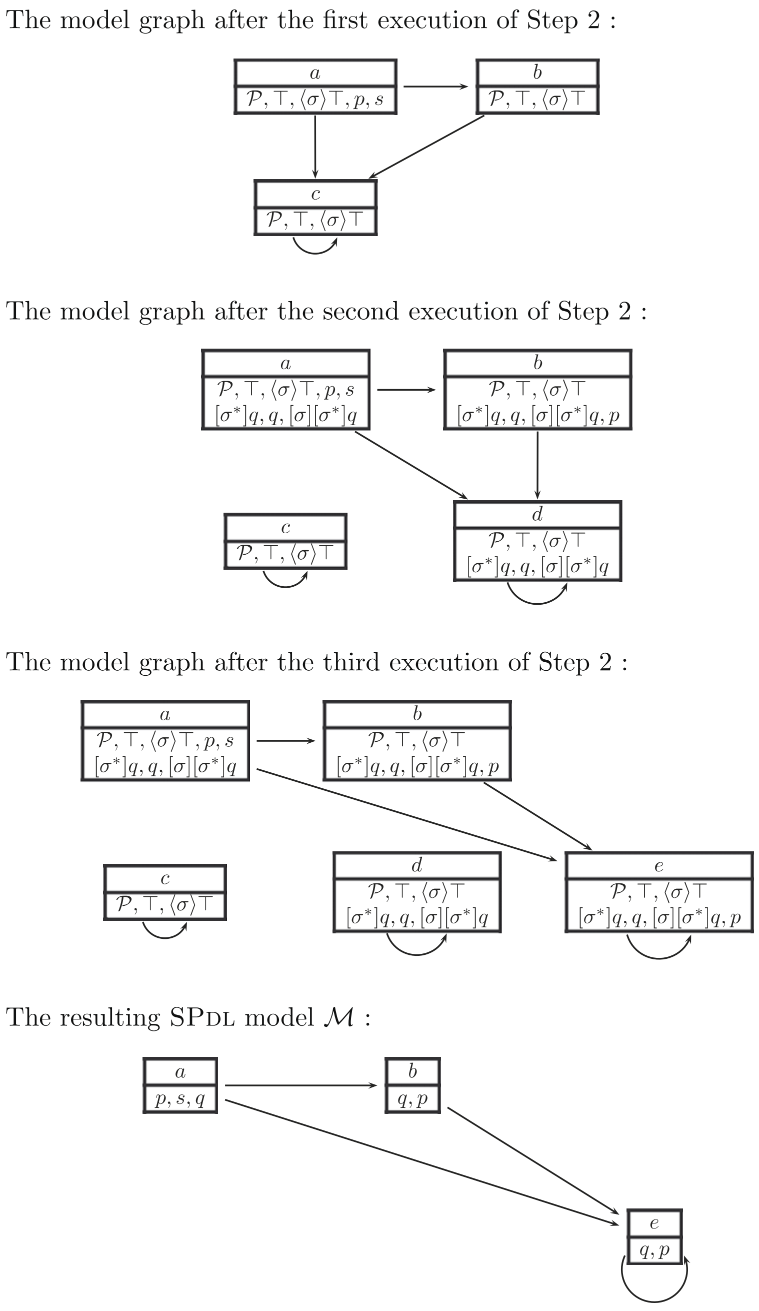
   As a solution, instead of modifying $H(y)$ we replace σ-transitions (x, y) by σ-transitions (x, y_*), where y_* is the object such that

$$H(y_*) = H(y) \cup \mathsf{Sat}(\{\psi\})$$

3. φ is of the form $\psi \to \xi$ (where ψ is a positive formula):
   if ψ "must hold"[3] for x then we would like to add ξ to $H(x)$. We do this for the case $x \in \Delta_0$. For the case $x \in \Delta \setminus \Delta_0$, analogously to the case when φ is of the form $[\sigma]\zeta$, we do not modify $H(x)$, but replace transitions (y, x) by transitions (y, x_*), where x_* is the object such that

$$H(x_*) = H(x) \cup \mathsf{Sat}(\{\xi\}).$$

[3] The statement "ψ must hold for x" intuitively means that "ψ follows from $H(x)$". As it can be seen later, a sufficient condition for the truth of this statement is that $x \in \psi^{\mathcal{M}}$ and $Next(y, \langle\sigma\rangle\top)$ is defined for every y reachable from x and every $\sigma \in \mathcal{MOD}$.



The model graph after the first execution of Step 2 :
a: $\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$
b: $\mathcal{P}, \top, \langle\sigma\rangle\top$
c: $\mathcal{P}, \top, \langle\sigma\rangle\top$
The model graph after the second execution of Step 2 :
a: $\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$, $[\sigma^*]q, q, [\sigma][\sigma^*]q$
b: $\mathcal{P}, \top, \langle\sigma\rangle\top$, $[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
c: $\mathcal{P}, \top, \langle\sigma\rangle\top$
d: $\mathcal{P}, \top, \langle\sigma\rangle\top$, $[\sigma^*]q, q, [\sigma][\sigma^*]q$
The model graph after the third execution of Step 2 :
a: $\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$, $[\sigma^*]q, q, [\sigma][\sigma^*]q$
b: $\mathcal{P}, \top, \langle\sigma\rangle\top$, $[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
c: $\mathcal{P}, \top, \langle\sigma\rangle\top$
d: $\mathcal{P}, \top, \langle\sigma\rangle\top$, $[\sigma^*]q, q, [\sigma][\sigma^*]q$
e: $\mathcal{P}, \top, \langle\sigma\rangle\top$, $[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
The resulting SPDL model $\mathcal{M}$:
a: p, s, q
b: q, p
e: q, p

Fig. 3. An illustration of the run of the algorithm shown in Fig. 2 for $\mathscr{P} = \{p \to [\sigma^*]q, [\sigma^*]q \to p\}$ and $\mathscr{A} = \{p(a), s(a), \sigma(a,b)\}$. We have that $\Delta_0 = \{a, b\}$. In the shown model graphs, an edge from a node x to a node y means $Next(x, \langle\sigma\rangle\top) = y$. The edges in the resulting model $\mathscr{M}$ represent the similarity relation $\sigma^{\mathscr{M}}$.

Example 3.10. Let $\mathscr{P} = \{p \to [\sigma^*]q, [\sigma^*]q \to p\}$ and $\mathscr{A} = \{p(a), s(a), \sigma(a,b)\}$. In Fig. 3 we illustrate the construction of a least SPDL model of $\mathscr{P}$ and $\mathscr{A}$.

Before we formally prove properties of the algorithm we need the following definitions.

Definition 3.11. The *Fischer–Ladner closure of an HSPDL program clause* φ, denoted by $\mathsf{FL}(\varphi)$, is the set of formulas defined as follows[4]:

[4] We treat an HSPDL program clause of the form $\psi \to \xi$ not as a usual formula, and our definition of Fischer–Ladner closure is slightly different from the traditional one given in [16].

$$\mathsf{FL}(\top) = \{\top\}, \quad \mathsf{FL}(p) = \{p\}$$
$$\mathsf{FL}(\psi \to \xi) = \{\psi \to \xi\} \cup \mathsf{FL}(\xi)$$
$$\mathsf{FL}(\psi \land \xi) = \{\psi \land \xi\} \cup \mathsf{FL}(\psi) \cup \mathsf{FL}(\xi)$$
$$\mathsf{FL}([\alpha]\psi) = \mathsf{FL}^{\Box}([\alpha]\psi) \cup \mathsf{FL}(\psi)$$
$$\mathsf{FL}(\langle\alpha\rangle\psi) = \mathsf{FL}^{\Diamond}(\langle\alpha\rangle\psi) \cup \mathsf{FL}(\psi)$$
$$\mathsf{FL}^{\Box}([\sigma]\psi) = \{[\sigma]\psi\}$$
$$\mathsf{FL}^{\Box}([\alpha\,;\,\beta]\psi) = \{[\alpha\,;\,\beta]\psi\} \cup \mathsf{FL}^{\Box}([\alpha][\beta]\psi) \cup \mathsf{FL}^{\Box}([\beta])\psi$$
$$\mathsf{FL}^{\Box}([\alpha \cup \beta]\psi) = \{[\alpha \cup \beta]\psi\} \cup \mathsf{FL}^{\Box}([\alpha]\psi) \cup \mathsf{FL}^{\Box}([\beta])\psi$$
$$\mathsf{FL}^{\Box}([\alpha^*]\psi) = \{[\alpha^*]\psi\} \cup \mathsf{FL}^{\Box}([\alpha][\alpha^*]\psi)$$
$$\mathsf{FL}^{\Box}([\psi?]\xi) = \{[\psi?]\xi, (\psi \to \xi)\}$$
$$\mathsf{FL}^{\Diamond}(\langle\sigma\rangle\psi) = \{\langle\sigma\rangle\psi\}$$
$$\mathsf{FL}^{\Diamond}(\langle\alpha\,;\,\beta\rangle\psi) = \{\langle\alpha\,;\,\beta\rangle\psi\} \cup \mathsf{FL}^{\Diamond}(\langle\alpha\rangle\langle\beta\rangle\psi) \cup \mathsf{FL}^{\Diamond}(\langle\beta\rangle)\psi$$
$$\mathsf{FL}^{\Diamond}(\langle\psi?\rangle\xi) = \{\langle\psi?\rangle\xi\} \cup \mathsf{FL}(\psi)$$

Definition 3.12. Let $\mathscr{P}$ be an HSPDL logic program. The *Fischer–Ladner closure of* $\mathscr{P}$, denoted by $\mathsf{FL}(\mathscr{P})$, is defined to be $\bigcup_{\varphi\in\mathscr{P}}\mathsf{FL}(\varphi)$.

It can be shown that the size of $\mathsf{FL}(\mathscr{P})$ is quadratic in the size of $\mathscr{P}$ (cf. Lemma 6.3 in [16]).

Lemma 3.13. *Let $\mathscr{M}$ be the model constructed by algorithm in Fig. 2 for $\mathscr{P}$ and $\mathscr{A}$. Assume that $\mathscr{P}$ is fixed, while $\mathscr{A}$ varies and has n assertions. Then $\Delta^{\mathscr{M}}$ has size $O(n)$ and the algorithm runs in $O(n^4)$ steps.*

Proof. We will refer to the data structures used in the algorithm shown in Fig. 2.

Observe that the Fischer–Ladner closure of $\mathscr{P}, \mathsf{FL}(\mathscr{P})$, depends only on $\mathscr{P}$. We have that $H(x) \subseteq \mathsf{FL}(\mathscr{P})$ for all $x \in \Delta \setminus \Delta_0$.[5] Since $\mathscr{P}$ is fixed and each $x \in \Delta \setminus \Delta_0$ has a unique $H(x)$, the set $\Delta \setminus \Delta_0$ contains only $O(1)$ elements. Hence Δ has size $O(n)$.

Note that the size of $H(x)$ for $x \in \Delta \setminus \Delta_0$ and the size of $H(a) \setminus \{p \mid p(a) \in \mathscr{A}\}$ for $a \in \Delta_0$ are bounded by a constant. Denote this assertion by $(*)$.

The total number of changes made at Steps 2a, 2(b)i, 2(c)i is $O(n)$. Note that if y is "simulated" by y_* at Step 2(b)ii then $H(y_*)$ extends $H(y)$. A similar statement can be said for x and x_* at Step 2(c)ii. Since $\Delta \setminus \Delta_0$ has size $O(1)$ and Δ has size $O(n)$ and $(*)$, the total number of times that Steps 2(b)ii and 2(c)ii make a change is $O(n)$. Hence, the loop at Step 3 executes only $O(n)$ times.

By $(*)$, the calls of Sat and Trans can be done in constant time. Each execution of Steps 2a, 2(b)i, 2(b)ii, 2(c)i, or 2(c)ii runs in time $O(n)$. By Lemma 2.3, checking $x \in \psi^{\mathscr{M}}$ at Step 2c runs in time $O(n^3 \times \mathit{length}(\psi)) = O(n^3)$. Checking the remaining part of the condition at Step 2c takes less time.

Summing up, the algorithm shown in Fig. 2 runs in $O(n^4)$ steps. □

Lemma 3.14. *The Kripke structure $\mathscr{M}$ constructed by the algorithm shown in Fig. 2 for $\mathscr{P}$ and $\mathscr{A}$ is a serial Kripke model of $\mathscr{P}$ and $\mathscr{A}$.*

Proof. Below we refer to the data structures used in the algorithm shown in Fig. 2. Observe that:

- if $\langle\sigma\rangle\psi \in H(x)$ then there exists y such that $\sigma^{\mathscr{M}}(x,y)$ holds and $\psi \in H(y)$,
- if $[\sigma]\psi \in H(x)$ and $\sigma^{\mathscr{M}}(x,y)$ holds then $\psi \in H(y)$,
- if $(\psi \to \xi) \in H(x)$ and $x \in \psi^{\mathscr{M}}$ then $\xi \in H(x)$.

These observations together with the definition of the saturation operator Sat and the fact that $H(x) = \mathsf{Sat}(H(x))$ for $x \in \Delta$ imply that: for every $x \in \Delta$ and every $\varphi \in H(x), x \in \varphi^{\mathscr{M}}$. (This can be proved by induction on the structure of φ.) Hence $\mathscr{M}$ is a model of $\mathscr{P}$ and $\mathscr{A}$. It is a serial Kripke structure because $\langle\sigma\rangle\top$ is included in $H(x)$ for every $\sigma \in \mathscr{MOD}$ and every $x \in \Delta$. □

Roughly speaking, the model $\mathscr{M}$ constructed by the algorithm shown in Fig. 2 for $\mathscr{P}$ and $\mathscr{A}$ is less than or equal to any model $\mathscr{M}'$ of $\mathscr{P}$ and $\mathscr{A}$ in SPDL because the objects of $\mathscr{M}$ are created only when necessary (cf. Condition 2 of Definition 3.2) with minimal sets $H(\_)$ of requirements (cf. Conditions 1 and 4 of Definition 3.2), which contain $\langle\sigma\rangle\top$ for all $\sigma \in \mathscr{MOD}$ (to guarantee Condition 3 of Definition 3.2). A formal analysis is given below.

Lemma 3.15. *Let $\mathscr{M}$ be the model constructed by the algorithm shown in Fig. 2 for $\mathscr{P}$ and $\mathscr{A}$, and $\mathscr{M}'$ be an arbitrary serial Kripke model of $\mathscr{P}$ and $\mathscr{A}$. Consider a moment after executing a numerated step[6] in the execution of the algorithm. Let*

$$r = \{(a, a^{\mathscr{M}'}) \mid a \text{ is an individual occurring in } \mathscr{A}\} \cup \{(x, x') \in \Delta^{\mathscr{M}} \times \Delta^{\mathscr{M}'} \mid x \text{ is not an individual and } \mathscr{M}', x' \models H(x)\}$$

[5] Since the ABox $\mathscr{A}$ is extensionally reduced, the assertions of $\mathscr{A}$ are not transferred to $H(x)$ for $x \in \Delta \setminus \Delta_0$.

[6] That is, one of 1, 2, 2b, 2(b)i, 2(b)ii, 2(b)iiA, 2(b)iiB, 2c, 2(c)i, 2(c)ii, 2(c)iiA, 2(c)iiB, 3, 4.

Then for every $u, v \in \Delta^{\mathcal{M}},\ u', v' \in \Delta^{\mathcal{M}'},\ \sigma \in \mathcal{MOD}$, every formula ζ and every individual a occurring in $\mathcal{A}$ the following assertions hold:

$$[r(u,u') \wedge (Next(u, \langle\sigma\rangle\zeta) = v) \wedge \sigma^{\mathcal{M}'}(u', v') \wedge v' \in \zeta^{\mathcal{M}'}] \rightarrow r(v, v') \tag{6}$$

$$\mathcal{M}', a^{\mathcal{M}'} \models H(a) \tag{7}$$

Proof. We prove this lemma by induction on the number of executed steps. The base case occurs after executing Step 1 and the assertions clearly hold. Consider some latter enumerated step K of the algorithm. Inductively assume that the assertions of the lemma hold before executing that step. We first prove the following remark by an inner induction on the construction of ψ.

Remark 3.16.

Let ψ be a positive formula. Suppose that $r(x,x')$ holds, $x \in \psi^{\mathcal{M}}$, and $Next(y, \langle\sigma\rangle\top)$ is defined for every y reachable from x and every $\sigma \in \mathcal{MOD}$. Then $x' \in \psi^{\mathcal{M}'}$.

Proof (*of Remark 3.16*). Let $\mathcal{M}_x$ (respectively, Δ_x) be the Kripke structure obtained by restricting $\mathcal{M}$ (respectively, Δ) to the objects reachable for x. We show that:

$$\forall u, v \in \Delta_x \forall u' \in \Delta^{\mathcal{M}'} \left[[\sigma^{\mathcal{M}}(u, v) \wedge r(u,u')] \rightarrow \exists v' \in \Delta^{\mathcal{M}'}[\sigma^{\mathcal{M}'}(u', v') \wedge r(v, v')]\right] \tag{8}$$

$$\forall u \in \Delta_x \forall u', v' \in \Delta^{\mathcal{M}'} \left[[\sigma^{\mathcal{M}'}(u', v') \wedge r(u,u')] \rightarrow \exists v \in \Delta_x[\sigma^{\mathcal{M}}(u, v) \wedge r(v, v')]\right] \tag{9}$$

Consider assertion (8). Let $u, v \in \Delta_x$ and suppose that $\sigma^{\mathcal{M}}(u, v)$ and $r(u,u')$ hold. If $v \in \Delta_0$ (i.e. v is an individual occurring in $\mathcal{A}$) then $\sigma(u, v) \in \mathcal{A}$ and $u \in \Delta_0$. In that case, take $v' = v^{\mathcal{M}'}$ and we have that $\sigma^{\mathcal{M}'}(u', v') \wedge r(v, v')$. Consider the case $v \notin \Delta_0$. Since $\sigma^{\mathcal{M}}(u, v)$, we must have that $Next(u, \langle\sigma\rangle\zeta) = v$ for some ζ. Thus $\langle\sigma\rangle\zeta \in H(u)$. Since $r(u,u')$, it follows that $u' \in (\langle\sigma\rangle\zeta)^{\mathcal{M}'}$. Hence, there exists $v' \in \Delta^{\mathcal{M}'}$ such that $\sigma^{\mathcal{M}'}(u', v')$ and $v' \in \zeta^{\mathcal{M}'}$. By the inductive assumption (6) of the outer induction, we have that $r(v, v')$ holds.

Consider assertion (9). Let $u \in \Delta_x$ and suppose that $\sigma^{\mathcal{M}'}(u', v')$ and $r(u,u')$ hold. Let $v = Next(u, \langle\sigma\rangle\top)$ (which exists by the assumption). Thus $\sigma^{\mathcal{M}}(u, v)$ holds. By the inductive assumption (6) of the outer induction, we have that $r(v, v')$ holds.

Applying Lemma 3.4 for $\mathcal{M}_x, \mathcal{M}'$ and r, we obtain that $x' \in \psi^{\mathcal{M}'}$, which completes the proof of Remark 3.16. □

Proof (*of Lemma 3.15 continued*). We now return to the outer induction. Suppose that after executing the step K we have $r_2,\ \Delta_2,\ H_2,\ Next_2,\ \mathcal{M}_2$ in the places of $r,\ \Delta,\ H,\ Next,\ \mathcal{M}$, respectively. We show that the following conditions hold for every $u, v \in \Delta_2,\ u', v' \in \Delta',\ \sigma \in \mathcal{MOD}$, every formula ζ and every individual a occurring in $\mathcal{A}$:

$$[r_2(u,u') \wedge (Next_2(u, \langle\sigma\rangle\zeta) = v) \wedge \sigma^{\mathcal{M}'}(u', v') \wedge v' \in \zeta^{\mathcal{M}'}] \rightarrow r_2(v, v') \tag{10}$$

$$\mathcal{M}', a^{\mathcal{M}'} \models H_2(a) \tag{11}$$

It suffices to consider Steps 2a, 2(b)i, 2(b)iiB, 2(c)i, 2(c)iiB.

- Consider the case $K = 2a$ and suppose that $Next(x, \langle\sigma\rangle\psi)$ is not defined. Let

$$y = \mathsf{Find}(\mathsf{Sat}(\{\psi\}) \cup \mathsf{Trans}(\mathsf{H}(x), \sigma) \cup \mathcal{P}')$$

It suffices to prove the assertion (10) for the case when $u = x,\ \ v = y$ and $\zeta = \psi$. Consider this case. Suppose that

$$r_2(x,u') \wedge (Next_2(x, \langle\sigma\rangle\psi) = y) \wedge \sigma^{\mathcal{M}'}(u', v') \wedge v' \in \psi^{\mathcal{M}'}$$

holds. We show that $r_2(y, v')$ holds. Since $r_2(x,u')$ holds, $r(x,u')$ also holds, and hence $\mathcal{M}', u' \models H(x)$. It follows that $\mathcal{M}', v' \models \mathsf{Trans}(H(x), \sigma)$ (since $\sigma^{\mathcal{M}'}(u', v')$ holds). Hence $\mathcal{M}', v' \models H_2(y)$, and $r_2(y, v')$ holds.
- Consider the case K = 2(b)i. Since $y \in \Delta_0$, we have that $r_2 = r$. Additionally, $Next_2 = Next$, hence the assertion (10) follows from the inductive assumption (6). For the assertion (11), it suffices to consider the case $y = a$ and show that $\mathcal{M}',\ a^{\mathcal{M}'} \models \psi$. Since $\sigma^{\mathcal{M}}(x,a)$ holds, x must be an individual occurring in $\mathcal{A}$ and $\sigma(x,a) \in \mathcal{A}$. By the inductive assumption (7), $\mathcal{M}',\ x^{\mathcal{M}'} \models H(x)$, and hence $\mathcal{M}',\ x^{\mathcal{M}'} \models [\sigma]\psi$. Since $\mathcal{M}'$ is a model of $\mathcal{A}$ and $\sigma(x,a) \in \mathcal{A}$, it follows that $\mathcal{M}',\ a^{\mathcal{M}'} \models \psi$.
- Consider the case K = 2(b)iiB. It suffices to show that if

$$r(x,x') \wedge (Next(x, \langle\sigma\rangle\xi) = y) \wedge \sigma^{\mathcal{M}'}(x',y') \wedge y' \in \xi^{\mathcal{M}'}$$

  then $r_2(y_*,y')$ holds. Suppose that the premise holds. By the inductive assumption (6), $r(y,y')$ holds and $\mathcal{M}',\ y' \models H(y)$. Since $r(x,x')$ holds, $\mathcal{M}',\ x' \models H(x)$. It follows that $\mathcal{M}',\ y' \models \psi$ (since $[\sigma]\psi \in H(x)$ and $\sigma^{\mathcal{M}'}(x',y')$ holds). Therefore $\mathcal{M}',\ y' \models H_2(y_*)$ and $r_2(y_*,y')$ holds.
- Consider the case K = 2(c)i. Since $r_2 = r$ and $Next_2 = Next$, the assertion (10) follows from the inductive assumption (6). For the assertion (11), it suffices to consider the case $x = a$ and show that $\mathcal{M}',\ a^{\mathcal{M}'} \models \xi$. Since $a \in \psi^{\mathcal{M}}$, by Remark 3.16, $a^{\mathcal{M}'} \in \psi^{\mathcal{M}'}$. By the inductive assumption (7), $\mathcal{M}',\ a^{\mathcal{M}'} \models H(a)$, and hence $\mathcal{M}',\ a^{\mathcal{M}'} \models (\psi \rightarrow \xi)$. It follows that $\mathcal{M}',\ a^{\mathcal{M}'} \models \xi$.

- Consider the case $K = 2(c)$iiB. Let x' be an element of $\Delta^{\mathcal{M}'}$ such that $r(x, x')$ holds. It suffices to show that $r_2(x_*, x')$ holds. We need only to show that $\mathcal{M}', x' \models \xi$. Since $r(x, x')$ holds and $(\psi \to \xi) \in H(x)$, we have that $\mathcal{M}', x' \models (\psi \to \xi)$. Since $x \in \psi^{\mathcal{M}}$ and $r(x, x')$ holds, by Remark 3.16, $x' \in \psi^{\mathcal{M}'}$. It follows that $\mathcal{M}', x' \models \xi$. □

Corollary 3.17. *Let $\mathcal{P}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{M}'$ be as in Lemma 3.15, and r be the relation defined as in Lemma 3.15 after the execution of the algorithm shown in Fig. 2. Then $\mathcal{M} \leqslant_r \mathcal{M}'$.*

Proof. We check the four conditions of $\mathcal{M} \leqslant_r \mathcal{M}'$. Condition 1 of $\mathcal{M} \leqslant_r \mathcal{M}'$ follows from the definition of r. Condition 4 of $\mathcal{M} \leqslant_r \mathcal{M}'$ follows from the definition of r and the assertion (7) of Lemma 3.15. Analogously as in the proof of Remark 3.16, it can be shown that Conditions 2 and 3 of $\mathcal{M} \leqslant_r \mathcal{M}'$ follow from the assertion (6) of Lemma 3.15. □

The following theorem is crucial for the querying machinery developed in this paper. Recall that the least model $\mathcal{M}$ has the property that for every positive formula φ and for every individual a, we have that $\mathcal{P}, \mathcal{A} \models_s \varphi(a)$ iff $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$. The model is then used to compute answers to queries.

Theorem 3.18. *Let $\mathcal{P}$ be an HSP<span style="font-variant:small-caps">dl</span> logic program and $\mathcal{A}$ an ABox. The Kripke structure $\mathcal{M}$ constructed by the algorithm shown in Fig. 2 for $\mathcal{P}$ and $\mathcal{A}$ is a least SP<span style="font-variant:small-caps">dl</span> model of $\mathcal{P}$ and $\mathcal{A}$.*

This theorem immediately follows from Lemma 3.14 and Corollary 3.17. As a consequence, we obtain the following result, showing that the proposed querying machinery is tractable.

Theorem 3.19. *Let $\mathcal{P}$ be an HSPDL logic program, $\mathcal{A}$ an ABox, φ a positive formula, and a an individual. Then checking $(\mathcal{P}, \mathcal{A}) \models_s \varphi(a)$ can be done in polynomial time in the size of $\mathcal{A}$. That is, the data complexity of HSPDL is in PTIME.*

Proof. Let $\mathcal{M}$ be the model constructed by the algorithm shown in Fig. 2 for $\mathcal{P}$ and $\mathcal{A}$. By Theorem 3.18, $(\mathcal{P}, \mathcal{A}) \models_s \varphi(a)$ iff $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$. Constructing $\mathcal{M}$ and checking $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$ both can be done in polynomial time the size of $\mathcal{A}$ (Lemmas 3.13 and 2.3). □

4. A case study: movability of objects

4.1. The scenario

Consider the following scenario:

Two robots, R_1 and R_2, have the goal to move objects from one place to another. Each robot is able to move objects of a specific signature,[7] and together they might be able to move objects of a combined signature. Some objects, when attempted to be moved, may cause some damages for robots. Robots are working independently, but sometimes have to cooperate to achieve their goals.

To design such robots one has to make a number of decisions as described below.

4.2. Formalizing movability of objects

We assume that the signature of movable objects for each robot is given by its specification together with a similarity relation defining the range of movable objects. Assume the following specification:

$$spec_1 \stackrel{\text{def}}{\equiv} (light \land smooth) \lor (heavy \land rough) \text{ — for robot } R_1 \tag{12}$$

$$spec_2 \stackrel{\text{def}}{\equiv} small \lor medium \text{ — for robot } R_2. \tag{13}$$

Movable objects are then specified by

$$spec_i \to movable_i \tag{14}$$

where $i \in \{1, 2\}$ and $movable_i$ is true for objects that can be moved by R_i.

The idea is that all objects similar to movable ones are movable too.[8] Let σ_1 and σ_2 be similarity relations reflecting perceptual capabilities of R_1 and R_2, respectively (for a discussion of such similarity relations based on various sensor models see [8]). Now, in addition to (14), movable objects are characterized by

$$\langle \sigma_i \rangle spec_i \to movable_i \tag{15}$$

Remark 4.1. Note that rather than (15) one could assume

$$[\sigma_i] spec_i \to movable_i$$

[7] For example, dependent on weight, size and type of surface.

[8] This is a very natural and quite powerful technique, allowing one to express the inheritance of particular properties of objects by similar objects.

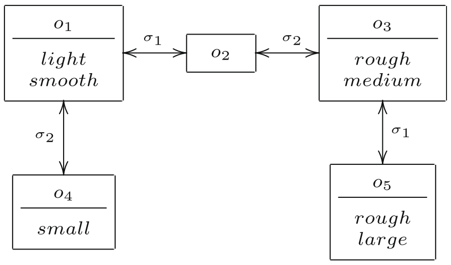


Fig. 4. An ABox for the example considered in Section 4.3.

In addition to (14), this would mean considering objects similar only to movable ones. In some applications this choice would indeed be reasonable and perhaps less risky.

Observe that in general it is impossible to automatically derive combined signatures that specify what robots can move together. Therefore, we introduce specification $spec_3$ and similarity expression α_3 as a specification of such joint capabilities. An example of $spec_3$ can be given by

$$spec_3 \overset{\text{def}}{\equiv} large \wedge rough \tag{16}$$

Objects movable by robots working together are then defined by

$$(spec_3 \vee \langle\alpha_3\rangle spec_3) \rightarrow movable\_by\_two \tag{17}$$

Observe that α_3 is usually computed on the basis of σ_1 and σ_2, since we do not assume any observer other than $R_1,\ R_2$, and $\sigma_1,\ \sigma_2$ reflect their perceptual capabilities. We shall assume that

$$\alpha_3 \overset{\text{def}}{=} \sigma_1 \cup \sigma_2 \cup (\sigma_1;\sigma_2) \cup (\sigma_2;\sigma_1) \tag{18}$$

The meaning of this expression is that an object o' is similar w.r.t. α_3 to an object o whenever

- o' is perceived similar to o by R_1 or by R_2, or
- there is an object o'' such that
  - o'' is perceived similar to o by R_1 and o'' is perceived similar to o'' by R_2, or
  - o'' is perceived similar to o by R_2 and o'' is perceived similar to o'' by R_1.

Clearly, one can use much more complex expressions, reflecting particular algorithms for computing α_3, since our operators are those accepted in PDL.[9]

4.3. The database

Let $\mathscr{A}$ be the ABox consisting of the assertions about objects $o_1, \ldots, o_5$ that are illustrated in Fig. 4. It contains, for example, $light(o_1),\ smooth(o_1),\ \sigma_1(o_1,o_2),\ \sigma_1(o_2,o_1)$, etc. Let $\mathscr{P}$ be the HSPDL logic program consisting of the clauses (14) and (15) for $i \in \{1,2\}$, and (17). We consider here the database consisting of $\mathscr{P}$ and $\mathscr{A}$ in SPDL (which adopts the axioms $\langle\sigma_1\rangle\top$ and $\langle\sigma_2\rangle\top$ for all the objects of the domain).

In Fig. 5 we present the least SPDL model $\mathscr{M}$ of $\mathscr{P},\ \mathscr{A}$ constructed by our algorithm (given in Fig. 2). The object o_6 is the only additional object, not satisfying any proposition.

4.4. Some queries

As discussed earlier, having the least SPDL model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$, to check whether an individual a has a positive property φ w.r.t. $\mathscr{P}$ and $\mathscr{A}$ in SPDL, it suffices to check whether $a \in \varphi^{\mathscr{M}}$.

In our example, we have that:

$$movable_1^{\mathscr{M}} = \{o_1, o_2\}$$
$$movable_2^{\mathscr{M}} = \{o_1, o_2, o_3, o_4\}$$
$$movable\_by\_two^{\mathscr{M}} = \{o_2, o_3, o_5\}$$
$$(movable\_by\_two \wedge \langle\sigma_1\rangle movable_1)^{\mathscr{M}} = \{o_2\}$$
$$(movable\_by\_two \wedge [\sigma_1] movable_1)^{\mathscr{M}} = \emptyset$$

[9] Of course, there are some restrictions, if one wants to stay in a tractable framework. Recall that we have accepted Definition 2.10 to guarantee tractability.

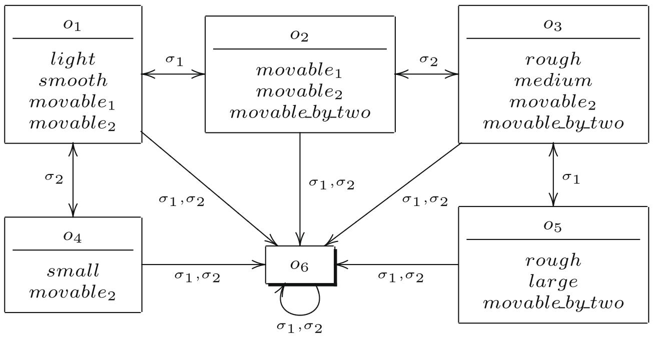


Fig. 5. A least SPDL model of the database considered in Section 4.3.

5. Using SPDL in epistemic reasoning

5.1. Epistemic interpretation of approximations

SPDL can be used for reasoning about common knowledge and common beliefs in the presence of similarity relations. Intuitively, an agent believes that a formula φ holds when $[\sigma]\varphi$, where σ reflects the agent's perception. This allows us to integrate agents' epistemic reasoning with the formalism of SPDL. Technically, individuals and objects of Kripke structures are used to denote possible worlds, while similarity relations are used to denote accessibility relations between possible worlds of agents and groups of agents. We assume here that each similarity relation symbol σ reflects the accessibility relation of a single agent.

Consider a group of k agents with $\{\sigma_1, \ldots, \sigma_k\}$ as their accessibility relations, respectively. Then:

- an expression of the form $\sigma_1 \cup \cdots \cup \sigma_k$ represents the integrated accessibility relation of the group of agents
- a formula of the form $[(\sigma_1 \cup \cdots \cup \sigma_k)^*]\varphi$ states that φ is a piece of common knowledge of the group of agents
- a formula of the form $[(\sigma_1 \cup \cdots \cup \sigma_k)^+]\varphi$, where $\alpha^+ = (\alpha\, ;\, \alpha^*)$, states that φ is a common belief of the group.[10]

The seriality axiom (1) can be reformulated as

$$[\sigma]\varphi \rightarrow \neg[\sigma]\neg\varphi$$

so in epistemic reasoning it states that if an agent believes that φ holds then it does not believe $\neg\varphi$. As the transitivity of σ is not assumed,[11] to express positive introspection of knowledge (respectively, belief) one can use σ^* (respectively, σ^+) as a basic accessibility relation instead of σ because $[\sigma^*]\varphi \rightarrow [\sigma^*][\sigma^*]\varphi$ (respectively, $[\sigma^+]\varphi \rightarrow [\sigma^+][\sigma^+]\varphi$) is valid in every Kripke structure. On the other hand, no suitable form reflecting the Euclidicity of σ for expressing negative introspection is a tautology of SPDL (i.e. valid in every serial Kripke structure).[12]

When SPDL is used in epistemic reasoning, individuals and ABoxes do not play an important role in complexity issues anymore. In typical cases, solely the actual world is explicitly used as an individual.

5.2. The wise men puzzle

In this section we formalize the wise men puzzle using HSPDL, showing again its usefulness and illustrating our algorithm given in Fig. 2.

The wise men puzzle is a famous benchmark of AI introduced by McCarthy [23]. It can be stated as follows (cf. [19]). A king wishes to know whether his three advisors (represented by[13] $\sigma_1,\ \sigma_2,\ \sigma_3$) are as wise as they claim to be. Three chairs are lined up, all facing the same direction, one behind the other. The wise men are instructed to sit down in the order $\sigma_1,\ \sigma_2,\ \sigma_3$, with σ_1 on front. Each of them can see the backs of the ones sitting before them (e.g. σ_3 can see σ_2 and σ_1). The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the color of his own card. Each man must announce the color of his own card as soon as he knows what it is. All know that this will happen. The room is silent; then, after a while, wise man σ_1 says "My card is white!".

[10] This concept of common belief corresponds to the generally recognized notion of common belief considered, e.g., in [13–15,24].

[11] Usually expressed by modal axiom **4** (see [2]).

[12] Euclidicity, i.e., the property stating that $\forall x \forall y \forall z[(\sigma(x,y) \wedge \sigma(x,z)) \rightarrow \sigma(y,z)]$, usually accepted in modal epistemic reasoning, is expressed by the modal axiom **5** (see [2]).

[13] In the rest of example to simplify the presentation we denote agents by their similarity relations.

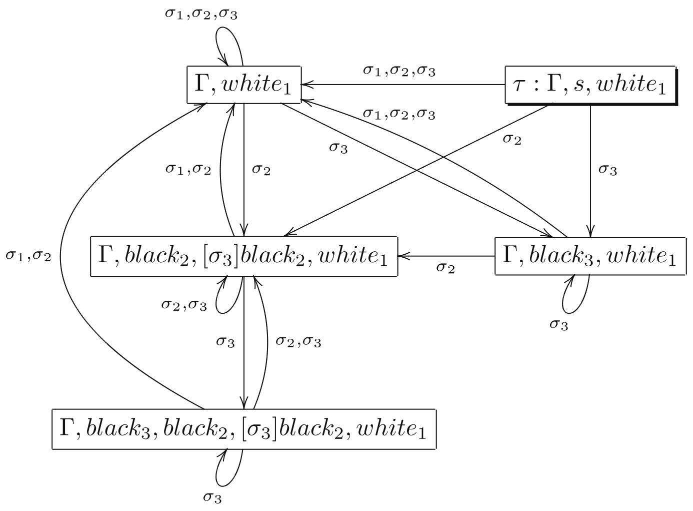

$\sigma_1,\sigma_2,\sigma_3$
$\Gamma, white_1$
$\sigma_1,\sigma_2,\sigma_3$
$\tau : \Gamma, s, white_1$
$\sigma_1,\sigma_2,\sigma_3$
σ_3
σ_2
σ_1,σ_2
σ_2
σ_3
σ_1,σ_2
$\Gamma, black_2, [\sigma_3]black_2, white_1$
σ_2
$\Gamma, black_3, white_1$
σ_2,σ_3
σ_3
σ_3
σ_2,σ_3
$\Gamma, black_3, black_2, [\sigma_3]black_2, white_1$
σ_3

Fig. 6. The model graph constructed by our algorithm (given in Fig. 2) for the database specified in Section 5.2. Γ is the set of formulas specified in Section 5.3. The node with a shaded frame represents the actual world τ. An edge from a node x to a node y with a label σ_i means $Next(x, \langle\sigma_i\rangle\xi) = y$ for some ξ, where $\langle\sigma_i\rangle\xi$ is one of $\langle\sigma_1\rangle\top$, $\langle\sigma_2\rangle\top$, $\langle\sigma_3\rangle\top$, $\langle\sigma_2\rangle black_2$, $\langle\sigma_3\rangle black_3$. The proposition $white_1$ is added to the bottom node due to the property ψ_7 (see Sections 5.3 and 5.2), and after that it is added to the other nodes due to the properties ψ_1 and ψ_2 (see Sections 5.3 and 5.2).

For $1 \leqslant i \leqslant 3$, let $white_i$ stand for "the card of σ_i is white", and $black_i$ stand for "the card of σ_i is black". The wise men puzzle can be formalized as follows (cf. [25,28]).

The wise men commonly know that if y sits behind x then x's card is white whenever y considers this possible:

$$\varphi_1 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](\langle\sigma_2\rangle white_1 \to white_1)$$
$$\varphi_2 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](\langle\sigma_3\rangle white_1 \to white_1)$$
$$\varphi_3 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](\langle\sigma_3\rangle white_2 \to white_2)$$

The following program clauses are "dual" to the above ones:

$$\varphi_4 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](black_1 \to [\sigma_2]black_1)$$
$$\varphi_5 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](black_1 \to [\sigma_3]black_1)$$
$$\varphi_6 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*](black_2 \to [\sigma_3]black_2)$$

The wise men commonly know that at least one of them has a white card:

$$\varphi_7 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]((black_2 \wedge black_3) \to white_1)$$
$$\varphi_8 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]((black_3 \wedge black_1) \to white_2)$$
$$\varphi_9 = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]((black_1 \wedge black_2) \to white_3)$$

The wise men commonly know that: each of σ_2 and σ_3 does not know the color of his own card; in particular, each of them considers that it is possible that his own card is black:

$$\varphi_{10} = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]\langle\sigma_2\rangle black_2$$
$$\varphi_{11} = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]\langle\sigma_3\rangle black_3$$

The formulas $\varphi_1, \ldots, \varphi_9$ are supposed to hold for every possible world, while the formulas φ_{10} and φ_{11} are only supposed to hold for the actual world. Since only extensionally reduced ABoxes are allowed, we encode the conjunction $\varphi_{10} \wedge \varphi_{11}$ by a proposition s, and assume that our ABox $\mathscr{A}$ is $\{s(\tau)\}$, where τ is the only individual which represents the actual world, and we treat $s \to (\varphi_{10} \wedge \varphi_{11})$ as a global assumption. Thus, our database consists of the mentioned ABox $\mathscr{A}$ and the HSPDL logic program $\mathscr{P} = \{\varphi_1, \ldots, \varphi_9, (s \to (\varphi_{10} \wedge \varphi_{11}))\}$.

The goal is to check whether wise man σ_1 believes that his card is white: that is, whether $([\sigma_1]white_1)(\tau)$ is a logical consequence of $\mathscr{P}, \mathscr{A}$ in SPDL.

5.3. The least SPDL model constructed by our algorithm

For $1 \leqslant i \leqslant 11$, let ψ_i be the formula such that $\varphi_i = [(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*]\psi_i$. Let

$$\begin{aligned}\Gamma &= \mathsf{Sat}(\{\varphi_1, \ldots, \varphi_{11}\}) \cup \{s \to (\varphi_{10} \wedge \varphi_{11})\}\\ &= \{\varphi_i, \psi_i, [\sigma_1 \cup \sigma_2 \cup \sigma_3]\varphi_i, [\sigma_j]\varphi_i \mid 1 \leqslant i \leqslant 11, 1 \leqslant j \leqslant 3\} \ \cup \{\top, \langle\sigma_1\rangle\top, \langle\sigma_2\rangle\top, \langle\sigma_3\rangle\top, (s \to (\varphi_{10} \wedge \varphi_{11}))\}\end{aligned}$$

In Fig. 6 we present the model graph constructed by our algorithm (given in Fig. 2) for $\mathscr{P}$ and $\mathscr{A}$. By Theorem 3.18, the model $\mathscr{M}$ corresponding to this model graph is a least SPDL model of $\mathscr{P}$ and $\mathscr{A}$. Since $\tau^{\mathscr{M}} \in ([\sigma_1]white_1)^{\mathscr{M}}$, we have that $([\sigma_1]white_1)(\tau)$ is a logical consequence of $\mathscr{P}, \mathscr{A}$ in SPDL. Notice also that $\tau^{\mathscr{M}} \in ([(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*][\sigma_1]white_1)^{\mathscr{M}}$. That is, the wise men commonly know (believe) that wise man σ_1 believes that his card is white.

6. Conclusions

In this paper we have presented a powerful formalism for approximate knowledge fusion, based on adaptation of Propositional Dynamic Logic. We have shown that restricting this logic to its suitably chosen Horn fragment results in tractable querying mechanism which can be applied in application, where approximate knowledge from various sources is to be fused, e.g., in robotics and multiagent systems.

Importantly, serial PDL, denoted by SPDL, is also useful as a description logic for domains where seriality condition appears naturally.[14] For example, in reasoning about properties of web pages one can assume that every considered web page has a link to another page (or to itself).

We plan to extend the framework to deal with other operations on similarity relations, expressing even more subtle approximations and fused knowledge structures. This would be applicable in different stages of teamwork in multiagent systems as discussed in [13,14].

References

[1] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), Description Logic Handbook, Cambridge University Press, 2002.
[2] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, 2001.
[3] D. Ciucci, T. Flaminio, Generalized rough approximations in $\Pi\frac{1}{2}$, Int. J. Approx. Reason. 48 (2) (2008) 544–558.
[4] T.H. Cormen, E.H. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., The MIT Press, 2001.
[5] S.P. Demri, E.S. Orłowska, Incomplete Information: Structure, Inference, Complexity, Monographs in Theoretical Computer Science An EATCS Series, Springer-Verlag, Heidelberg, 2002.
[6] P. Doherty, B. Dunin-Kęplicz, A. Szałas, Dynamics of approximate information fusion, in: M. Kryszkiewicz, J. Peters, H. Rybinski, A. Skowron (Eds.), Proceedings of the RSEISP 2007, Rough Sets and Emerging Intelligent Systems Paradigms, LNAI, vol. 4585, Springer-Verlag, 2007, pp. 668–677.
[7] P. Doherty, W. Łukaszewicz, A. Skowron, A. Szałas, Knowledge Representation Techniques. A Rough Set Approach, Studies in Fuziness and Soft Computing, vol. 202, Springer-Verlag, 2006.
[8] P. Doherty, W. Łukaszewicz, A. Szałas, Communication between agents with heterogeneous perceptual capabilities, J. Inf. Fusion 8 (1) (2007) 56–69.
[9] P. Doherty, M. Magnusson, A. Szałas, Approximate databases: a support tool for approximate reasoning, J. Appl. Non-Classical Logics 16 (1–2) (2006) 87–118. Special issue on Implementation of logics.
[10] P. Doherty, A. Szałas, On the correspondence between approximations and similarity, in: S. Tsumoto, R. Slowinski, J. Komorowski, J.W. Grzymala-Busse (Eds.), Proceedings of 4th International Conference on Rough Sets and Current Trends in Computing (RSCTC'2004), LNAI, vol. 3066, Springer, 2004, pp. 143–152.
[11] V. Dubois, M. Quafafou, Concept learning with approximation: Rough version spaces, in: J.J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing (RSCTC 2002), LNAI, vol. 2475, Springer-Verlag, 2002, pp. 239–246.
[12] B. Dunin-Kęplicz, A. Szałas, Towards approximate BGI systems, in: H.-D. Burkhard, G. Lindeman, L. Varga, R. Verbrugge (Eds.), Proceedings of the 5th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 20 07), LNAI, vol. 4696, Springer-Verlag, 2007, pp. 277–287.
[13] B. Dunin-Kęplicz, R. Verbrugge, Collective intentions, Fundam. Inform. 51 (3) (2002) 271–295.
[14] B. Dunin-Kęplicz, R. Verbrugge, A tuning machine for cooperative problem solving, Fundam. Inform. 63 (2004) 283–307.
[15] R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, MA, 1995.
[16] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, 2000.
[17] U. Hustadt, B. Motik, U. Sattler, Data complexity of reasoning in very expressive description logics, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proceedings of IJCAI-05, Professional Book Center, 2005, pp. 466–471.
[18] J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron, Rough sets: atutorial, in: Pal and Skowron [29], pp. 3–98.
[19] K. Konolige, Belief and incompleteness, Technical Report 319, SRI Inter., 1984.
[20] T.-J. Li, Y. Leung, W.-X. Zhang, Generalized fuzzy rough approximation operators based on fuzzy coverings, Int. J. Approx. Reason. 48 (3) (2008) 836–856.
[21] G. Liu, Y. Sai, A comparison of two types of rough sets induced by coverings, Int. J. Approx. Reason. 50 (3) (2009) 521–528.
[22] J. Maluszyński, A. Szałas, A. Vitória, Paraconsistent logic programs with four-valued rough sets, in: C.-C. Chan, J. Grzymala-Busse, W. Ziarko (Eds.), Proceedings of 6th International Conference on Rough Sets and Current Trends in Computing (RSCTC 2008), LNAI, vol. 5306, pp. 41–51, 2008.
[23] J. McCarthy, First order theories of individual concepts and propositions, Mach. Intell. 9 (1979) 120–147.
[24] J.-J.Ch. Meyer, W. vander Hoek, Epistemic Logic for AI and Theoretical Computer Science, Cambridge University Press, Cambridge, 1995.
[25] L.A. Nguyen, Multimodal logic programming, Theor. Comput. Sci. 360 (2006) 247–288.
[26] L.A. Nguyen, On the deterministic Horn fragment of test-free PDL, in: I. Hodkinson, Y. Venema (Eds.), Advances in Modal Logic, vol. 6, King's College Publications, 2006, pp. 373–392.
[27] L.A. Nguyen, Weakening Horn knowledge bases in regular description logics to have PTIME data complexity, in: S. Ghilardi, U. Sattler, V. Sofronie-Stokkermans, A. Tiwari (Eds.), Proceedings of Automated Deduction: Decidability, Complexity, Tractability (ADDCT'07), 2007, pp. 32–47.
[28] L.A. Nguyen, Constructing finite least Kripke models for positive logic programs in serial regular grammar logics, Logic J. IGPL 16 (2) (2008) 175–193.
[29] S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization: A New Trend in Decision-Making, Springer-Verlag, Singapore, 1999.
[30] Z. Pawlak, Rough Sets, Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, 1991.
[31] Z. Pawlak, A. Skowron, Rough sets and boolean reasoning, Inform. Sci. 177 (1) (2007) 41–73.
[32] Z. Pawlak, A. Skowron, Rough sets: some extensions, Inform. Sci. 177 (1) (2007) 28–40.
[33] Z. Pawlak, A. Skowron, Rudiments of rough sets, Inform. Sci. 177 (1) (2007) 3–27.
[34] A. Skowron, J. Stepaniuk, Tolerance approximation spaces, Fundam. Inform. 27 (1996) 245–253.

[14] Related work on Horn fragments of description logics can be found in [27]. The papers by other authors do not consider PDL but only versions of the description logic $\mathscr{SHI}$.

[35] R. Słowiński, D. Vanderpooten, Similarity relation as a basis for rough approximations, in: P. Wang (Ed.), Advances in Machine Intelligence & Soft Computing, Bookwrights, Raleigh NC, 1997, pp. 17–33.
[36] R. Słowiński, D. Vanderpooten, A generalized definition of rough approximations based on similarity, IEEE Trans. Data Knowl. Eng. 12 (2) (2000) 331–336.
[37] J. van Benthem, Correspondence theory, in: D.M. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, vol. 2, D. Reidel Pub. Co., 1984, pp. 167–247.
[38] Y.Y. Yao, Generalized rough set models, in: L. Polkowski, A. Skowron (Eds.), Rough Sets in Knowledge Discovery, 1998, pp. 286–318.
[39] Y.Y. Yao, T.Y. Lin, Generalization of rough sets using modal logic, Intell. Autom. Soft Comput. – Int. J. 2 (2) (1996) 103–120.