Tame the infinite—simplification problems for infinite-state systems
PhD dissertation in Computer Science
Radosław Piórkowski

Supervisor:
prof. dr hab. Sławomir Lasota

Auxirary supervisor:
dr hab. Wojciech Czerwiński

University of Warsaw

Faculty of Mathematics,
Informatics and Mechanics

Institute of Informatics

Outline

Computers have become an integral part of human lives and made it possible to solve many previously unattainable problems. Some tasks entrusted to them are very responsible. In situations when human life is at stake (e.g., when a computer is steering an autonomous car or conducting a medical surgery) there is no place for programming errors. There arises a *need for dependable software systems* which are provably error-free. This is one of the issues addressed by theoretical computer science, which developed various approaches to solving this problem.

One of them, *verification*, aims at proving that programs satisfy some given specification (describing their desired or undesired behaviours or properties). This is obtained through a rigorous analysis of programs, which often incorporates representing inner workings of real systems with simpler mathematical models. Another technique, known as *program synthesis*, assumes a different initial setup, where only specification needs to be provided, and the task is to automatically construct a program implementing it.

Both programs and their specifications can be represented through the means of *models of computation*. The trade-off between the expressive power of a model (which phenomena can they describe) and the simplicity of its analysis (ease of verifying its properties) cuts across the field of computation theory. A multitude of models exists, each resolving this difficult compromise in a different way.

On one side of the spectrum, there are finite automata—one of the simplest and most natural models. They are essentially finite transition systems labeled with letters from some alphabet—finite directed graphs with letter-labeled edges. Their computation, starting and ending in designated initial and final locations, corresponds to a string of letters visited in-between. Languages thereof form a class of regular languages—type three in the landmark Chomsky hierarchy. A well-studied[1] class of models, many of their properties being easily verifiable, finite automata are easy to work with, but not very expressive.

[1] Although here, too, there are few problems waiting to be solved, such as the question of size of the smallest automaton that distinguishes two given words of length n.

On the opposite side lie Turing machines. Like finite automata, they are based on finite transition systems, but also feature a kind of memory: an infinite tape with read and write access. The machine starts with an input word written on the tape, and performs computations until reaching an accepting state. If there is no finite accepting computation, the word is rejected. This more complicated model is more powerful, as it can recognise exactly all recursively enumerable languages, which constitute level zero of the Chomsky hierarchy. Unfortunately, it is much harder to verify properties of Turing machines. Most natural decision problems, like question whether given machine ever stops, are undecidable—there does not exist a reliable algorithmic way of determining the answer.

There is a great number of models of computation situated between these extreme cases, some of them addressing specific needs of particular domain and differing in the trade-offs made between expressive power and simplicity of analysis. Compared to Turing machines, such in-between devices may operate on other types of memory, be subject to syntactic or semantic constraints, and vary in design, e.g., working on infinite alphabets. Custom models often offer better fit to the use-case for which they were created, while avoiding increased computational complexity or undecidability of problems.

It should come as no surprise that researchers often try to determine circumstances in which it is possible to effectively replace a complex model with a simpler device. The quest of finding less involved devices that approximate the behavior of a complex system with sufficient precision is the central theme of this dissertation. We focus on several instances of that general scheme, considering the following questions:

Q1 Given a complex specification, does there exist a simpler device realising it?
(synthesis problem)

Q2 Can disjoint languages of complex models be distinguished (separated) in a simpler way?
(separability problem)

Q3 Can a complex device be equivalently described in a simpler way?
(membership problem)

Q4 Is a complex system bound to contain a simpler, more structured subsystem?
(pumping lemma)

Before we can state these problems more precisely and present our contributions in §2, we need to introduce the models of computation that are of interest in this dissertation.

1 Models of computation

Automata over infinite alphabets

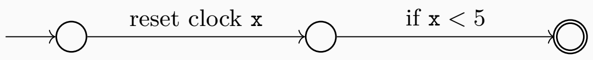
In the upcoming §§1.1 and 1.2, we present two models which both recognise words over infinite alphabets: $\Sigma \times \mathbb{Q}_{\geq 0}$ and $\Sigma \times \mathbb{A}$, where each input letter from a finite set Σ is paired with an element of an infinite set. In case of the first model—*timed automata*—the element is a nonnegative rational number interpreted as a *timestamp*, while for the next—*register automata*—$\mathbb{A}$ is an arbitrary countably infinite set of *atoms*. Words over these alphabets are called *timed words* and *data words*, and sets thereof are called timed and data languages, respectively. Both models are build upon standard finite automata. To deal with the extended alphabets, they are equipped with so-called *clocks* and *registers*, respectively.

1.1 Automata with clocks (NTA/DTA)

Timed automata are one of the most widespread models of real-time reactive systems. They are an extension of finite automata with rational-valued clocks[2] which can be reset and compared by inequality constraints (which may use integer constants).

Example 1.1. timed automaton distinguishing timestamps

Timed words $w_1 = (a, 0)(a, 4.5)$ and $w_2 = (a, 0)(a, 5.7)$ can be distinguished by a timed automaton. Consider the following automaton $\mathcal{A}$ with one clock $\mathtt{x}$ and three states.



It resets a clock $\mathtt{x}$ at the first symbol and then accepts if $\mathtt{x} < 5$ upon arrival of the next timed symbol. Note that the clocks measure time at the same pace as the timestamps of the input letters increase.

As with most models of computation, we distinguish between *nondeterministic* (NTA) and *deterministic* (DTA) timed automata.

Nondeterministic timed automata

The nonemptiness problem for NTA is decidable and, in fact, PSpace-complete, as shown by Alur and Dill in their landmark 1994 paper [2]. This paved the way for the automatic verification of timed systems, leading to mature tools such as UPPAAL [8], UPPAAL Tiga (timed games) [14], and PRISM (probabilistic timed automata) [57]. The reachability problem is still a very active research area these days [1, 37, 40, 41, 44, 46], as are expressive generalisations thereof, such as the binary reachability problem [24, 34, 39, 56]. As a testimony to the model's importance, the authors of [2] received the 2016 Church Award [16] for the invention of timed automata.

Deterministic timed automata

These are a strict subclass of NTA, where the successive configuration is uniquely determined by the current one and the timed input symbol. The class of DTA enjoys stronger properties than NTA, such as decidable universality/equivalence/inclusion problems, and closure under complementation [2]. Moreover, the more restrictive nature of DTA is needed for several applications of timed automata, such as test generation [65], fault diagnosis [10], learning [76, 81], winning conditions in timed games [4, 11, 49], and in a notion of recognisability of timed languages [60].

For these reasons, and for the more general quest of understanding the nature of the expressive power of nondeterminism in timed automata, many researchers have focused

[2]This generally accepted name is not very accurate, because the clocks of timed automata have more affinity to *stopwatches* than to clocks, which cannot be easily reset.

on defining *determinisable* classes of timed automata, such as strongly non-zeno NTA [5], event-clock NTA [3], and NTA with integer-resets [74]. These classes are not exhaustive, in the sense that there are NTA recognising deterministic timed languages not falling into any of them. In fact, the class of determinisable NTA is undecidable. Thus, it is of interest to be able to decide whether a timed language presented as an NTA is actually recognisable by a DTA.

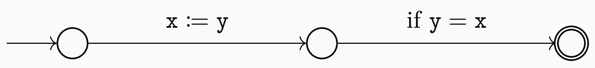
Another remarkable subclass of NTA is obtained by requiring the presence of just one clock (without epsilon transitions). The resulting class of NTA_1 is incomparable with DTA: For instance, NTA_1 are not closed under complement (unlike DTA), and there are very simple DTA languages that are not recognisable by any NTA_1. Nonetheless, NTA_1, like DTA, have decidable inclusion, equivalence, and universality problems [58, 66], although the complexity is non-primitive recursive [58, Corollary 4.2] (see also [67, Theorem 7.2] for an analogous lower bound for the satisfiability problem of metric temporal logic). Moreover, the nonemptiness problem for NTA_1 is NLogSpace-complete (vs. PSpace-complete for unrestricted NTA and DTA, already with two clocks [37]), and the binary reachability relation of NTA_1 can be computed as a formula of existential linear arithmetic of polynomial size, which is not the case in general [20].

1.2 Automata with registers (NRA/DRA)

The theory of register automata shares many similarities with that of timed automata. *Nondeterministic register automata* (NRA) have been introduced by Kaminski and Francez around the same time as timed automata [50]. They were defined as an extension of finite automata with finitely many registers which can store input values (now called data values, or *atoms*) and be compared with equality and disequality constraints.

Example 1.2. register automaton distinguishing two data words

Data words $w_1 = (a, \underline{0})(a, \underline{0})$ and $w_2 = (a, \underline{0})(a, \underline{1})$ can be distinguished by a register automaton $\mathcal{A}$ with one register $\mathtt{x}$ and three states:



Above, in the operations on edges, $\mathtt{y}$ denotes the currently read input value, while $\underline{0}, \underline{1}$ stand for some particular atoms.

Kaminski and Francez have shown, among other things, that nonemptiness is decidable [50, Theorem 1]. It was later realised that the problem is, in fact, PSpace-complete [33, Theorems 4.3 and Theorem 5.1]. The class of NRA recognisable languages is not closed under complementation [50, Proposition 5]; moreover, universality (and thus equivalence and inclusion) of NRA is undecidable [64, Theorem 5.1] (already for NRA with two registers [33, Theorem 5.4]).

Deterministic register automata

One way to regain decidability is to consider the deterministic register automata (DRA). This strict subclass of NRA is effectively closed under complement and thus has decidable inclusion problem[3] (entailing also universality and equivalence). DRA also provide the foundations of learning algorithms for data languages [63]. A recent result completing the theory of register automata has shown that a data language is DRA recognisable if, and only if, both this language and its complement are NRA recognisable [53].

As in the case of timed automata, it has been observed that restricting the number of registers results in algorithmic gains. Already in the seminal work of Kaminski and Francez, it has been proved that the inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ is decidable when $\mathcal{A}$ is an NRA and $\mathcal{B}$ is an NRA with one register [50, Appendix A], albeit the complexity is non-primitive recursive in this case [33, Theorem 5.2].

Other atoms

The notion of register automaton by Kaminski and Francez was subsequently generalised to allow input values (now commonly referred to as *atoms*) to originate from relational structure $\mathbb{A}$ given as a parameter. In this setting, the original variant of register automata is obtained by choosing $\mathbb{A}$ to be the equality atoms $\langle \mathbb{N}, = \rangle$. Countless other choices for atoms include total order atoms $\langle \mathbb{Q}, \leq \rangle$, or even timed atoms $\langle \mathbb{Q}, \leq, +1 \rangle$, which make register automata similar (but not identical) to timed ones.

The results shown in the thesis, for the sake of simplicity, are presented within the original regime of equality atoms; however, they can be generalised to other kinds of atoms, the method of this generalisation being described in the relevant chapters.

1.3 Automata with counters (VASS)

Vector addition systems [51] (VAS) are a widely accepted model of concurrency equivalent to Petri nets. Another equivalent model, called *vector addition systems with states* (VASS) [47], is an extension of VAS with a finite control graph. It can also be seen as a finite automaton over a unary alphabet with counters, on which the transitions can perform operations of increment or decrement (but no zero tests), with the proviso that counter values are initially set to zero and must remain non-negative along a run. The number of counters d defines the *dimension* of a VASS, model of dimension d being denoted as VASS_d.

One of the main theoretically relevant questions for VASS is the problem of reachability. It can be formulated as follows: given a VASS with two distinguished states p, q, does there exist a run from p to q, starting and ending with all counters set to zero? The problem's status has recently been clarified—several decades after the first decidability proof of Mayr [61] was published in 1981. However, the resulting ACKERMANN-completeness of the problem (dropping to PSPACE-completeness for VASS_2) essentially precludes practical applications. For this reason, when working with the difficult to analyse VASS model, one may be tempted to try to seek a simpler description of its properties. One such approach is the question of the regular separability of languages of labelled VASS, which is open already for VASS_2.

[3]In fact, even the inclusion problem $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ with A an NRA and B a DRA is decidable.

Described in the dissertation is a *pumping method* for runs of VASS_2, which we developed with the hope of eventually solving the regular separability problem in the two-dimensional case. Although we did not achieve our ultimate goal, our method—internally making use of a variety of geometric observations—seems to be a step in the right direction.

Pumping

Broadly speaking, *pumping* is a class of techniques exploiting repetitions of states in runs. It is a ubiquitous phenomenon which typically provides valuable tools in proving short run properties, characterizing set of runs or showing language inexpressibility results. It seems to be particularly relevant in case of VASS, as even the core of the seminal decision procedure for the reachability problem in VASS by Mayr and Kosaraju [55, 62] is fundamentally based on pumping. The decision procedure they propose essentially decomposes a VASS into a finite number of more structured VASS, each having the property that every path can be pumped up so that it induces a run.

Pumping techniques are used even more explicitly when dealing with subclasses of VASS of bounded dimension. The PSpace upper bound for the reachability problem in VASS_2 [9] relies on various un-pumping transformations of an original run, leading to a simple run of at most exponential length, in the form of a short path with adjoined short disjoint cycles. A smart combinatorial manipulation of these simple runs was also used to obtain a stronger upper bound (NL) in case when the transition effects are represented in unary [36]. Un-pumping is also used in [15] to provide a quadratic bound on the length of the shortest run for VASS_1, also known as one counter automata without zero tests, and for unrestricted one counter automata. See also [6, 59] for pumping techniques in one counter automata.

2 Contributions grouped by problems

Here, we summarise the main results found in the dissertation. Each of following §§2.1 to 2.4 first sketches the problem and the state of the art on it, and then presents the newly proven theorems.

2.1 Synthesis problem and synthesis games

The principal result of the dissertation concerns the previously listed question Q1

Q1 Given a complex specification, does there exist a simpler device realising it? (synthesis problem)

where the (complex) specification is specified in terms of NRA/NTA, and the goal is to provide a deterministic device realising it. Before we discuss it, we need to introduce the notions of ω-regular languages and synthesis games.

ω-regular languages

For finite automata over alphabet Σ, there are many ways of adapting them to recognise *infinite words* (from Σ^ω) instead of finite ones (from Σ^*). Some of these approaches only modify semantics, while others introduce subtle alterations of the syntax, too. One may consider DRA or NRA with updated acceptance conditions, i.a., Büchi, Muller, or parity condition. Most[4] of these variants have the same set of languages they can express—this way, the class of *ω-regular* languages arises.

Synthesis games

One of the areas in which ω-regular languages find their use are *synthesis games*. These infinite-duration games are played by two players, Alice and Bob, who both are assigned finite alphabets of actions, A and B. An instance of a game is given as its winning set—an ω-regular language $W \subseteq (A \times B)^\omega$. Each turn consists of a single action of Alice followed by one of Bob. The completed infinite play is a word $w \in (A \times B)^\omega$, and Alice wins if $w \in W$. A *finite memory strategy* in such a game is a deterministic automaton with additional output[5] letter written on every transition. It reads the opponent's actions one by one and responds to them synchronously, with current player's actions.

Synthesis problem

In their famous result [12], Büchi and Landweber have shown that one can

- decide the winner of such games,
- compute (synthesize) a finite-memory strategy for the winner.

Intuitively, if a player can win using arbitrary (possibly infinite) memory, they can also win using a finite memory strategy, which can be effectively computed.

We generalise these games to the setting of automata over infinite alphabets, defining *timed synthesis games* and *register synthesis games*. Since the two variants share many similarities, we present them jointly here. The differences are highlighted with square brackets [X,Y], where X and Y concern timed and register variant, respectively.

Timed and register synthesis games

There are two players, called Alice and Bob, who take turns in a strictly alternating fashion. At the i-th round, Alice selects a letter a_i from a finite alphabet and [a nonnegative timestamp/an atom] χ_i from $[\mathbb{Q}_{\geq 0}/\mathbb{A}]$. Bob replies with a letter b_i from a finite alphabet. This process continues infinitely. At the doomsday, the two players will have built an infinite play $\pi = (a_1, b_1, \chi_1)(a_2, b_2, \chi_2)\cdots$, and Alice wins if, and only if, π belongs to her winning set W, which is specified as the language of a [NTA/NRA] (possibly with ε-transitions).

Analogously to original synthesis games, finite-memory strategies in above game correspond to [timed/register] automata with output.

[4] All classes but DRA equipped with Büchi condition, which can only express a strict subset of ω-regular languages.

[5] Other names by which such automaton with output is known include 'letter-to-letter transducer', and 'Mealy machine'.

Timed and register synthesis problems

The *[timed/register] synthesis problem* is to compute, if it exists, a winning controller for Bob —[DTA/DRA] with output—which ensures that every play π conformant to that controller is winning for Bob. We also consider two more restricted variants of it. For a fixed number of [clocks/registers] $k \in \mathbb{N}$, the *[k-clock timed/k-register] synthesis problem* asks—as before—about a controller for Bob, but this time using at most k [clocks/registers]. Finally, exclusively for the timed variant, we define a *k-clock m-constrained timed synthesis problem*, which is analogous to k-clock timed synthesis, but also requires the controller's transitions only to feature constraints with constant bounded by m in absolute value. With all problems, we naturally associate decision problems ('does a winning controller exist?').

We show the computability of the restricted variants of synthesis problem, which are analogues to the Büchi-Landweber result.

Theorem 2.1. synthesis of DRA_k

For every fixed $k \in \mathbb{N}$, the k-register synthesis problem is computable.

Theorem 2.2. synthesis of $\mathrm{DTA}_{k,m}$

For every fixed $k, m \in \mathbb{N}$, the k-clock m-constrained timed synthesis problem is computable.

Theorem 2.3. synthesis of DTA_k

For every fixed $k \in \mathbb{N}$, the k-clock timed synthesis problem is computable.

The main part of our proof is the reduction of these games to finite-state games with an ω-regular winning conditions. This is done via introduction of a special protocol using orbits/regions. Then, they can be solved using the seminal result of Büchi and Landweber [12].

It is important to note that the k-clock timed synthesis problem requires the synthesis of the maximal constant m, which is a technical novelty not shared with the current literature on timed games. This result is obtained, intuitively, through a design of protocol in which Bob demands from Alice to be informed when clocks elapse one time unit. We require the number of such consecutive requests to be finite, yielding a bound on m (when such a value exists).

Additionally, our construction of synthesis games in itself is noteworthy, as we thoughtfully designed them to provide high expressiveness, while avoiding unnecessary complication:

- It was our informed choice to only allow Alice to use an infinite alphabet of actions. This way, finite-memory strategies of Bob in our games are equivalent to timed/register automata with output symbols (a.k.a. letter-to-letter transducers).
- Additionally, it is important that we specify the winning condition for Alice. In fact, the same problem—with the set of winning plays for Bob specified by a nondeterministic timed language—becomes uncomputable (cf. [35] for a similar observation).

When compared with our setup, variant of register synthesis problem considered by Khalimov et al. [52] (settled independently) may ostensibly seem more expressive. In their framework—not specified in terms of game—the transducer (automaton with output) can also output atoms. In our terminology, this situation corresponds to a game, in which both Alice and Bob can play atoms, but Bob is limited only to play these which Alice has used so far. However, already in our original, simpler setting, one can obtain undecidability of register synthesis problem

Theorem 2.4. synthesis of DRA und.

The register synthesis decision problem is undecidable, and this holds already when Alice's winning condition is an NRA_2 language.

and of timed synthesis, too:

Theorem 2.5. synthesis of DTA und.

The timed synthesis decision problem is undecidable, and this holds already when Alice's winning condition is an NTA_2 language.

Moreover, as in [52] the transducer is limited to use only atoms stored in registers, in context of [k-clock/k-register] synthesis problem, his set of choices of symbols is in fact always bounded by k. This makes it amenable to simulation with a finite alphabet. It follows that our k-register synthesis setup is equally expressive as one of [52], while additionally allowing other atom domains.

2.2 Deterministic separability problem

Separability is a classical problem in theoretical computer science and mathematics. For sets S, L, M, we say that S *separates* L from M if $L \subseteq S$ and $S \cap M = \varnothing$.

Intuitively, a separator S provides a certificate of disjointness, yielding information on the structure of L, M up to some level of granularity.

There are many elegant results in computer science and mathematics showing that separators with certain properties always exist, such as Lusin's separation theorem in topology (two disjoint analytic sets are separable by a Borel set), Craig's interpolation theorem in logic (two contradictory first-order formulas can be separated by one containing only symbols in the shared vocabulary), in model theory (two disjoint projective classes of models are separable by an elementary class), in formal languages (two disjoint Büchi languages of infinite trees are separable by a weak language, generalising Rabin's theorem [72]), in computability (two disjoint co-recursively enumerable sets are separable by a recursive set), in the analysis of infinite-state systems (two disjoint languages recognisable by well-structured transition systems are regular separable [27]), etc.

When separability is not trivial, one may ask whether existence of separator is decidable. Let $\mathcal{C}$ and $\mathcal{S}$ be two classes of sets. The $\mathcal{S}$-*separability* problem for $\mathcal{C}$ amounts to decide whether, for every input sets $L, M \in \mathcal{C}$ there is a set $S \in \mathcal{S}$ separating L, M. Many results of this kind exist when $\mathcal{C}$ is the class of regular languages of finite words over finite alphabets, and $\mathcal{S}$ ranges over piecewise-testable languages [29, 68] (later generalised to context-free languages [30] and finite trees [43]), locally and locally threshold testable languages [69], first-order logic definable languages [71] (generalised to some fixed levels of the first-order hierarchy [70]). For classes of languages $\mathcal{C}$ beyond the regular ones, decidability results are more rare. For example, regular separability of context-free languages is undecidable [48, 54, 75]. Nonetheless, there are positive decidability results for separability problems on several infinite-state models, such as Petri nets [19], Parikh automata [17], one-counter automata [25], higher-order and collapsible pushdown automata [23, 45], and others.

In this dissertation, we study the separability problem for languages of timed and register automata. In particular, we are interested in the *deterministic separability problem*, in which we seek a simple—deterministic—separator for languages of more complex—nondeterministic—devices. This matches the previously stated question Q2:

Q2 Can disjoint languages of complex models be distinguished (separated) in a simpler way? (separability problem)

More precisely, the problem asks, given two nondeterministic automata $\mathcal{A}$ and $\mathcal{B}$ with disjoint languages, be it timed or register ones, whether there exists a deterministic automaton $\mathcal{S}$ such that $\mathcal{L}(\mathcal{S})$ separates $\mathcal{L}(\mathcal{A})$ from $\mathcal{L}(\mathcal{B})$. Various ways of quantifying the models' parameters, like the desired number of clocks/registers of the separator, give rise to the following variants of the separability question:

- The *k-clock m-constrained deterministic separability problem* for timed automata asks whether $\mathcal{A}, \mathcal{B} \in \mathsf{NTA}$ can be separated by a deterministic automaton $\mathcal{S} \in \mathsf{DTA}$ with k clocks measuring up to m units of time.
- The *k-clock deterministic separability problem* for timed automata is defined similarly, but with the restraint on m is lifted. This way the resulting separator $\mathcal{S}$ can measure the time up to arbitrary fixed number of time units.
- Finally, the *k-register deterministic separability* for register automata asks whether $\mathcal{A}, \mathcal{B} \in \mathsf{NRA}$ can be separated by a deterministic automaton $\mathcal{S} \in \mathsf{DRA}$ with k registers.

Intuitively, deterministic separability problem strengthens disjointness by additionally requiring a simple (deterministic) reason of disjointness of more complex (nondeterministic) models.

We can also see $\mathcal{A}$ as recognising a set of good behaviours which we want to preserve and $\mathcal{B}$ recognising a set of bad behaviours which we want to exclude. From this point of view, a deterministic separator, when it exists, provides a kind of compromise between these two conflicting requirements.

Concerning separability, our main contribution is decidability of k-clock m-constrained and k-clock deterministic separability, obtained by means of synthesis games discussed in §2.1. We provide analogous results for both timed and register automata:

Theorem 2.6. separability by DRA_k

For $k \in \mathbb{N}$, the k-register deterministic separability problem for NRA is decidable.

Theorem 2.7. separability by $\mathsf{DTA}_{k,m}$

For $k, m \in \mathbb{N}$, the k-clock m-constrained separability problem for NTA is decidable.

Theorem 2.8. separability by DTA_k

For $k \in \mathbb{N}$, the k-clock deterministic separability problem for NTA are decidable.

To the best of our knowledge, separability problems for timed and register automata have not been investigated before.

Decidability of timed deterministic separability should be contrasted with undecidability of the corresponding deterministic membership problem—the question whether a nondeterministic timed language is deterministic is undecidable [38, 78] (also cf. §2.3). This is a rare circumstance, which is shared with languages recognised by one-counter nets [25], and conjectured to be the case for the full class of Petri net languages[6].

Similar situation holds also for register automata, as we outline in the next section §2.3.

2.3 Deterministic membership problem

Here, we consider an instance of the question Q3:

Q3 Can a complex device be equivalently described in a simpler way? (membership problem)

where 'complex' and 'simpler' mean 'nondeterministic' and 'deterministic', respectively. We provide analogous results for both timed and register automata.

Timed automata

The DTA *membership problem* asks, given an NTA, whether there exists a DTA recognising the same language. There are two natural variants of this problem, which are obtained by restricting the resources available to the sought DTA. Let $k \in \mathbb{N}$ be a bound on the number of clocks, and let $m \in \mathbb{N}$ be a bound on the maximal absolute value of numerical constants.

- The DTA_k is a variant of the problem above where the DTA is required to have at most k clocks.
- The $\mathsf{DTA}_{k,m}$ *membership problem* asks for the automaton with at most k clocks and constants bounded by m in absolute value.

[6] All these classes of languages have a decidable disjointness problem, however regular separability is not always decidable in this case [77].

Notice that we do not bound the number of control locations of the DTA, which makes the problem non-trivial. (Indeed, there are finitely many DTA with a bounded number of clocks, control locations, and maximal constant.)

Since untimed regular languages are deterministic, the DTA_k membership problem can be seen as a quantitative generalisation of the regularity problem. For instance, the DTA_0 membership problem is precisely the regularity problem since a timed automaton with no clocks is essentially the same as a finite automaton. We remark that the regularity problem is usually undecidable for nondeterministic models of computation generalising finite automata, e.g., context-free grammars/pushdown automata [73, Theorem 6.6.6], labelled Petri nets under reachability semantics [80], Parikh automata [13], etc. One way to obtain decidability is to either restrict the input model to be deterministic (e.g., [7, 79, 80]), or to consider more refined notions of equivalence, such as bisimulation (e.g., [42]).

This negative situation is generally confirmed for timed automata. For every number of clocks $k \in \mathbb{N}$ and maximal constant m, the DTA, DTA_k, and $\mathsf{DTA}_{k,m}$ membership problems are known to be undecidable when the input NTA has ≥ 2 clocks, and for 1-clock NTA with epsilon transitions [38, 78]. To the best of our knowledge, the deterministic membership problem was not studied before when the input automaton is NTA_1 without epsilon transitions.

Register automata

The situation with register automata is similar to that of timed automata, only simpler.

- The DRA_k *membership problem* asks, given an NRA, whether there exists a DRA with k registers recognising the same language.
- The DRA *membership problem* is the same problem, but with no apriori bound on the number of registers of the deterministic acceptor.

Deterministic membership problems for register automata do not seem to have been considered before in the literature.

Contributions

We complete the study of the decidability border for the deterministic membership problem initiated for timed automata in [38, 78], and we extend these results to register automata.

Upper bounds

Our main results on the deterministic membership problem for timed automata are

Theorem 2.9. DTA_k membership

For every fixed $k \in \mathbb{N}$, the DTA_k membership problem is decidable for NTA_1 languages.

Theorem 2.10. $\mathsf{DTA}_{k,m}$ membership

For every fixed $k, m \in \mathbb{N}$, the $\mathsf{DTA}_{k,m}$ membership problem is decidable for NTA_1 languages.

Our decidability result contrasts starkly with the abundance of undecidability results for the regularity problem. We establish decidability by showing that if an $\mathsf{NTA}_{1,m}$ recognises a DTA_k language, then, in fact, it recognises a $\mathsf{DTA}_{k,m}$ language and, moreover, there is a computable bound on the number of control locations of the deterministic acceptor. This provides a decision procedure since there are finitely many different DTA once the number of clocks, the maximal constant, and the number of control locations are fixed.

In our technical analysis, we find it convenient to introduce the so-called *always resetting* subclass of NTA_k. These automata are required to reset at least one clock at every transition and are thus of expressive power intermediate between NTA_{k-1} and NTA_k. Always resetting NTA_2 are strictly more expressive than NTA_1: For instance, the language of timed words of the form $(a, \tau_0)(a, \tau_1)(a, \tau_2)$ such that $\tau_2 - \tau_0 > 2$ and $\tau_2 - \tau_1 < 1$ can be recognised by an always resetting NTA_2 but by no NTA_1. Despite their increased expressive power, always resetting NTA_2 still have a decidable universality problem (the well-quasi order approach of [66] goes through), which is not the case for NTA_2. Thanks to this restricted form, we are able to prove a lemma providing an elegant characterisation of those NTA_1 languages which are recognised by an always resetting DTA_k.

We prove a result analogous to theorem 2.9 in the setting of register automata.

Theorem 2.11. DRA_k membership

For every fixed $k \in \mathbb{N}$, the DRA_k membership problem is decidable for NRA_1 languages.

Thanks to the effective elimination of ε-transition rules from NRA_1, the decidability result above also holds for data languages presented as NRA_1 with ε-transition rules.

Lower bounds

We complement the decidability results above by showing that the deterministic membership problem becomes undecidable if we do not restrict the number of clocks/registers of the deterministic acceptor.

Theorem 2.12. DTA and $\mathsf{DTA}_{\bullet,m}$ membership und.

The DTA and $\mathsf{DTA}_{\bullet,m}$ $(m > 0)$ membership problems are undecidable for NTA_1 without epsilon transitions.

Theorem 2.12 is shown by improving on an analogous result from [38, Theorem 1] for NTA_2. We obtain a similar undecidability result in the setting of register automata:

Theorem 2.13. DRA membership und.

The DRA membership problem is undecidable for NRA_1.

The following lower bounds further refine the analysis from [38] in the case of a fixed number of clocks of a deterministic acceptor.

Theorem 2.14. undecidability and hardness for DTA_k membership

For every fixed $k, m \in \mathbb{N}$, the DTA_k and $\mathsf{DTA}_{k,m}$ membership problems are:

1. undecidable for NTA_2,
2. undecidable for $\mathsf{NTA}_1^{\varepsilon}$ (NTA with epsilon transitions),
3. HyperAckermann-hard for NTA_1.

A similar landscape holds for register automata, where the deterministic membership problem for a fixed number of registers of the deterministic acceptor remains undecidable when given in input either an NRA_2 or an NRA_1 with guessing[7]. In the decidable case of an NRA_1 input, the problem is nonetheless not primitive recursive.

Theorem 2.15. undecidability and hardness for DRA_k membership

Fix a $k \geq 0$. The DRA_k membership problem is:

1. undecidable for NRA_2,
2. undecidable for NRA_1^{g} (NRA_1 with guessing), and
3. not primitive recursive (Ackermann-hard) for NRA_1.

2.4 Pumping technique for VASS of dimension two

The pumping techniques mentioned in §1.3 are mostly oriented towards reachable sets, and henceforth may ignore certain runs as long as the reachable set is preserved. As consequence, they are not very helpful in solving decision problems formulated in terms of the language accepted by a VASS, like the regular separability problem (cf. [18, 28, 32]).

In the dissertation, we restrict ourselves exclusively to the VASS of dimension two, our primary objective being to design means of pumping applicable to *every* run of a VASS_2. Therefore, as our main technical contribution related to VASS_2, we perform a meticulous classification of runs, in form of a dichotomy:

For every run π of a VASS_2 starting and ending with both counters set to zero,

- either π is *thin*, by which we mean that the counter values along the run stay within belts, whose direction and width are all bounded polynomially in the number n of states and the largest absolute value M in vectors of the VASS_2;
- or π is *thick*, by which we mean that a number of cycles is enabled along the run, and the effect vectors of these cycles, roughly speaking, span the whole plane. Additionally, the lengths of cycles and the initial and final factors of π are all bounded polynomially in M and exponentially in n.

[7]Register automata with guessing are a more expressive family of automata where a register can be updated with a data value not necessarily coming from the input word, i.e., it can be *guessed*.

The precise formulation of the dichotomy, omitting the definitions of thin and thick runs, is as follows. It speaks of $(0,0)$-runs, i.e., runs starting and ending with both counters set to 0.

Theorem 2.16. thin/thick dichotomy

There is a polynomial p such that every $(0,0)$-run in a VASS$_2$ V is either $p(nM)^n$-thin or $p(nM)^n$-thick.

The dichotomy immediately entails a pumping theorem for VASS$_2$.

Theorem 2.17. pumping runs of VASS$_2$

There is a polynomial p such that every $(0,0)$-run τ in a VASS$_2$ of length greater that $p(nM)^n$ factors into $\tau = \tau_0\, \tau_1 \, \ldots \, \tau_k$ $(k \geq 1)$, so that for some non-empty cycles $\alpha_1, \ldots, \alpha_k$ of length at most $p(nM)^n$, the path $\tau_0\, \alpha_1^i\, \tau_1\, \alpha_2^i\, \ldots,\, \alpha_k^i\, \tau_k$ is a $(0,0)$-run for every $i \in \mathbb{N}$. Furthermore, the lengths of τ_0 and τ_k are also bounded by $p(nM)^n$.

Intuitively, adapted pumping scheme of VASS$_1$ is used in case of thin runs, whereas for thick runs we we utilize the collected cycles to construct correct 'inflated' runs.

As a more subtle application of the dichotomy, we derive an alternative proof of the exponential run property (shown originally in [9]), which immediately implies PSPACE-membership of the reachability problem.

Theorem 2.18. exponential run property

There is a polynomial p such that for every $(0,0)$-run τ in a VASS$_2$, there is a $(0,0)$-run of length bounded by $p(nM)^n$ with the same source and target as τ.

3 Source materials

Many of the results presented in the thesis stem from the author's close collaboration with several people. This has borne fruit in scientific articles, already published or awaiting publication. Other results of this collaboration, previously unpublished, are presented for the first time. The list below summarises the relationship between the source materials and theorems in this thesis.

Sources

1. *Timed Games and Deterministic Separability* [22] (ICALP 2020)
   Authors: Lorenzo Clemente, Sławomir Lasota and P.
   - theorems 2.2, 2.3 and 2.5 (timed synthesis)
   - theorems 2.7 and 2.8 (DTA separability)
2. *Determinisability of One-Clock Timed Automata* [21] (CONCUR 2020)
   Authors: Lorenzo Clemente, Sławomir Lasota and P.
   - theorems 2.9, 2.10 and 2.14 (DTA membership)

3. *Determinisability of register and timed automata* (submitted to LMCS)
   Authors: Lorenzo Clemente, Sławomir Lasota and P.
   - theorems 2.9 to 2.11, 2.14 and 2.15 (DTA and DRA membership)
4. *New Pumping Technique for 2-Dimensional VASS* [26] (MFCS 2019)
   Authors: Wojciech Czerwinski, Slawomir Lasota, Christof Löding and P.
   - theorems 2.17 and 2.18 (pumping technique for VASS_2)
5. Previously unpublished results
   Authors: Lorenzo Clemente, Sławomir Lasota and P.
   - theorems 2.1 and 2.4 (register synthesis)
   - theorem 2.6 (DRA separability)

4 Conclusion

In this dissertation, we present an extensive set of results that fall within the general quest for simplifying more complex systems. We propose a relatively simple framework of register and timed synthesis games—a generalisation of the Church's synthesis problem to infinite-state systems. The computability results we obtain for controllers with a limited number of registers or clocks allow us to easily settle the corresponding variants of the deterministic separability problem. Unusually, this solution is common to both models. Unusually—as, often, in spite of their far-reaching similarities, timed and register automata generate inconsistencies which make it impossible to apply uniform proof techniques. Our set of results is completed by the definite solution to the deterministic membership problem (resolving one-register/clock corner cases), and by the introduction of a run pumping technique based on geometric observations, with potential application in proving further results for VASS.

Open problems

While working on the results of this dissertation, we encountered several problems which have not yet been resolved. These include:

- decidability status of DRA- and DTA-separability problem
  Unlike the corresponding synthesis result, separability resists an undecidability proof because of its simpler, more constraining structure.
- decidability status of regular separability for VASS_2
  So far, the only positive results obtained in this area assume one of the separated languages belongs to VASS_1 (cf. [31]). This is the simplest as yet unsolved case of the general regular separability problem of languages of VASS_k.

We leave these questions open as a set-up for further research work.

Bibliography

[1] S. Akshay, Paul Gastin, and Shankara Narayanan Krishna. Analyzing Timed Systems Using Tree Automata. *Logical Methods in Computer Science*, Volume 14, Issue 2, May 2018. URL: `https://lmcs.episciences.org/4489`, `doi:10.23638/LMCS-14(2:8)2018`.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.

[3] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theor. Comput. Sci.*, 211:253–273, January 1999.

[4] Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Proc. of HSCC'99*, HSCC '99, pages 19–30, London, UK, UK, 1999. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=646879.710314`.

[5] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. of SSSC'98*, volume 31 of *5th IFAC Conference on System Structure and Control*, pages 447–452, 1998. URL: `http://www.sciencedirect.com/science/article/pii/S1474667017420325`, `doi:https://doi.org/10.1016/S1474-6670(17)42032-5`.

[6] Mohamed Faouzi Atig, Dmitry Chistikov, Piotr Hofman, K. Narayan Kumar, Prakash Saivasan, and Georg Zetzsche. The complexity of regular abstractions of one-counter languages. In *Proc. LICS'16*, pages 207–216, 2016.

[7] Vince Bárány, Christof Löding, and Olivier Serre. Regularity problems for visibly pushdown languages. In *Proc. of STACS'06*, STACS'06, pages 420–431, Berlin, Heidelberg, 2006. Springer-Verlag. URL: `http://dx.doi.org/10.1007/11672142_34`, `doi:10.1007/11672142_34`.

[8] Gerd Behrmann, Alexandre David, Kim G. Larsen, John Hakansson, Paul Petterson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, QEST '06, pages 125–126, Washington, DC, USA, 2006. IEEE Computer Society. `doi:10.1109/QEST.2006.59`.

[9] Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proc. LICS'15*, pages 32–43, 2015.

[10] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In *Proc. of FOSSACS'05*, FOSSACS'05, pages 219–233, Berlin, Heidelberg, 2005. Springer-Verlag. `doi:10.1007/978-3-540-31982-5_14`.

[11] Thomas Brihaye, Thomas A. Henzinger, Vinayak S. Prabhu, and Jean-François Raskin. Minimum-time reachability in timed games. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *Proc. of ICALP'07*, pages 825–837, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[12] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL: `http://www.jstor.org/stable/1994916`.

[13] Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the expressiveness of Parikh automata and related models. In Rudolf Freund, Markus Holzer, Carlo Mereghetti, Friedrich Otto, and Beatrice Palano, editors, *Proc. of NCMA'11*, volume 282 of *books@ocg.at*, pages 103–119. Austrian Computer Society, 2011.

[14] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *Proc. of CONCUR'05*, pages 66–80, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[15] Dmitry Chistikov, Wojciech Czerwinski, Piotr Hofman, Michal Pilipczuk, and Michael Wehar. Shortest paths in one-counter systems. In *Proc. of FOSSACS'16*, pages 462–478, 2016.

[16] The 2016 alonzo church award for outstanding contributions to logic and computation. `https://siglog.org/the-2016-alonzo-church-award-for-outstanding-contributions-to-logic-and-computation/`, 2016.

[17] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular separability of parikh automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proc. of ICALP'17*, volume 80, pages 117:1–117:13, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7497`, `doi:10.4230/LIPIcs.ICALP.2017.117`.

[18] Lorenzo Clemente, Wojciech Czerwiński, Slawomir Lasota, and Charles Paperman. Regular Separability of Parikh Automata. In *The 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017))*, Varsovie, Poland, 2017. URL: `https://hal.archives-ouvertes.fr/hal-01587616`, `doi:10.4230/LIPIcs.ICALP.2017.117`.

[19] Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In *Proc. of STACS'17*, volume 66 of *LIPICs*, pages 24:1–24:14, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7009`, `doi:10.4230/LIPIcs.STACS.2017.24`.

[20] Lorenzo Clemente, Piotr Hofman, and Patrick Totzke. Timed Basic Parallel Processes. In Wan Fokkink and Rob van Glabbeek, editors, *Proc. of CONCUR'19*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*,

pages 15:1–15:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10917`, `doi:10.4230/LIPIcs.CONCUR.2019.15`.

[21] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Determinisability of One-Clock Timed Automata. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory (CONCUR 2020)*, volume 171 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/12854`, `doi:10.4230/LIPIcs.CONCUR.2020.42`.

[22] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Timed Games and Deterministic Separability. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 121:1–121:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/12528`, `doi:10.4230/LIPIcs.ICALP.2020.121`.

[23] Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *Proc. of LICS'16*, 2016. URL: `http://doi.acm.org/10.1145/2933575.2934527`, `doi:10.1145/2933575.2934527`.

[24] Hubert Comon and Yan Jurski. Timed automata and the theory of real numbers. In *Proc. of CONCUR'99*, CONCUR '99, pages 242–257, London, UK, UK, 1999. Springer-Verlag.

[25] Wojciech Czerwiński and Sławomir Lasota. Regular Separability of One Counter Automata. *Logical Methods in Computer Science*, Volume 15, Issue 2, June 2019. URL: `https://lmcs.episciences.org/5563`.

[26] Wojciech Czerwinski, Slawomir Lasota, Christof Löding, and Radoslaw Piórkowski. New Pumping Technique for 2-Dimensional VASS. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11006`, `doi:10.4230/LIPIcs.MFCS.2019.62`.

[27] Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9573`, `doi:10.4230/LIPIcs.CONCUR.2018.35`.

[28] Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9573`, `doi:10.4230/LIPIcs.CONCUR.2018.35`.

[29] Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proc. of ICALP'14*, ICALP'13, pages 150–161, Berlin, Heidelberg, 2013. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-39212-2_16`, `doi:10.1007/978-3-642-39212-2_16`.

[30] Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proc. of FCT'15*, 2015. URL: `http://dx.doi.org/10.1007/978-3-319-22177-9_14`, `doi:10.1007/978-3-319-22177-9_14`.

[31] Wojciech Czerwiński and Georg Zetzsche. An approach to regular separability in vector addition systems. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 341–354, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3373718.3394776`.

[32] Wojciech Czerwiński and Slawomir Lasota. Regular separability of one counter automata. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017. `doi:10.1109/LICS.2017.8005079`.

[33] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. `doi:10.1145/1507244.1507246`.

[34] C. Dima. Computing reachability relations in timed automata. In *Proc. of LICS'02*, pages 177–186, 2002.

[35] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In Helmut Alt and Afonso Ferreira, editors, *Proc. of STACS'02*, pages 571–582, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[36] Matthias Englert, Ranko Lazic, and Patrick Totzke. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *Proc. of LICS '16*, pages 477–484, 2016.

[37] John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation*, 243:26–36, 2015. URL: `http://www.sciencedirect.com/science/article/pii/S0890540114001564`, `doi:http://dx.doi.org/10.1016/j.ic.2014.12.004`.

[38] Olivier Finkel. Undecidable problems about timed automata. In *Proc. of FORMATS'06*, FORMATS'06, pages 187–199, Berlin, Heidelberg, 2006. Springer-Verlag. URL: `http://dx.doi.org/10.1007/11867340_14`, `doi:10.1007/11867340_14`.

[39] Martin Fränzle, Karin Quaas, Mahsa Shirmohammadi, and James Worrell. Effective definability of the reachability relation in timed automata. *Information Processing Letters*, 153:105871, 2020. URL: `http://www.sciencedirect.com/science/article/pii/S0020019019301541`, `doi:https://doi.org/10.1016/j.ipl.2019.105871`.

[40] Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in Timed Automata with Diagonal Constraints. In Sven Schewe and Lijun Zhang, editors, *Proc. of CONCUR'18*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9566`, `doi:10.4230/LIPIcs.CONCUR.2018.28`.

[41] Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Fast algorithms for handling diagonal constraints in timed automata. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 41–59, Cham, 2019. Springer International Publishing.

[42] Stefan Göller and Paweł Parys. Bisimulation finiteness of pushdown systems is elementary. In *Proc. of LICS'20*, pages 521–534, 2020.

[43] Jean Goubault-Larrecq and Sylvain Schmitz. Deciding Piecewise Testable Separability for Regular Tree Languages. In *Proc. of ICALP'16*, volume 55 of *LIPIcs*, pages 97:1–97:15, 2016. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6232`, `doi:10.4230/LIPIcs.ICALP.2016.97`.

[44] R. Govind, Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Revisiting Local Time Semantics for Networks of Timed Automata. In Wan Fokkink and Rob van Glabbeek, editors, *Proc. of CONCUR 2019*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10918`, `doi:10.4230/LIPIcs.CONCUR.2019.16`.

[45] Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *Proc. of POPL'16*, POPL 2016, pages 151–163, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2837614.2837627`, `doi:10.1145/2837614.2837627`.

[46] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. *Information and Computation*, 251:67–90, 2016. URL: `http://www.sciencedirect.com/science/article/pii/S0890540116300438`, `doi:https://doi.org/10.1016/j.ic.2016.07.004`.

[47] John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. `doi:10.1016/0304-3975(79)90041-0`.

[48] H. B. Hunt, III. On the decidability of grammar problems. *J. ACM*, 29(2):429–447, April 1982. URL: `http://doi.acm.org/10.1145/322307.322317`, `doi:10.1145/322307.322317`.

[49] Marcin Jurdziński and Ashutosh Trivedi. Reachability-time games on timed automata. In *Proc. of ICALP'07*, pages 838–849, Berlin, Heidelberg, 2007. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=2394539.2394637`.

[50] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[51] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.

[52] Ayrat Khalimov, Benedikt Maderbacher, and Roderick Bloem. Bounded synthesis of register transducers. In Shuvendu Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, pages 494–510, 2018. 16th International Symposium on Automated Technology for Verification and Analysis, ATVA 2018 ; Conference date: 07-10-2018 Through 10-10-2018. `doi:10.1007/978-3-030-01090-4_29`.

[53] Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk. Nondeterministic and co-nondeterministic implies deterministic, for data languages. *Proc. of FOSSACS'21*, 12650:365–384, 03 2021. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7984108/`, `doi:10.1007/978-3-030-71995-1{\_}19`.

[54] Eryk Kopczynski. Invisible pushdown languages. In *Proc. of LICS'16*, pages 867–872, 2016. URL: `http://doi.acm.org/10.1145/2933575.2933579`, `doi:10.1145/2933575.2933579`.

[55] S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *STOC'82*, pages 267–281, 1982.

[56] Pavel Krčál and Radek Pelánek. On sampled semantics of timed systems. In Sundar Sarukkai and Sandeep Sen, editors, *Proc. of FSTTCS'05*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005. URL: `http://dx.doi.org/10.1007/11590156_25`.

[57] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[58] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Logic*, 9(2):10:1–10:27, 2008. URL: `http://doi.acm.org/10.1145/1342991.1342994`, `doi:10.1145/1342991.1342994`.

[59] Michel Latteux. Langages à un compteur. *J. Comput. Syst. Sci.*, 26(1):14–33, 1983.

[60] Oded Maler and Amir Pnueli. On recognizable timed languages. In Igor Walukiewicz, editor, *Proc. of FOSSACS'04*, volume 2987 of *LNCS*, pages 348–362. Springer Berlin Heidelberg, 2004. URL: `http://dx.doi.org/10.1007/978-3-540-24727-2_25`, `doi:10.1007/978-3-540-24727-2_25`.

[61] Ernst W. Mayr. An algorithm for the general petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, page 238–246, New York, NY, USA, 1981. Association for Computing Machinery. `doi:10.1145/800076.802477`.

[62] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *STOC'81*, pages 238–246, 1981.

[63] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. Learning nominal automata. In *Proc. of POPL'17*, POPL 2017, pages 613–625, New York, NY, USA, 2017. ACM. URL: `http://doi.acm.org/10.1145/3009837.3009879`, `doi:10.1145/3009837.3009879`.

[64] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, July 2004. `doi:10.1145/1013560.1013562`.

[65] Brian Nielsen and Arne Skou. Automated test generation from timed automata. *International Journal on Software Tools for Technology Transfer*, 5(1):59–77, Nov 2003. `doi:10.1007/s10009-002-0094-1`.

[66] Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 54–63, 2004. `doi:10.1109/LICS.2004.1319600`.

[67] Joël Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, Volume 3, Issue 1, February 2007. URL: `https://lmcs.episciences.org/2230`, `doi:10.2168/LMCS-3(1:8)2007`.

[68] Thomas Place, Lorijn Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In Krishnendu Chatterjee and Jirí Sgall, editors, *Proc. of MFCS'13*, pages 729–740. Springer, 2013. URL: `http://dx.doi.org/10.1007/978-3-642-40313-2_64`, `doi:10.1007/978-3-642-40313-2_64`.

[69] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. *LMCS*, 10(3), 2014. URL: `http://dx.doi.org/10.2168/LMCS-10(3:24)2014`, `doi:10.2168/LMCS-10(3:24)2014`.

[70] Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Proc. of ICALP'14*, pages 342–353, Berlin, Heidelberg, 2014. Springer. URL: `http://dx.doi.org/10.1007/978-3-662-43951-7_29`, `doi:10.1007/978-3-662-43951-7_29`.

[71] Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12(1), 2016. URL: `http://dx.doi.org/10.2168/LMCS-12(1:5)2016`, `doi:10.2168/LMCS-12(1:5)2016`.

[72] Michael O. Rabin. Weakly definable relations and special automata. In Yehoshua Bar-Hillel, editor, *Mathematical Logic and Foundations of Set Theory*, volume 59 of *Studies in Logic and the Foundations of Mathematics*, pages 1 – 23. Elsevier, 1970. URL: `http://www.sciencedirect.com/science/article/pii/S0049237X08719293`, `doi:https://doi.org/10.1016/S0049-237X(08)71929-3`.

[73] Jeffrey Shallit. *A Second Course in Formal Languages and Automata Theory.* Cambridge University Press, 2008.

[74] P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, and Lakshmi Manasa. Timed automata with integer resets: Language inclusion and expressiveness. In *Proc. of FORMATS'08*, FORMATS '08, pages 78—92, Berlin, Heidelberg, 2008. Springer-Verlag. `doi:10.1007/978-3-540-85778-5_7`.

[75] Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976. URL: `http://dx.doi.org/10.1137/0205019`, `arXiv:http://dx.doi.org/10.1137/0205019`, `doi:10.1137/0205019`.

[76] Martin Tappler, Bernhard K. Aichernig, Kim Guldstrand Larsen, and Florian Lorber. Time to learn - learning timed automata from tests. In Étienne André and Mariëlle Stoelinga, editors, *Proc. of FORMATS'19*, pages 216–235, Cham, 2019. Springer International Publishing.

[77] Ramanathan S. Thinniyam and Georg Zetzsche. Regular Separability and Intersection Emptiness Are Independent Problems. In Arkadev Chattopadhyay and Paul Gastin, editors, *Proc. of FSTTCS'19*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2019/11613`, `doi:10.4230/LIPIcs.FSTTCS.2019.51`.

[78] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, September 2006.

[79] Leslie G. Valiant. Regularity and related problems for deterministic pushdown automata. *J. ACM*, 22(1):1–10, January 1975. URL: `http://doi.acm.org/10.1145/321864.321865`, `doi:10.1145/321864.321865`.

[80] Rüdiger Valk and Guy Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23(3):299–325, 1981. URL: `http://www.sciencedirect.com/science/article/pii/0022000081900672`, `doi:http://dx.doi.org/10.1016/0022-0000(81)90067-2`.

[81] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. An algorithm for learning real-time automata. In *Proc of. the Annual Belgian-Dutch Machine Learning Conference (Benelearn'078)*, 2007.