Statement of scientific achievements

March 2019

1 Name

Lorenzo Clemente

2 Degrees

- PhD in theoretical computer science, University of Edinburgh, 2012.
  Thesis: *Generalized Simulation Relations with Applications in Automata Theory.*
- MSc in computer engineering, University of Rome "Tor Vergata", 2008.
  Thesis: *Algorithms and Tools for the Formal Verification of Concurrent Systems.*

3 Professional career

- University of Edinburgh (UK), 2008–2011 (3 years), teaching assistant.
- Université de Bordeaux (France), 2011–2013 (2 years), post-doc.
- Université Libre de Bruxelles (Belgium), 2013–2014 (7 months), post-doc.
- University of Warsaw (Poland), 2014–2016 (2 years), post-doc.
- University of Warsaw (Poland), 2016–present, associate professor.

4 Scientific achievements

4.1 Title

Adding time to infinite state systems

4.2 Publications included in the scientific achievements

[A] Lorenzo Clemente, Sławomir Lasota. *Binary reachability of timed pushdown automata via quantifier elimination.* In Proc. of ICALP'18, pages 118:1–118:14.

[B] Lorenzo Clemente, Sławomir Lasota, Ranko Lazić, and Filip Mazowiecki. *Timed pushdown automata and branching vector addition systems.* In Proc. of LICS'17, pages 1–12.

[C] Lorenzo Clemente, Sławomir Lasota. *Timed Pushdown Automata Revisited.* In Proc. of LICS'15, pages 738–749.

[D] Lorenzo Clemente and Sławomir Lasota. *Reachability Analysis of First-order Definable Pushdown Systems.* In Proc. of CSL'15, pages 244–259.

[E] Lorenzo Clemente, Frédéric Herbreteau, and Grégoire Sutre. *Decidable Topologies for Communicating Automata with FIFO and Bag Channels.* In Proc. of CONCUR'14, pages 281–296.

[F] Lorenzo Clemente, Frédéric Herbreteau, Amelie Stainer, and Grégoire Sutre. *Reachability of Communicating Timed Processes.* In Proc. of FOSSACS'13, pages 81–96.

4.3 Description of the results

4.3.1 Introduction

Our every day life increasingly depends on the correct functioning of ICT systems (Information and Communication Technology), which come in a wide variety of forms, such as embedded systems, cryptocurrencies (e.g., Bit Coin), communication protocols, and computer software. Malfunctioning ICT systems can cause loss of money at best, and loss of human life in critical applications. For a long time the paradigmatic example of the importance of the correct functioning of ICT systems was that of the Ariane-5 rocket, whose explosion in 1996 due to a flaw in a numeric procedure caused the loss of the ∼7 billion USD spent during a decade of research. Nowadays, cryptocurrencies provide an even more striking example, such as the infamous *DAO attack* on Ethereum [**?**], whereby an attacker exploited a vulnerability of the corresponding smart contract and managed to put the equivalent of ∼60 million USD under her control.

The area of *formal methods* addresses the problem of showing that ICT systems do not have bugs *formally* (i.e., *mathematically*). This is achieved with a variety of techniques, such as theorem proving and model checking. In the *model checking* approach, one starts by building a mathematical model of the system. Given a specification in some logical formalism, one verifies that the model of the system meets the specification, i.e., the absence of bugs, through the exhaustive exploration of all its reachable states [**?**]. Classically, the model checking approach has been applied to systems composed of finitely many states. This guarantees that the exploration procedure terminates after a finite amount of time, and thus model checking is a completely automated approach. This contrasts with theorem proving, which relies on the user to manually provide a formal proof of the correctness of the system, which then the theorem prover can verify. On the one hand, this means that theorem proving is not a completely automated procedure (at least in its classical incarnation). On the other hand, theorem proving can be applied to the wider class of systems with *infinitely many states*.

There has been a trend starting in the 90's aiming at extending the scope of model checking from finite to infinite systems (but of course finitely represented) [**?**]. Infinite state systems are

modelled as (infinite) *transition systems*, i.e., graphs, $G = (S, \rightarrow)$, where S is a set of *states* and $\rightarrow \subseteq S \times S$ is the *transition relation*, whereby we write $s \rightarrow t$ whenever the system can go from state $s \in S$ to state $t \in S$ in one discrete step. In this overview, we will concentrate on the following algorithmic question, which is arguably the most fundamental algorithmic problem in formal verification.

THE REACHABILITY PROBLEM
Input: A finite presentation of a (potentially infinite) graph $G = (S, \rightarrow)$ and two vertices $s, t \in S$.
Output: Is it the case that $s \rightarrow^* t$?

(In the box above, "$\rightarrow^*$" is the *reachability relation*, i.e., the reflexive and transitive closure of the transition relation "$\rightarrow$".) The importance of the reachability problem stems from the fact that it can be used to prove that a system will never reach an erroneous configuration. Once we have identified a decision problem, such as reachability, the first question is whether this problem is decidable, i.e., whether there exists an algorithm which, given an instance of the problem, always terminates, and answers YES precisely when $s \rightarrow^* t$ holds.

Decidability: Is the reachability problem decidable?

Fundamental negative results in computability theory tell us that the reachability problem is undecidable in general, i.e., for Turing machines [**?**]. Nonetheless, there are non-trivial classes of infinite state systems with a decidable reachability problem, which are obtained by extending finite systems with discrete data structures, such as

1. a single pushdown stack, yielding *pushdown automata* [**?**, **?**] which are used to model programs with recursive procedures;
2. non-negative integer counters, yielding *Petri nets* (a.k.a. vector addition systems) [**?**, **?**] which are used to model concurrent systems comprised of unboundedly many processes;
3. FIFO communication channels, yielding *communicating automata* [**?**, **?**], and their lossy variant [**?**, **?**] which are used to model communicating protocols.

Establishing that reachability is decidable for a class of infinite state systems (as the ones above) is just the first step towards a more complete understanding thereof. The second natural question is determining the precise computational complexity of the reachability problem, i.e., whether the problem belongs to complexity classes such as LOGSPACE, PTIME, NPTIME, etc.

Complexity: What is the computational complexity of the reachability problem?

It can be argued that knowing the exact *theoretical* computational complexity of a decision problem does not necessarily have an implication on the *empirical* complexity in solving instances of the problem that occur in practice (i.e., the SAT problem is NPTIME-complete, but modern SAT solvers are blazingly fast on real-life SAT instances), and therefore this raises the question of why this is an important question to settle. We will give three answers. Determining the exact complexity of a decision problem 1) is a precise mathematical question, which can be given a precise mathematical answer—this is not the case with the empirical complexity; 2) can be taken as a synonym of having understood the problem, at least as a first approximation; 3) requires new mathematical insights interesting on their own, and those insights may have practical consequences.

All the systems mentioned above are *discrete* in the sense that a single transition $s \rightarrow t$ cannot be further decomposed into smaller units. This level of abstraction is too coarse if one is interested in explicitly modelling more quantitative aspects of the computation, such as the amount of time necessary to execute a transition. This is important, since eliminating the timing information from a real-time system may result in the introduction of spurious bugs in the model, which cannot

otherwise appear in the real system, thus leading to false positives when looking for bugs. The most successful model addressing this issue is that of *timed automata* (TA) [**?**], which extend finite transition systems with *clocks*, i.e., real-valued variables that can be manipulated in order to impose timing constraints on the behaviour of the system. Notwithstanding the increased expressive power of timed automata, their reachability problem is decidable, with optimal PSPACE complexity.

These two lines of research, that of extending finite systems with either unbounded discrete data structures, or with timing information, have happily been remarried in many works from the last two decades. Examples include *timed Petri nets* [**?**, **?**], *timed lossy channel systems* [**?**], *timed pushdown automata* [**?**], and *timed communicating automata* [**?**].

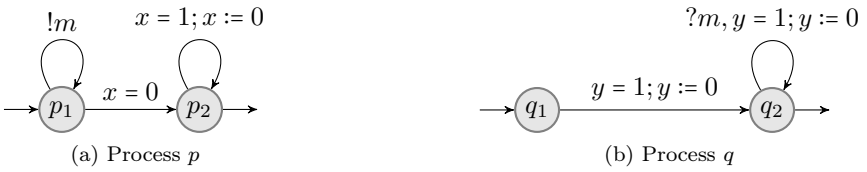
Outline. In this overview, we will concentrate on our new decidability and complexity results about the last two models, namely, timed pushdown automata and timed communicating automata. More precisely, our works [F] and [E] investigate decidability and complexity of the reachability problem for timed communicating automata in the setting of *arbitrary communication topologies*, thus extending [**?**] which addresses fixed topologies with one or two processes only. In the work [C] and [D] we investigate *timed-register pushdown automata* (TRPDA), an alternative approach of introducing timing constraints for pushdown automata, namely, by using *timed-registers* (instead of clocks, as it is most commonly done). While TRPDA have in general an undecidable reachability problem, under a technical assumption (called orbit-finiteness) we show that TRPDA retain decidability of the reachability problem; moreover, we identify a subclass of orbit-finite TRPDA simulating with optimal complexity timed pushdown automata from [**?**]. In the work [B] we show that we can in fact lift the technical assumption above while preserving decidability; we achieve this by uncovering an interesting connection with an expressive model called *branching vector addition systems* (BVAS). Finally, in the work [A] we extend *timed pushdown automata* (TPDA) from [**?**] with expressive integral and fractional clock constraints, and we show that, for this more general model, we can solve the reachability problem (and in fact the more general binary reachability problem; cf. Sec. **??**). In the rest of this section we explain the results above in more detail.

4.3.2 Timed communicating automata [F]

Communicating automata (CA) are a fundamental model for studying concurrent processes exchanging messages over unbounded channels [**?**, **?**]. However, the model is Turing-powerful, and even basic verification questions, like reachability, are undecidable. To obtain decidability, various restrictions have been considered, including allowing only one kind of messages to be sent [**?**], making channels unreliable [**?**, **?**, **?**], or restricting to half-duplex communication [**?**] (later generalized to mutex [**?**]). Decidability can also be obtained when restricting to executions satisfying additional restrictions, such as bounded context-switching [**?**], or bounded channels. Finally, and this is the restriction that we are interested in, decidability is obtained by constraining the communication topology. We say that a communication topology is a *polyforest* if it does not contain an undirected cycle. It is well-known that reachability is decidable (and in fact, PSPACE-complete) if, and only if, the topology is a s.t. [**?**, **?**].

An extension of communicating automata with timing constraints (in the form of clocks) was studied in [**?**], where the authors show that reachability is decidable for the two process topology $p \to q$ and undecidable for the three process topology $p \to q \to r$. The undecidability result above crucially relies on the so-called *urgent semantics* for receptions, whereby if a message can be received by a process, then all other internal actions are disabled. This gives a further synchronisation facility, which, coupled with the implicit synchronisation through the elapse of time, yields the undecidabiliy result above.

We study *timed communicating automata* (TCA) generalising [**?**] over arbitrary communication topologies. We show that the urgent semantics is equivalent to channel *emptiness tests*, allowing us to express our results over general communication topologies in a uniform way. Our first main result provides a complete characterisation of decidable communication topologies in the setting



(a) Process p (b) Process q

Figure 1: Channels cannot be bounded in timed communicating automata.

of discrete time; in the statement below, a *polytree* is a connected component in a polyforest (cf. [Theorem 3, F]).

Theorem 1 (TCA decidability [F]) *The reachability problem for discrete time* TCA *is decidable if, and only if, the communication topology is a polyforest and in every polytree there exists at most one channel which can be tested for emptiness.*

Over the more general dense time, we show decidability in the test-free case (cf. [Theorem 5, F][1]). This extends the previous work [**?**] regarding both decidable and undecidable topologies. First, our characterisation above vastly extends the class of decidable topologies: Not only $p \to q$ is decidable, but also, for instance, $p \to q \to r$ (and $p \to q \leftarrow r$) provided at most one of the two channels has emptiness tests/urgent semantics. Second, our characterisation also extends the class of undecidable topologies: For instance, the four process topology $p \to q \to r \to s$ is undecidable as soon as any two channels have emptiness tests/urgent semantics.

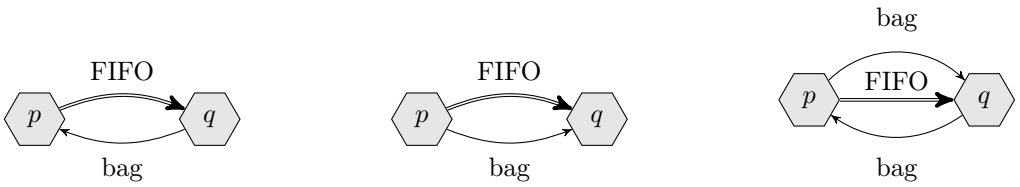
In order to give an intuition of which technical difficulties have to be overcome in order to obtain decidability, consider the simple (decidable) communication topology $p \to q$. Without time, decidability of the reachability problem follows from the simple observation that we can restrict our attention to schedulings immediately matching every transmission $!m$ of p with a corresponding reception $?m$ of q, which keeps the channel length bounded (by 1); this yields a finite state system equivalent w.r.t. reachability to the original one. In the presence of time, this technique breaks down, and this happens already in discrete time. For an illustration, consider the two processes p and q in Fig. **??**. The two clocks x, y are initially 0. Process p starts in control location p_1 and from there can send an unbounded number of messages m to q. At some point it can move to control location p_2, *without elapsing time* as required by the guard $x = 0$. Process q starts in control location q_1, elapses one time unit and goes to q_2 (as required by the guard $y = 1$), and the clock y is reset (as requried by $y := 0$). From the latter location, process q can read one message $?m$ per time unit. The crucial point is that q needs to wait one time unit before any message can be received, while p can send an unbounded number of messages within the first time unit. Thus, it would not be complete to restrict the executions by bounding the size of the channel, because there are runs which cannot be rescheduled as to keep the channel bounded. This is the essential difficulty in analysis systems comprising channels and timing requirements.

If we forbid emptiness tests (which do not increase the expressive power in the untimed setting), our characterisation from Theorem **??** is the same as in the untimed setting. However, in the timed setting the complexity provably worsens, and this happens already over discrete time. This is our second major result (cf. [Corollary 1, F]).

Theorem 2 (TCA complexity [F]) *Already over discrete time, the reachability problem for* TCA *is not elementary.*

The lower bound above is shown by using connected topologies to simulate Petri nets/counter machines, and using the recent breakthrough on the complexity of the latter [**?**]. While the complexity above may seem discouraging, if we forbid emptiness tests and ignore the value of clocks at the end of the run, then the complexity becomes ExpSpace (by reduction to the coverability problem for Petri nets/counter machines, which is in ExpSpace [**?**]).

[1] In yet unpublished work we actually proved that Theorem **??** holds even in dense time [**?**].



(a) Opposite channels (decidable) (b) Parallel channels (decidable) (c) Both ways (undecidable)

4.3.3 Communicating automata with FIFO and bag channels [E]

The work [E] studies a generalisation of TCA over discrete time. Our proof of Theorem **??** [F] shows that we can simulate discrete-time TCA with untimed CA by introducing extra communication channels: More precisely, each TCA FIFO channel $p \to q$ can be simulated by replacing it with two (untimed) CA channels, one FIFO $p \to q$ and the other bag $q \to p$, going in the opposite direction. A *bag channel* is one where the messages can be freely reordered, and thus it is weaker than a FIFO channel in general. We show with a general argument that the resulting topology, shown in Fig. **??**, has a decidable reachability problem. On the other hand, if we allow the channel $q \to p$ to be FIFO, then we obtain an undecidable topology, since it forms an undirected FIFO cicle (i.e., it is not a polyforest [**?**, **?**]).

This naturally raises the question of which communication topologies mixing FIFO and bag channels have a decidable reachability problem[2]. The motivation of studying such mixed topologies is not restricted to the simulation of discrete-time TCA. Bag channels can be used to model asynchronous procedure calls in models where such calls can be freely reordered [**?**, **?**, **?**]. Additionally, bag channels offer a non-trivial over-approximation of FIFO channels, turning undecidable FIFO topologies into decidable ones. For example, we prove that Fig. **??** and **??** are decidable, which would not be the case were both channels FIFO. We also have non-trivial examples of undecidable FIFO/bag topologies: The previous two decidable examples are maximal, since we show that combining them yields an undecidable topology; cf. Fig. **??**.

Our main result is a complete characterisation of which topologies $\mathcal{T}$ of FIFO and bag channels have a decidable reachability problem. In the statement below, P is a decidable property of topologies involving certain cycles of FIFO and bag channels (cf. [Section 5, E] for a formal definition and [Theorem 5.3, E] for the characterisation); for instance P holds for the first two topologies above, but not for the third one.

Theorem 3 (Characterisation of decidable topologies [E]) *The reachabiliy problem for* CA *over a topology $\mathcal{T}$ of FIFO and bag channels is decidable if, and only if, $P(\mathcal{T})$ holds.*

Notice that a topology of only bag channels is effectively equivalent to a Petri net, and thus reachability is decidable. Thus, is always possible to over-approximate FIFO CA by turning *all* channels into bags. Thanks to our characterisation, a much finer analysis is obtained by selectively over-approximating only *some* of the FIFO channels (while preserving decidability).

4.3.4 Timed-register pushdown automata [B], [C], [D]

In the works [B], [C], and [D] we study the problem of introducing timing constraints into pushdown automata. An early attempt in the literature is *pushdown timed automata* (PDTA) [**?**], which extend classical pushdown automata by replacing the finite control with a timed automaton. Thus, the stack in a PDTA is just a classical untimed stack, which severely limits the expressive power of the model. For instance, PDTA cannot recognise natural timed context-free languages such as even-length *timed palindromes* (below w^R is the reversal of w):

$$L = \{ww^R \mid w \in (\Sigma \times \mathbb{R})^*\}. \tag{1}$$

[2]The similar problem of mixing perfect and *lossy* FIFO channels has been studied in [**?**].

For a while, a candidate remedy to this situation seemed to be *dense-timed pushdown automata* (DTPDA) [**?**], which extend PDTA with *stack clocks* that can be pushed on and popped according to non-diagonal interval constraints. One can say that DTPDA are "infinite in two dimensions", since the stack is unbounded, and each stack symbol carries a clock, which is also unbounded. Our work started from the observation that DTPDA semantically collapse to PDTA, in the sense that the two models are effectively equivalent w.r.t. the class of timed languages they recognise (cf. [Theorem II.1, C]).

Theorem 4 (DTPDA=PDTA [C]) *For every* DTPDA *one can effectively construct a* PDTA *recognising the same timed language.*

The somehow surprising result above is the consequence of the unexpected interplay between the monotonicity of time and the well-nested stack discipline. In fact, we show that the same holds even when allowing more liberal *diagonal* clock constraints between stack and control clocks. The question arises as to whether there is any sensible timed extension of pushdown automata not suffering from a semantic collapse to its "timeless stack" version (as it is the case with DTPDA). This is the starting point of our enquiry.

4.3.5 Set with atoms I: Oligomorphic and homogeneous atoms [D]

One answer to the problem above is to be found in the *sets with atoms* approach, a uniform way of extending automata theory—and in fact, mathematics—obtained by relaxing the notion of finiteness to orbit-finiteness [**?**] (cf. also the recent book [**?**]). The setting is parameterised with a relational structure $\mathbb{A}$, called *atoms*. The simplest such example is that of *equality atoms* $(\mathbb{N}, =)$, where the domain is any countably infinite set and the only relational symbol in the signature is equality. A richer structure is *(dense) total order atoms* $(\mathbb{Q}, \leq)$, which can be used to model a qualitative notion of dense time. We shall use total order atoms as an illustrative example for the rest of the section.

The key intuition is that two sets with atoms are indistinguishable if there exists an atom *automorphism*[3] mapping one into the other; in this case, we say that they are in the same *orbit*. For instance, automorphisms of total order atoms are precisely the monotone isomorphisms of $\mathbb{Q}$; the two tuples $\bar{a} = (2, 1.1, 2)$ and $\bar{b} = (3, 2.9, 3)$ are indistinguishable since there exists an automorphism $\alpha : \mathbb{Q} \to \mathbb{Q}$ extending $[2 \mapsto 3, 1.1 \mapsto 2.9]$ s.t. $\alpha(\bar{a}) = \bar{b}$; more generally, the orbit of $\bar{a}$ is precisely $\{(a, b, c) \in \mathbb{Q}^3 \mid a = c, b < a\}$.

The crucial notion in the theory of sets with atoms is that of orbit-finiteness, which is a generalisation of finiteness: A set with atoms is *orbit-finite* if it can be partitioned into finitely many orbits. In other words, an orbit-finite set is finite up to automorphism. For example, $\mathbb{Q}$ and $\{(a, b) \in \mathbb{Q}^2 \mid a \neq b\}$ are both orbit-finite: The former has exactly one orbit (itself), and the second has two orbits, $\{(a, b) \in \mathbb{Q}^2 \mid a < b\}$ and $\{(a, b) \in \mathbb{Q}^2 \mid a > b\}$.

In our context, we are interested in generalising pushdown automata to the setting of sets with atoms. This is achieved by lifting all components in the usual textbook definition of pushdown automata (such as input and stack alphabets, control locations, and transition relation) from finite sets to sets with atoms definable in first-order logic. We call the resulting model *(first-order) definable pushdown automata* (DPDA). Each choice of the atoms $\mathbb{A}$ yields a different notion of pushdown automaton. For instance, if we take $\mathbb{A}$ to be equality atoms $(\mathbb{N}, =)$, then we obtain a model of register pushdown automata studied in [**?**]. More choices of atoms will be described below. We have studied the reachability problem for definable pushdown automata over well-behaved classes of atoms. We say that a relational structure $\mathbb{A}$ is *oligomorphic* if $\mathbb{A}^k$ is orbit-finite for every k [**?**], and that it is *(first-order) decidable* if the satisfiability problem for first-order logic over $\mathbb{A}$ is decidable. For instance, equality and total order atoms are oligomorphic and decidable; we will give later an example of non-oligomorphic structure. Our first result is that DPDA over decidable oligomorphic atoms have a decidable reachability problem (cf. [Theorem 4, D]).

[3] An automorphism of a relational structure $\mathbb{A}$ is an isomorphism of the domain of $\mathbb{A}$ which preserves and reflects the interpretation of all relational symbols of $\mathbb{A}$.

Theorem 5 (Oligomorphic DPDA [D]) *Let $\mathbb{A}$ be a decidable oligomorphic relational structure. The reachability problem is decidable for definable pushdown automata over $\mathbb{A}$.*

(In fact we have shown a much more general result, namely that, under the assumptions of the theorem, DPDA enjoy a generic saturation procedure based on orbit-finite automata [**?**].) The result above is very general, since it only requires that the first-order satisfiability problem be decidable for the atoms, without any complexity assumption. This generality comes at the price that no complexity upper bound on the DPDA reachability problem can be given without further assumptions.

We say that a relational structure $\mathbb{A}$ is *homogeneous* if every partial isomoprhism extends to an automorphism of the whole structure [**?**]. Homogenuousness has the pleasant consequence that the number of orbits of $\mathbb{A}^k$ is exponentially bounded[4] In particular, homogeneous structures are oligimorphic. (There are oligimorphic relational structures which are not homogeneous, such as bit vector atoms [**?**].) Our second result is that the DPDA reachability problem becomes *fixed-parameter tractable* over homogeneous structures, in the sense that its running time is of the form $O(f(k) \cdot \mathsf{poly}(n))$, where n is the size of the DPDA and k the dimension (cf. [Theorem 7, Corollary 8, D]).

Theorem 6 (Homogeneous DPDA [D]) *Let $\mathbb{A}$ be a homogeneous relational structure. The reachability problem for* DPDA *over $\mathbb{A}$ is fixed-parameter tractable, with the dimension being the parameter.*

Our third result is that, for commonly occurring structures, we can do even better. Let $\mathbb{A}$ and $\mathbb{B}$ be two relational structures. We say that $\mathbb{A}$ is a *substructure* of $\mathbb{B}$ if they have the same signature and the domain of $\mathbb{A}$ is included in that of $\mathbb{B}$; if additionally the intepretation of all relation symbols agree on the domain of $\mathbb{B}$, then we say that $\mathbb{A}$ is an *induced substructure* of $\mathbb{B}$. The *induced substructure problem* amounts to determine, given two relational structures $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A}$ is an induced substructure of $\mathbb{B}$. Our observation is that the induced substructure problem is efficiently solvable, i.e., in PTime, for the following commonly occurring homogenous relational structures ([Section 6, D]; cf. [**?**]):

- Equality atoms $(\mathbb{N}, =)$: They can be used to model data values from an infinite set which can only be compared with equality.
- Total order atoms $(\mathbb{Q}, \leq)$: As mentioned above, they can be used to model the total order between real-time events.
- Betweenness order atoms $(\mathbb{Q}, B)$, where $(x, y, z) \in B$ iff either $y < x < z$ or $z < x < y$: They can be used to model real-time events where one can only observe whether an event x occurs between two other events y and z, but has no further information on the ordering of y and z themselves.
- Cyclic order atoms $(\mathbb{Q}, K)$, where $(x, y, z) \in K$ iff $x < y < z$, or $y < z < x$, or $z < x < y$: They can be used to model the ordering of the fractional values of timestamps of events. This is a natural choice, since K is invariant under time elapse: $(x, y, z) \in K$ iff $(x \oplus \delta, y \oplus \delta, z \oplus \delta)$ for every $x, y, z \in [0, 1)$ and $\delta \in \mathbb{R}$, where $a \oplus b$ is the fractional part of $a + b$.
- Partial order atoms: They can be used to describe real-time events which can occurr independently of each other, but still need to satisfy certain causality constraints.
- Tree order atoms: They can be used to model the dynamic creation of processes which do not interact further in the future.
- Other examples include: Equivalence, preorder, graph, and tournament atoms.

[4] Two tuples $\bar{a}, \bar{b} \in \mathbb{A}^k$ are isomorphic precisely when they satisfy the same relations of $\mathbb{A}$, and thus there are $O(2^{\mathsf{poly}(k)})$ equivalence classes of tuples. Since every partial isomorphism extends to an automorphism of $\mathbb{A}$, the same quantity above bounds the number of orbits of $\mathbb{A}^k$.

The following result applies to each of the previous examples (cf. [Corollary 10, D]).

Theorem 7 (Tractable homogeneous DPDA [D]) *Let $\mathbb{A}$ be a homogeneous relational structure with a* PTime *induced substructure problem. The reachability problem for definable pushdown automata over $\mathbb{A}$ is in* ExpTime*. Moreover, it is* ExpTime*-hard already for equality atoms $(\mathbb{N}, =)$.*

The list of atoms above shows the wide applicability of Theorem **??**[5]. The major limitation of all the structures previously mentioned is that they are essentially *qualitative*, and thus not apt at modelling more quantitative aspects of real-time events, such as their exact length. The next section is devoted to remedy this situation.

4.3.6 Set with atoms II: Timed atoms [C]

In order to model more quantitative features of real-time systems, we consider richer structures, such as (in increasing order of expressive power)

- *discrete time* atoms $(\mathbb{Z}, +1, \leq)$,
- *dense time* atoms $(\mathbb{Q}, +1, \leq)$, and
- *hybrid time* atoms $\mathbb{H} = (\mathbb{Z}, +1, \leq) \uplus (\mathbb{Q}, \leq)$, a two-sorted structure with a discrete component for integral values of timestamps and a dense component for fractional values of timestamps.

We collectively call the structures above *timed atoms*, and we call *unconstrained timed-register pushdown automata* (unconstrained trPDA) definable pushdown automata over timed atoms. None of the timed atoms is oligomorphic (and thus not homogeneous)[6], and our generic technique from Sec. **??** cannot be applied to trPDA. Even worse, the reachability problem for definable finite automata over discrete or dense time atoms is in fact undecidable, and this holds already for discrete time atoms in dimension 3: The set of control locations is a subset of $\mathbb{Z}^3$ and the three components (x, y, z) simulate a two counter Minsky machine (which has an undecidable reachability problem [**?**]): Incrementing/decrementing counters are simulated with definable transitions $(x, y, z) \mapsto (x \pm 1, y, z)$ for the first counter and $(x, y, z) \mapsto (x, y \pm 1, z)$ for the second counter, and zero tests by $x = z$ and $y = z$, resp.

The source of undecidability is that one can remember timestamps which can be arbitrarily far from each other (such as x and z above). For definable automata, this issue is immediately solved by requiring that the set of control locations be orbit-finite. The resulting model of *orbit-finite automata* over timed atoms has a decidable reachability problem [**?**]. However, this is not sufficient for unconstrained trPDA, since one can still encode two Minsky counters even if control locations are orbit-finite, already in dimension one: The first counter is encoded as the height of the stack (as classically done), and the second counter is encoded as the difference between the control value and the value on top of the stack (cf. [Theorem IV.1, C]).

Theorem 8 (trPDA [C]) *The reachability problem for unconstrained* trPDA *is undecidable.*

The simulation above is made possible by the fact that push operations are allowed to see the value on top of the stack; (cf. classical pushdown automata, where such a feature does not increase the expressiveness of the model). As a solution, we forbid push operations from reading the top of the stack altogether, and obtain what we call below just trPDA. The expressiveness of trPDA is witnessed by the fact that they recognise timed palindromes (**??**), which can be done as follows. The automaton works in two phases. In the first phase, it reads an input symbol of the form $(a, x) \in \Sigma \times \mathbb{R}$ and pushes it on the stack. Then, the automaton nondeterministically guesses

[5] The complexity upper bound on the induced substructure problem is a necessary assumption, since we can construct homogeneous structures whose induced substructure problem can simulate the membership problem in arbitrary subsets of $\mathbb{N}$ [Theorem 9, D].

[6] For instance, already $\mathbb{Z}^2$ contains infinitely many orbits: For every integer $\mathbb{Z}^2$, the set $\{(x, x + k) \mid x \in \mathbb{Z}\}$ is a distinct orbit of $\mathbb{Z}^2$. This is due to the fact that automorphisms of $\mathbb{Z}$ need to preserve the distance between points, and thus there are "not enough" automorphism.

the middle of the word, and switches to the second phase where it reads from the input (a, x) and simultaneously pops (a, y) from the stack provided $x = y$. A trPDA can even go beyond context-free languages: Consider the language of untimed palindromes over the alphabet $\Sigma = \{a, b\}$ containing the same number of a's and b's:

$$M = \{ww^R \mid w \in \Sigma^*, \#_a(w) = \#_b(w)\}. \tag{2}$$

Language M can be recognised by a trPDA as follows: Initially, the automaton pushes $(\bot, x)$ on the stack and remembers x in the state, where $x \in \mathbb{Z}$ is a guessed integer to be used at the end of the run. The rest is similar to timed palindromes; additionally, the local control value x is increased/decreased by 1 upon reading a/b. At the end of the run, in order to check that the word contained the same number of a's and b's, the automaton pops $(\bot, x)$ only if x is the same as the local control value.

Notwithstanding the fact that trPDA languages go beyond the class of context-free languages, our main result is that they are decidable (cf. [Theorem 1, B]).

Theorem 9 (trPDA [B]) *The reachability problem for* trPDA *is in* 2ExpTime.

The result above is obtained by establishing a tight connection between trPDA and *branching vector addition systems* in dimension one with addition and subtraction ($\mathbb{Z}$-BVASS) [**?**]. The latter is an expressive generalisation of Petri nets/counter automata/vector addition systems, which in its basic variant with addition is decidable in PTime in dimension one [**?**]; in higher dimension, this is a long-standing open problem in the infinite-state systems community [**?**]. Our main technical result leading to Theorem **??** is that $\mathbb{Z}$-BVASS are decidable, and in fact, elementary (cf. [Theorem 6, B]).

Theorem 10 ($\mathbb{Z}$-BVASS [B]) *The reachability problem for* $\mathbb{Z}$-BVASS *in dimension one is in* ExpTime.

We show that the complexity improves for the following progressively less expressive subclasses of trPDA. In an *orbit-finite* trPDA we require that the combined set of control locations and top-most stack symbols be orbit-finite (hence the name). Under this restriction, we are able to show an improved upper complexity bound (cf. [Theorem IV.8, C]).

Theorem 11 (orbit-finite trPDA [C]) *The reachability problem for orbit-finite* trPDA *is in* NExpTime *and* ExpTime*-hard.*

Moreover, if we forbid timing information from the stack altogether, we obtain an even better complexity (cf. [Theorem IV.5, C]). This is particularly interesting, since trPDA with timeless stack suffice to simulate dtPDA and PDTA [Corollary IV.10, C], and in this case the complexity is optimal since the latter models are already ExpTime-hard [**?**].

Theorem 12 (trPDA with timeless-stack [C]) *The reachability problem for* trPDA *with timeless stack is* ExpTime*-complete.*

Finally, and this may be the most interesting trPDA subclass, trPDA over *monotonic time* are also ExpTime-complete [**?**, Corollary 6.8].

trPDA constitute an expressive and yet decidable class of timed infinite-state systems combining recursion with real-time constraints. This was achieved by deviating from the standard clock-based approach, and studying instead a register-based model in the general framework of sets with atoms. The question remains whether we can find an expressive model of timed pushdown automata within the realm of real-time clocks. This is what we tackle in the next section.

4.3.7 Timed pushdown automata [A]

The semantic collapse of Theorem **??** shows that traditional clock constraints are not sufficient to exploit the power of a timed stack. For example, consider the language of timed odd length palindromes over $\Sigma = \{a, b\}$ s.t. matching symbols are at integral distance:

$$N = \{(a_0, x_0)\cdots(a_{2n}, x_{2n}) \in (\Sigma \times \mathbb{R})^* \mid \forall (0 \le i \le n) \cdot a_i = a_{2n-i} \wedge x_i - x_{2n-i} \in \mathbb{N}\}. \tag{3}$$

Clearly, we need a timed stack to recognise N, since we need to remember the fractional part of the x_i's that we are pushing on the stack in the first part of the run in order to check that it equals the fractional part of the x_{2n-i}'s in the second part of the run. In Sec. **??**, we addressed this issue by introducing timed-register pushdown automata (TRPDA), a register-based model obtained by instantiating definable pushdown automata to timed atoms. However, the question arises as to whether a classical clock based model can recognise truly timed context-free languages such as N.

We show in [A] that this is indeed the case. We introduce *timed pushdown automata* (TPDA), a model extending classical pushdown automata with both control and stack clocks, which can be pushed to and popped from the stack. Since the the stack is unbounded, TPDA are "infinite in two dimensions". As time elapses, all clocks increase their values at the same rate. Clocks can be compared according to boolean combinations of classical and fractional *clock constraints*[7]:

	(classical)	(fractional)
(non-diagonal)	$x \le k, \quad k \in \mathbb{Z}$	$\{x\} = 0,$
(diagonal)	$x - y \le k, \quad k \in \mathbb{Z}$	$\{x\} \le \{y\}.$

Control clocks can be reset and compared according to the constraints above. At the time of a push operation, new stack clocks are created whose values are initialised, possibly non-deterministically, as to satisfy a given push constraint between stack clocks and control clocks; similarly, a pop operation requires that stack clocks to be popped satisfy a given pop constraint of analogous form.

The timed stack of a TPDA cannot be untimed (unlike for DTPDA; cf. [**?**] for a similar observation) since we can construct a TPDA recognising N above: There is one control clock x, which is initially 0, and is never reset. In the first phase, the automaton reads input symbol $c \in \Sigma$, and pushes it on the timed stack together with a fresh stack clock y satisfying the diagonal constraint $x = y$. In the second phase, the automaton reads $c \in \Sigma$ and pops (c, y) from the stack, where y is the stack clock associated with stack symbol c, provided it has the same fractional value as the control clock x, i.e., $\{x\} = \{y\}$.

In the context of TPDA, it is natural to consider the reachability relation ograniczona restricted to initial and final configurations with empty stack. Let X be the finite set of control clocks, let L be the finite set of control locations, and let $\mathbb{R}^X_{\ge 0}$ be the set of clock valuations. In the rest of this section, the *reachability relation* is the family of relations $\{\rightsquigarrow_{pq}\}_{p,q \in L}$ defined as follows: For two clock valuations $\mu, \nu \in \mathbb{R}^X_{\ge 0}$ and control locations $p, q \in L$,

$$\mu \rightsquigarrow_{pq} \nu$$

precisely when there is a run starting from control state p, control clocks valuation μ, and empty stack, ending in control state q, control clocks valuation ν, and empty stack. For fixed control locations p, q, the reachability relation $\rightsquigarrow_{pq}$ is a subset of $\mathbb{R}^X_{\ge 0} \times \mathbb{R}^X_{\ge 0}$ and can thus be described with a logical formula over the real numbers. We study the *binary reachability problem* for TPDA, which is more general than mere reachability: Instead of checking whether a *given* source configuration (p, μ) can reach a given target configuration (q, ν), i.e., $\mu \rightsquigarrow_{pq} \nu$, we are interested in expressing the reachability relation $\rightsquigarrow_{pq}$ itself with a formula φ_{pq} in a decidable logical formalism. We say that φ_{pq} *expresses* $\rightsquigarrow_{pq}$ if, for every clock valuations $\mu, \nu \in \mathbb{R}^X_{\ge 0}$,

$$\mu \rightsquigarrow_{pq} \nu \quad \text{iff} \quad \mu, \nu \vDash \varphi_{pq}.$$

[7] We denote with $\{x\}$ the fractional value of clock x.

Reachability reduces to binary reachability, since given a formula φ_{pq} expressing $\rightsquigarrow_{pq}$, we can decide whether $\mu \rightsquigarrow_{pq} \nu$ holds by simply evaluating φ on μ, ν.

Our main result is to show that TPDA reachability relations are expressible in an efficiently decidable logic. Let $\mathcal{A}_{\mathbb{Z}} = (\mathbb{Z}, \leq, (\equiv_m)_{m\in\mathbb{N}}, +, (k)_{k\in\mathbb{Z}})$ and $\mathcal{A}_{\mathbb{Q}} = (\mathbb{Q}, \leq, +, (k)_{k\in\mathbb{Z}})$, with the expected interpretation of symbols in the signature. The first-order languages of these structures are the well-known *Presburger arithmetic* and, resp., *rational arithmetic*. We consider a more general two-sorted structure $\mathcal{A} = \mathcal{A}_{\mathbb{Z}} \uplus \mathcal{A}_{\mathbb{Q}}$, whose first-order language we call *linear arithmetic*. Intuitively, we use the first sort to describe the integral values of clocks, and the second sort to describe the fractional values of clocks. The following is our main result in the algorithmic analysis of TPDA (cf. [Theorem 5, A], first part).

Theorem 13 (TPDA [A]) *The* TPDA *reachability relation* $\rightsquigarrow$ *can be expressed by a family of existential linear arithmetic formulas computable in* 2ExpTime.

Without a stack, i.e., for *timed automata* (TA) [**?**], and, more generally, for a timeless stack, i.e., for PDTA [**?**], the complexity improves by one exponential (cf. [Theorem 5, A], second part).

Theorem 14 (Timeless stack TPDA and TA [A]) *The reachability relation* $\rightsquigarrow$ *for timeless stack* TPDA *and for* TA *can be expressed by a family of existential linear arithmetic formulas computable in* ExpTime.

Our results above improve similar expressibility results from the literature on TA [**?**, **?**, **?**, **?**] and PDTA [**?**]. Along the way, we prove a result interesting on its own about effective elimination of quantifiers for a sublogic of linear arithmetic corresponding to clock constraints [Corollary 3, A]. Quantifier elimination allows us to reduce to TPDA where all constraints are fractional. Finally, we reduce fractional TPDA to DPDA over cyclic order atoms, introduced earlier in Sec. **??**, to which we apply our previous results [D]. This is interesting, since it shows the applicability of a register model, such as DPDA, to the analysis of a clock model, such as TPDA.

4.3.8 Conclusions

We have provided an extensive investigation of the reachability problem for expressive classes of infinite state systems combining unbounded discrete data structures with expressive timing constraints, and thus "infinite in two dimensions". In particular, we have considered FIFO communication channels in our works [E] and [F], and pushdown stacks in [A], [B], [C], and [D].

A general picture emerging from our investigations is that, generally speaking, the reachability problem stays decidable, albeit the computational complexity sometimes worsens. For instance, in the case of communicating automata (CA; works [E] and [F]) it was known that reachability is decidable iff the topology is a polyforest. From our results it follows that, if we add time to the model (obtaining TCA), then the set of decidable topologies does not change (without emptiness tests; cf. Theorem **??**). However, the complexity is ExpSpace-complete if we allow the processes to elapse an arbitrary amount of time at the end of the run, and non-elementary in the general case (cf. Theorem **??**). This should be contrasted with the fact that, in the untimed setting, reachability is PSpace-complete.

Similarly, for timed pushdown automata (TPDA) the reachability problem is still decidable, however, the complexity worsens from PTime in the untimed setting, to ExpTime-complete in the timed setting. On the positive side, the ExpTime upper bound holds for DPDA over a very large class of "qualitative data" (tractable homogeneous atoms; cf. Theorem **??** [D]), for trPDA and TPDA with timeless stack (cf. Theorem **??** [C] and, resp., **??** [A]), and trPDA over monotonic time [**?**, Corollary 6.8]. For the even more expressive classes of trPDA and TPDA (with timed stack), we have provided a 2ExpTime complexity upper bound (cf. Theorem **??** [B] and, resp., **??** [A]). As future work, it remains to investigate whether the 2ExpTime complexity above is optimal.

5 Other publications

5.1 Publications on higher-order recursive schemes

The following two papers concern the algorithmic analysis of *higher-order recursive schemes* (HORS), a very expressive formalism used to model higher-order recursion, i.e., functions taking other functions as argument.

- Lorenzo Clemente, Paweł Parys, Sylvain Salvati, Igor Walukiewicz. *Ordered Tree-Pushdown Systems.* In Proc. of FSTTCS'15, pages 163–177 [**?**].

  In this paper we propose an alternative approach on the verification of HORS based on tree-pushdown automata, which generalise ordinary pushdown automata by allowing the stack to be a tree instead of a word. We prove a general *preservation of regularity* result: The inverse image $(\to^*)^{-1}(T)$ of the reachability relation $\to^*$ of a regular set of trees T is itself a regular set of trees. The result is effective, in the sense that given a finite tree automaton for the target set T, we compute a finite tree automaton for the inverse image of T.

- Lorenzo Clemente, Paweł Parys, Sylvain Salvati, Igor Walukiewicz. *The Diagonal Problem for Higher-Order Recursion Schemes is Decidable.* In Proc. of LICS'16, pages 96–105 [**?**].

  The main contribution of this work is proving that the following non-trivial non-regular property of HORS is decidable: Is it the case that, for every bound $n \in \mathbb{N}$, the HORS recognises a finite tree where each letter from a finite alphabet occurrs at least n times? Note that such unboundedness property is not regular, and thus decidability does not follow from [**?**]. Decidability of the diagonal problem has a number of interesting consequence, such as computability of the downward closure over finite words [**?**], decidability of the separability problem by piecewise testable languages [**?**], and decidability of reachability of parameterized asynchronous shared-memory concurrent systems [**?**].

5.2 Publications on separability problems

A set S *separates* two sets A, B if $S \subseteq A$ and $S \cap B = \varnothing$. Given two classes of sets $\mathcal{A}$ and $\mathcal{S}$, the *separability problem* asks whether two given sets $A, B \in \mathcal{A}$ can be separated by some set $S \in \mathcal{S}$. The separability problem is a central topic in computer science (such as in formal language theory and computability theory) and mathematics (for example in topology). In computer science separability is intimately related to other important problems, such as computing downward closures, the emptiness of intersection problem, and the characterisation problem. In the following papers we study two separability problems.

- Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. *Separability of Reachability Sets of Vector Addition Systems.* In Proc. of STACS'17, pages 24:1–24:14 [**?**].

  We consider as input sets $\mathcal{A}$ the class of subsets of $\mathbb{N}^k$ which are reachability sets of vector addition systems (and generalisation thereof), and as separators $\mathcal{S}$ the class of modular and unary subsets of $\mathbb{N}^k$. Our main result is that the separability problem is decidable. This is a first step towards a more ambitious goal: Establishing decidability of regular separability of languages recognised by vector addition systems.

- Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. *Regular Separability of Parikh Automata.* In Proc. of ICALP'17, pages 117:1–117:13 [**?**].

  We consider as input sets $\mathcal{A}$ languages (i.e., subsets of Σ^*) recognised by Parikh automata (finite automata with acceptance by counting transitions), and as separators $\mathcal{S}$ the class of regular languages. Our main result is decidability of the separability problem. This is somewhat surprising, since the regularity problem for Parikh automata is undecidable. On the way, we show decidability of separability of semilinear sets of $\mathbb{N}^k$ by unary and modular sets.

5.3 Publications on stochastic games

The following two publications study stochastic games [?] on finite and infinite graphs.

- Lorenzo Clemente and Jean-François Raskin. *Multidimensional beyond Worst-Case and Almost-Sure Problems for Mean-Payoff Objectives.* In Proc. of LICS'15, pages 257–268 [?] (cf. also the survey[?]).

  The *beyond worst-case* (BWC) problem combines determinstic 2-player games and stochastic 1-player games: In its original formulation [?], it asks whether the player can win against an arbitrary deterministic opponent, and at the same optimise the expected value w.r.t. a (fixed) stochastic model of the opponent. Our main result is coNPTime-completeness of the multi-dimensional BWC problem on finite graphs, for both finite-memory and unrestricted strategies; we also show that, if we relax the first requirement to almost-sure winning, then the problem becomes solvable in PTime. This generalises previous work in one-dimension [?].

- Parosh Abdulla, Lorenzo Clemente, Richard Mayr, and Sven Sandberg. *Stochastic Parity Games on Lossy Channel Systems.* Logical Methods in Computer Science, 2014, volume 10, number 4 [?] (special journal issue of the conference paper [?]).

  We study a class of stochastic games played on infinite, yet well-behaved graphs, namely, configuration graphs of lossy channel systems. We consider almost-sure winning w.r.t. ω-regular objectives and we show that such games are decidable under the constraint that both player use finite-memory strategies. This is a necessary assumption, since they are undecidable already for almost-sure co-Büchi objectives [?].

5.4 Publications on language inclusion and automata simplification

The following works tackle the related problems of efficiently deciding language inclusion of two automata and efficiently reducing the size of automata. Checking inclusion between is a central problem in automata theory, with important applications such as *model-checking* [?] and *size-change termination* [?]. Automata simplification is fundamental in automata-based decision procedures for logical theories [?], in the analysis of infinite state systems such as Petri nets [?], and in regular model checking [?]. Both problems are PSpace-complete in general, and thus efficient algorithms capable of solving practically arising instances are important. We focus of languages of infinite words recognised by nondeterministic and alternating Büchi automata.

- Lorenzo Clemente and Richard Mayr. *Multipebble simulations for alternating automata.* In Proc. of CONCUR'10, pages 297–312 [?].

  A successful approach to the reduction of the number of states of an automaton is that of quotienting. It relies on the identification of equivalence relations on states s.t. 1) computing the relation should be efficient, and 2) quotienting should preserve the language recognised by the automaton. Suitable simulations and bisimulation relations have previously been defined for nondeterministic [?] and alternating Büchi automata [?]. A generalisation to multipebble simulations was proposed for nondeterministic Büchi automata [?]. We join the last to works and study multipebble simulations for alternating Büchi automata. We show that, multipebble simulations imply language inclusion, a variant thereof can be used for quotienting, and they can be computed in PTime for any fixed number of pebbles.

- Parosh Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukáš Holík, Chih-Duo Hong, Richard Mayr, and Tomáš Vojnar. *Simulation subsumption in Ramsey-based Büchi automata universality and inclusion testing.* In Proc. of CAV'10, pages 132–147 [?].

  We propose an efficient algorithm to solve the universality and inclusion problems for nondeterministic Büchi automata. It was previously shown how to use set-based subsumption to improve the efficiency of a Ramsey-based algorithm for the universality problem [?]. We

improve the approach by using simulation relations to obtain even larger subsumption relations, and we extend the method from universality to the more general inclusion problem. We show that this results in a significant performance gain on both random automata and test cases taken from mutual exclusion algorithms.

- Parosh Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukáš Holík, Chih-Duo Hong, Richard Mayr, and Tomáš Vojnar. *Advanced Ramsey-based Büchi automata inclusion testing.* In Proc. of CONCUR'11, pages 187–202 [**?**].

  We improve on the previous point by defining an even coarser subsumption relation based on forward as well as backward simulations, plus other algorithmic improvements.

- Lorenzo Clemente. *Büchi automata can have smaller quotients.* In Proc. of ICALP'11, pages 258–270 [**?**].

  We investigate generalisations of simulation relations for Büchi automata suitable for quotienting. We propose *fixed-word delayed simulation* to be the maximal such relation which can be used for quotienting. To support this claim, we show that its multipebble extensions collapses to the one-pebble case.

- Lorenzo Clemente and Richard Mayr. *Efficient reduction of nondeterministic automata with application to language inclusion testing.* Logical Methods in Computer Science, volume 15, issue 1, 2019 [**?**] (journal version of [**?**]).

  We present efficient algorithms to reduce the size of nondeterministic finite and Büchi automata. We propose extensions of classic quotienting of the state space, as well as innovative techniques for removing transitions (pruning) and adding transitions (saturation)[8]. These techniques use criteria based on combinations of backward and forward trace inclusions and simulation relations. Since trace inclusion relations are themselves PSPACE-complete, we introduce *lookahead simulations* as good PTIME approximations thereof. The quality of the reduction is witnessed by the fact that it often reduces universal automata to the trivial one-state universal automaton. In general, extensive experiments show that our algorithm consistently reduces the size far more than all previous techniques.

[8] Adding transitions may cause more states to be quotiented together.