The ABCs of Petri net reachability relaxations

Michael Blondin, Université de Sherbrooke, Canada

Petri nets form a widespread model of concurrency well suited for the verification of systems with infinitely many configurations. Deciding configuration reachability in Petri nets, which plays a central role in their formal analysis, suffers from a nonelementary time complexity lower bound. We survey relaxations that alleviate this tremendous complexity, both for classical Petri nets and for extensions with affine transformations, branching rules and colored tokens.

1. INTRODUCTION

Petri nets are a widespread formalism to model and analyze concurrent systems with possibly infinite configuration spaces over nonnegative counters, *i.e.* $\mathbb{N}^k$. They offer a great tradeoff between graphical modeling and amenability to algorithmic analysis. In particular, they find applications ranging from program verification (*e.g.* [German and Sistla 1992; Kaiser et al. 2014; Atig et al. 2011; Delzanno et al. 2002]) to the formal analysis of chemical, biological and business processes (*e.g.* [Esparza et al. 2017; Heiner et al. 2008; van der Aalst 1998]). One of the central questions in Petri net theory consists in determining whether a given target configuration, typically an error of a system, is reachable from an initial configuration. This reachability problem, which we will shortly define formally, has been extensively studied over several decades. For a long time, the computational complexity of the problem lay between EXPSPACE-hardness [Lipton 1976] and decidability [Mayr 1981; Kosaraju 1982; Lambert 1992; Leroux 2012]. However, it was recently narrowed down between TOWER-hardness [Czerwiński et al. 2019], *i.e.* a tower of exponentials, and Ackermaniann time [Leroux and Schmitz 2019].

Due to this colossal complexity, practical applications have avoided solving (exact) reachability for Petri nets with infinitely many configurations.[1] For example, the annual Model Checking Contest focuses on bounded Petri nets.[2] More generally, applications have often relied on subproblems (*e.g.* the coverability problem), structural restrictions (bounded configuration spaces, bounded reversals, acyclicity, flatness, etc.), structural analysis (traps/siphons, place invariants, etc.) and approximations.

We report on a specific type of approximations, coined *reachability relaxations*, which has renewed interest in the verification of infinite systems. Here, relaxations refer to approximations based on enlarged configuration domains (*e.g.* $\mathbb{Z}^k$ or $\mathbb{R}_+^k$), which share a resemblance with relaxations in the field of mathematical optimization.

In the rest of this article, which could have perhaps been entitled *"Dodging hard problems with simpler ones,"* we discuss the complexity of reachability relaxations for Petri nets and three of their extensions (affine transformations, branching rules and colored tokens), with a bias towards recent work of the author.

2. PETRI NETS

A *Petri net* is a bipartite weighted directed graph described by a triple $\mathcal{N} = (P, T, W)$ where P and T are finite disjoint sets, whose elements are called *places* and *transitions*, and where $W \colon (P \times T \cup T \times P) \to \mathbb{N}$ assigns weights to arcs connecting places and transitions. A *marking* of $\mathcal{N}$ is a vector from $\mathbb{N}^P$ that indicates the number of *tokens* in each of its places. Figure 1 depicts a (marked) Petri net.

[1] One exception being [Dixon and Lazić 2020] who implemented the algorithm of [Kosaraju 1982] this year.
[2] See https://mcc.lip6.fr/.

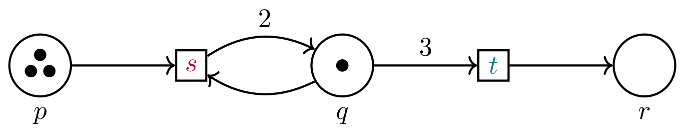

Fig. 1. Example of a Petri net with $P = \{p, q, r\}$, $T = \{s, t\}$, $W(p, s) = W(q, s) = W(t, r) = 1$ (weight omitted on arcs), $W(s, q) = 2$, $W(q, t) = 3$ and $W(-, -) = 0$ elsewhere (arcs with zero weight not shown). The net is marked with $[p\colon 3, q\colon 1]$.

A transition t is *firable* in some marking $\boldsymbol{u}$ if each ingoing place p of t contains at least as many tokens as the arc weight from p to t, *i.e.* $\boldsymbol{u}(p) \geq W(p, t)$. We call this requirement the *firing constraint* of t. If it holds, then firing t consumes the respective number of tokens from each place, and produces as many tokens as specified by arcs from t, *i.e.* it leads from marking $\boldsymbol{u}$ to marking $\boldsymbol{v}$ defined as:

$$\boldsymbol{v} \coloneqq \boldsymbol{u} + \underbrace{[p\colon W(t, p) - W(p, t) \mid p \in P]}_{\text{the "effect" vector of } t \text{ which we denote } \boldsymbol{\Delta}(t)}.$$

We write such a firing as $\boldsymbol{u} \xrightarrow{t} \boldsymbol{v}$. For example, for the Petri net of Figure 1, we have:

$$[p\colon 3, q\colon 1] \xrightarrow{s} [p\colon 2, q\colon 2] \xrightarrow{s} [p\colon 1, q\colon 3] \xrightarrow{t} [p\colon 1, r\colon 1].$$

Note that neither s nor t is firable in the last above marking since place q is empty.

The notation naturally extends to sequences, which we write $\xrightarrow{\sigma}$ or simply $\xrightarrow{*}$, depending on whether specifying the firing sequence $\sigma \in T^*$ is relevant or not. This yields a binary relation over markings, known as the *reachability relation*, which gives rise to the central decision problem for Petri nets:

Reachability

GIVEN:	a Petri net $\mathcal{N}$, and markings $\boldsymbol{u}$ and $\boldsymbol{v}$;
DETERMINE:	whether $\boldsymbol{u} \xrightarrow{*} \boldsymbol{v}$.

3. REACHABILITY RELAXATIONS

3.1. Pseudo-reachability

Recall that the Petri net reachability problem has nonelementary complexity: solving it requires a tower of exponentials of time and space [Czerwiński et al. 2019]. This may be surprising to someone not accustomed to Petri nets, as reachability may appear to boil down to solving an integer linear program. However, this is not the case: the difficulty lies in the firing (nonnegativity) constraints.

Nonetheless, this holds for deciding *pseudo-reachability*, *i.e.* whether $\boldsymbol{u} \overset{*}{\dashrightarrow} \boldsymbol{v}$ holds, where $\dashrightarrow$ is defined like $\longrightarrow$ but without any firing constraint, and consequently where places may temporarily hold negative numbers of tokens. Indeed, the order in which transitions are fired becomes irrelevant, and hence it suffices to solve the following system of linear Diophantine equations, which is known as the *marking equation*:

$$\exists \boldsymbol{x} \in \mathbb{N}^T : \boldsymbol{v} - \boldsymbol{u} = \sum_{t \in T} \boldsymbol{\Delta}(t) \cdot \boldsymbol{x}(t).$$

Solving the marking equation simply amounts to checking the feasibility of an integer linear program, which is well-known to be NP-complete. While it may be considered intractable by some, this complexity pales in comparison to the TOWER-hardness of reachability.

Pseudo-reachability turns out to be relevant in practice. Indeed, to disprove that a marking representing an error state cannot be attained, it suffices to show that it is not pseudo-reachable. For example, using *only* the marking equation, [Esparza et al. 2014] could verify some safety properties to be satisfied by 84 concurrent systems out of 115 instances arising from mutual exclusion algorithms, communication protocols, multi-threaded C programs with shared-memory, ERLANG programs, and systems modeling message provenance analysis of a bug-tracking system and a medical messaging system.[3]

3.2. Continuous reachability

Using pseudo-reachability to approximate reachability comes at a great price: control over the order in which transitions can be fired is entirely lost. An alternative avenue consists in preserving (nonnegative) firing constraints, but relaxing the *discreteness* of markings: We allow the firing constraints and effects of transitions to be scaled by any factor $\lambda \in (0, 1]$, provided the number of tokens remains nonnegative. Hence, in this setting, places may contain "pieces of tokens" (which could be seen as liquid).

Formally, we write $\boldsymbol{u} \xrightarrow{\lambda t}\!\!\twoheadrightarrow \boldsymbol{v}$ iff $\boldsymbol{u}(p) \geq \lambda \cdot W(p, t)$ for every incoming place $p \in P$ of t, and $\boldsymbol{v} = \boldsymbol{u} + \lambda \cdot \boldsymbol{\Delta}(t)$. Note that we use a double arrow tip for this relaxation, while dotted lines specified pseudo-reachability (we will use shortly $\cdots\!\!\twoheadrightarrow$ for a combination of both). This gives rise to the *continuous reachability* relation where sequences are drawn from

$$T^\dagger \coloneqq ((0, 1] \times T)^* \text{ instead of } T^* \cong (\{1\} \times T)^*.$$

For example, for the Petri net depicted in Figure 1, we have:

$$[p\colon 3, q\colon 1] \xrightarrow{1s}\!\!\twoheadrightarrow [p\colon 2, q\colon 2] \xrightarrow{\frac{2}{3}t}\!\!\twoheadrightarrow [p\colon 2, r\colon 2/3].$$

Note that neither s nor t is firable in the last above marking since place q is empty.

Petri nets equipped with continuous reachability are perhaps more commonly known as *continuous Petri nets*. The latter were introduced by [David and Alla 1987; David and Alla 2010] to model, *e.g.*, physical systems depending on continuous variables.

This relaxation also incurs a cost. For example, if places count the number of threads at some location of a replicated concurrent program, then a thread may now split in *half*, which surely cannot happen in reality. Yet, some ratios are preserved between places, and we cannot obtain negative amounts as with pseudo-reachability. Moreover, continuous reachability turns out to be *easier* than the latter: it is P-complete.

This precise complexity has been established by [Fraca and Haddad 2015] who described a polynomial time algorithm exploiting their following characterization:

THEOREM 3.1. *We have $\boldsymbol{u} \xrightarrow{*}\!\!\twoheadrightarrow \boldsymbol{v}$ iff there exist $\boldsymbol{u}', \boldsymbol{v}' \in \mathbb{R}_+^P$ and $\sigma, \sigma_{\blacktriangleright\!\blacktriangleright}, \sigma_{\blacktriangleleft\!\blacktriangleleft} \in T^\dagger$ s.t. all three sequences use exactly the same transitions (possibly organized differently) and*

$$\boldsymbol{u} \overset{\sigma}{\cdots\!\!\twoheadrightarrow} \boldsymbol{v},\ \boldsymbol{u} \xrightarrow{\sigma_{\blacktriangleright\!\blacktriangleright}}\!\!\twoheadrightarrow \boldsymbol{u}',\ \boldsymbol{v}' \xrightarrow{\sigma_{\blacktriangleleft\!\blacktriangleleft}}\!\!\twoheadrightarrow \boldsymbol{v}.$$

By "organized differently," we mean that transitions appearing in σ, $\sigma_{\blacktriangleright\!\blacktriangleright}$ and $\sigma_{\blacktriangleleft\!\blacktriangleleft}$ may be ordered differently, with distinct scaling factors, and with varying numbers of occurrences, *i.e.* what matters is whether a transition appears at least once or not at all.

Theorem 3.1 states that continuous reachability amounts to *continuous pseudo-reachability* — which relaxes *both* nonnegativity and discreteness — provided there exist sequences $\sigma_{\blacktriangleright\!\blacktriangleright}$ and $\sigma_{\blacktriangleleft\!\blacktriangleleft}$ that witness forward and backward firability (regardless of the markings reached).

[3]They could verify 96 instances by combining the marking equation with a more sophisticated approach.

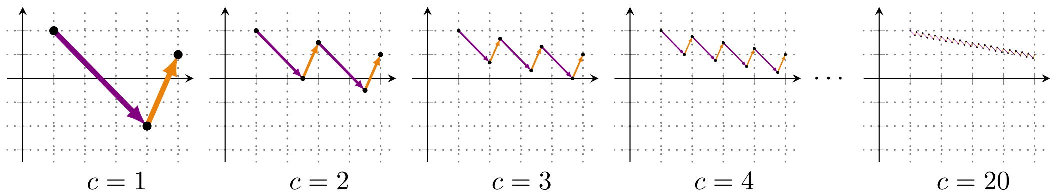

Fig. 2. Illustration of $\text{✂}_c(st)$, where $\mathbf{\Delta}(s) = (3, -4)$, $\mathbf{\Delta}(t) = (1, 3)$ and $(1, 2) \overset{st}{\dashrightarrow\!\!\!\!\rightarrow} (5, 1)$. The x-axis and y-axis each indicate the tokens count of a place, and each point depicts a (continuous) marking.

Let us give some insights on why Theorem 3.1 involves three sequences. A natural question is whether continuous reachability simply amounts to continuous *pseudo-reachability*, *i.e.*:

$$\boldsymbol{u} \overset{*}{\twoheadrightarrow} \boldsymbol{v} \overset{?}{\iff} \boldsymbol{u} \overset{*}{\dashrightarrow\!\!\!\!\rightarrow} \boldsymbol{v}.$$

Of course, the implication from left to right holds as continuous pseudo-reachability is less restrictive, but what about the one from right to left?

Let $\boldsymbol{u} \overset{\sigma}{\dashrightarrow\!\!\!\!\rightarrow} \boldsymbol{v}$. Consider the sequence $\text{✂}_c(\sigma) \coloneqq \overbrace{(1/c)\sigma \cdots (1/c)\sigma}^{\text{repeated } c \text{ times}}$, where $(1/c)\sigma$ corresponds to σ in which all transitions are multiplied by $1/c$. Note that $\text{✂}_c(\sigma)$ globally consumes and produces the same amount of tokens as σ, as each transition is scaled down by $1/c$ and copy/pasted c times. Figure 2 depicts an example of this transformation where $\sigma = st$ with $\mathbf{\Delta}(s) = (3, -4)$ and $\mathbf{\Delta}(t) = (1, 3)$. In this example, we have

$$(1, 2) \overset{\sigma}{\dashrightarrow\!\!\!\!\rightarrow} (5, 1), \text{ but } \textit{not } (1, 2) \overset{\sigma}{\twoheadrightarrow} (5, 1),$$

as a place drops below zero (see the y-axis for $c = 1$ in Figure 2). However, we *do* have

$$(1, 2) \xrightarrow{\text{✂}_3(\sigma)} (5, 1).$$

Therefore, if c tends to infinity, then $\text{✂}_c(\sigma)$ becomes a "straight line" with the same initial and target markings (see Figure 2 up to $c = 20$). Hence, we *informally* have

$$\boldsymbol{u} \xrightarrow{\text{✂}_\infty(\sigma)} \boldsymbol{v}.$$

Of course, we cannot set $c = \infty$ as the sequence must remain finite. But, $c = 3$ suffices in our example, and it is tempting to conclude that we can always pick c large enough.

Unfortunately, this is not always the case. Consider the example depicted in Figure 3 which is a slight modification of our previous example where we go from $(1, 1)$ to $(5, 0)$ rather than from $(1, 2)$ to $(5, 1)$. There, no matter the value of c, the second place (y-axis) will always be negative at some point, although increasingly closer to zero. This is due to the fact that we are attempting to reach zero *from below*.

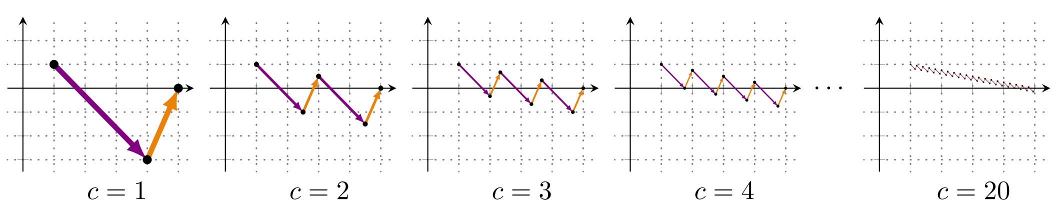

Fig. 3. Illustration of $\text{✂}_c(st)$ as in Figure 2, but starting from marking $(1, 1)$ rather than $(1, 2)$.

It can be shown that we can find $c \in \mathbb{N}$ such that $\boldsymbol{x} \overset{\tau}{\dashrightarrow\!\!\!\twoheadrightarrow} \boldsymbol{y}$ implies $\boldsymbol{x} \xrightarrow{\text{✄}_c(\tau)}\!\!\!\twoheadrightarrow \boldsymbol{y}$, if:

(a) $\boldsymbol{x}(p) > 0$ for each place p from which τ ever consumes tokens;
(b) $\boldsymbol{y}(p) > 0$ for each place p in which τ ever produces tokens.

These conditions are the reason behind sequences $\sigma_{\blacktriangleright\blacktriangleright}$ and $\sigma_{\blacktriangleleft\blacktriangleleft}$ appearing in Theorem 3.1.

With these insights in mind, let us give a proof sketch of the implication going from right to left in Theorem 3.1. We must turn markings $\boldsymbol{u}$ and $\boldsymbol{v}$ into markings satisfying (a) and (b). Pictorially, we have:

$$\begin{tikzcd} \boldsymbol{u} \arrow[rrr, dotted, two heads, "\sigma"] \arrow[dr, two heads, "\sigma_{\blacktriangleright\blacktriangleright}"'] & & & \boldsymbol{v} \\ & \boldsymbol{u}' & \boldsymbol{v}' \arrow[ur, two heads, "\sigma_{\blacktriangleleft\blacktriangleleft}"'] & \end{tikzcd}$$

By rescaling a sequence $\boldsymbol{x} \xrightarrow{\tau = \lambda_1 t_1 \lambda_2 t_2 \cdots \lambda_n t_n}\!\!\!\twoheadrightarrow \boldsymbol{y}$ with exponentially smaller factors, we can "saturate" the target marking. More precisely, consider the following sequence for some suitable constant $d \in \mathbb{N}$:

$$\boldsymbol{x} \xrightarrow{\lightning(\tau) \coloneqq (\lambda_1/d)t_1(\lambda_2/d^2)t_2 \cdots (\lambda_n/d^n)t_n}\!\!\!\twoheadrightarrow \boldsymbol{y}'.$$

This operation, denoted $\lightning(\cdot)$, yields a marking $\boldsymbol{y}'$ with *as many nonempty* places as possible. By rescaling globally with a small factor $\lambda \in (0, 1]$, we can further obtain a vector $\boldsymbol{y}''$ arbitrarily close to $\boldsymbol{x}$, as "almost nothing" is fired. The same idea also applies backwards, *i.e.* going from a target marking $\boldsymbol{y}$ to some small saturated marking $\boldsymbol{x}''$. Thus, using the same notation informally for both directions, we obtain:

$$\begin{tikzcd} \boldsymbol{u} \arrow[rrr, dotted, two heads, "\sigma"] \arrow[dr, two heads, "\underbrace{\lambda \cdot \lightning(\sigma_{\blacktriangleright\blacktriangleright})}_{\sigma'_{\blacktriangleright\blacktriangleright}}"'] & & & \boldsymbol{v} \\ & \boldsymbol{u}'' & \boldsymbol{v}'' \arrow[ur, two heads, "\overbrace{\lambda \cdot \lightning(\sigma_{\blacktriangleleft\blacktriangleleft})}^{\sigma'_{\blacktriangleleft\blacktriangleleft}}"'] & \end{tikzcd}$$

Recall that σ, $\sigma_{\blacktriangleright\blacktriangleright}$ and $\sigma_{\blacktriangleleft\blacktriangleleft}$ all use the exact same set of transitions. This is also the case for $\sigma'_{\blacktriangleright\blacktriangleright}$ and $\sigma'_{\blacktriangleleft\blacktriangleleft}$, as operation $\lightning(\cdot)$ only changes scaling factors, not transitions. Hence, markings $\boldsymbol{u}''$ and $\boldsymbol{v}''$ now both satisfy (a) and (b).

Moreover, $\boldsymbol{u}'' \overset{\sigma}{\dashrightarrow\!\!\!\twoheadrightarrow} \boldsymbol{v}''$ *almost* holds as these markings are respectively *very close* to $\boldsymbol{u}$ and $\boldsymbol{v}$. Fortunately, we can bridge this tiny gap. Indeed, since λ was picked sufficiently small and since all three sequences share the same transitions, we can "subtract" both $\sigma'_{\blacktriangleright\blacktriangleright}$ and $\sigma'_{\blacktriangleleft\blacktriangleleft}$ from σ, which corresponds to decreasing scaling factors occurring within σ.

Therefore, we are done since $\boldsymbol{u} \xrightarrow{\sigma'_{\blacktriangleright\blacktriangleright}\, \text{✄}_c(\sigma - \sigma'_{\blacktriangleright\blacktriangleright} - \sigma'_{\blacktriangleleft\blacktriangleleft})\, \sigma'_{\blacktriangleleft\blacktriangleleft}}\!\!\!\twoheadrightarrow \boldsymbol{v}$ holds for some $c \in \mathbb{N}$:

$$\begin{tikzcd} \boldsymbol{u} \arrow[rrr, dotted, two heads, "\sigma"] \arrow[dr, two heads, "\sigma'_{\blacktriangleright\blacktriangleright}"'] & & & \boldsymbol{v} \\ & \boldsymbol{u}'' \arrow[r, two heads, "\text{✄}_c(\sigma - \sigma'_{\blacktriangleright\blacktriangleright} - \sigma'_{\blacktriangleleft\blacktriangleleft})"'] & \boldsymbol{v}'' \arrow[ur, two heads, "\sigma'_{\blacktriangleleft\blacktriangleleft}"'] & \end{tikzcd}$$

In essence, the algorithm of [Fraca and Haddad 2015] identifies sequence σ in polynomial time via linear programming, while $\sigma_{\blacktriangleright\blacktriangleright}$ and $\sigma_{\blacktriangleleft\blacktriangleleft}$ are obtained by a graph exploration procedure. As the three sequences must use the same transitions, they progressively refine candidate transitions until a greatest fixed point is reached.

3.3. Continuous reachability in practice

While the algorithm of [Fraca and Haddad 2015] for continuous reachability can theoretically run in polynomial time, it relies on solving several linear programs: between a linear and quadratic number depending on the implementation. Moreover, it is risky to use a numerical procedure, such as most industrial implementations of the simplex algorithm. Indeed, floating-point errors could technically lead to erroneous outcomes, wrongly concluding unreachability. This is not particularly desirable in the context of formal verification where unreachability typically corresponds to the absence of errors.

The second issue can be addressed by using an exact implementation of the simplex algorithm (*e.g.* [Applegate et al. 2007]) or an SMT solver supporting linear real arithmetic (*e.g.* [de Moura and Bjørner 2008; Barrett et al. 2011]). However, there exists an alternative approach which relies on a *single* call to an SMT solver [Blondin et al. 2017]. The idea consists in translating the continuous reachability relation into an existentially quantified formula from linear real arithmetic, based on the conditions of Theorem 3.1:

$$\psi_{\overset{*}{\twoheadrightarrow}}(\boldsymbol{u}, \boldsymbol{v}) = \exists \boldsymbol{x} \in \mathbb{R}_+^T : \varphi_{\text{mark-eq}}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{v}) \land \varphi_{\blacktriangleright\!\blacktriangleright}(\boldsymbol{u}, \boldsymbol{x}) \land \varphi_{\blacktriangleleft\!\blacktriangleleft}(\boldsymbol{x}, \boldsymbol{v}).$$

Here, x represents sequence σ of Theorem 3.1 in the sense that $\boldsymbol{x}(t)$ indicates the sum of all scaling factors across occurrences of transition t in σ, and $\varphi_{\text{mark-eq}}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{v})$ is the marking equation over $\mathbb{R}_+$:

$$\varphi_{\text{mark-eq}}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{v}) \coloneqq \left(\boldsymbol{v} - \boldsymbol{u} = \sum_{t \in T} \boldsymbol{\Delta}(t) \cdot \boldsymbol{x}(t) \right).$$

Formulas $\varphi_{\blacktriangleright\!\blacktriangleright}(\boldsymbol{u}, \boldsymbol{x})$ and $\varphi_{\blacktriangleleft\!\blacktriangleleft}(\boldsymbol{x}, \boldsymbol{v})$ check for the existence of firing sequences $\sigma_{\blacktriangleright\!\blacktriangleright}$ and $\sigma_{\blacktriangleleft\!\blacktriangleleft}$ of Theorem 3.1 with respect to transitions $\{t \in T \mid \boldsymbol{x}(t) > 0\}$. By adapting a construction of [Verma et al. 2005], these two formulas can be made of *linear size*, which yields an overall formula $\psi_{\overset{*}{\twoheadrightarrow}}$ of linear size.

Experimental results show that solving $\psi_{\overset{*}{\twoheadrightarrow}}$ allows to efficiently verify safety of concurrent systems in practice [Blondin et al. 2017], using the simple observation that

$$\neg\psi_{\overset{*}{\twoheadrightarrow}}(\boldsymbol{w}_{\text{init}}, \boldsymbol{w}_{\text{error}}) \text{ implies } \neg(\boldsymbol{w}_{\text{init}} \xrightarrow{*} \boldsymbol{w}_{\text{error}}).$$

Moreover, solving $\psi_{\overset{*}{\twoheadrightarrow}}$ works well as a pruning method within a complete procedure such as the backward reachability algorithm [Abdulla et al. 2000], which fits within the more general framework of combining forward invariant generation with backward reachability analysis [Geffroy et al. 2018].

A natural question arising from this approach is whether the full power of existential linear arithmetic is needed. As it turns out, it is *not* the case: continuous reachability is characterized by a fragment of linear arithmetic which admits a *polynomial time* decision procedure [Blondin and Haase 2017]. This fragment $\mathcal{L}$ is a syntactic restriction where variables are quantified over $\mathbb{R}_+$, and where a formula is a conjunction of *convex semi-linear Horn clauses*, which are of the form:

$$(\boldsymbol{a} \cdot \boldsymbol{x} \succ b) \lor \bigvee_{1 \leq i \leq m} \bigwedge_{1 \leq j \leq n_i} \boldsymbol{x}(j) > 0 \quad \text{where each } \boldsymbol{a}(\ell) \in \mathbb{R},\, b \in \mathbb{R} \text{ and } \succ\, \in \{=, \geq, >\}.$$

Clauses of the form "$\boldsymbol{a} \cdot \boldsymbol{x} = b$" and "$\boldsymbol{x}(\ell) > 0 \lor \bigvee_{1 \leq i \leq m} \bigwedge_{1 \leq j \leq n_i} \boldsymbol{x}(j) > 0$" can respectively implement $\varphi_{\text{mark-eq}}$ (immediate) and $\varphi_{\blacktriangleright\!\blacktriangleright} \land \varphi_{\blacktriangleleft\!\blacktriangleleft}$ (much less obvious).

Observe that clauses of the form "$\boldsymbol{a} \cdot \boldsymbol{x} \succ b$" correspond to constraints of linear programs. Moreover, $\mathcal{L}$ can express the family $\{\neg(y_0 > 0) \lor y_1 > 0 \lor \cdots \lor y_k > 0\}_{k > 0}$ which cannot be defined by any convex polytope. Hence, in terms of expressiveness, $\mathcal{L}$ strictly lies in between linear programs and linear arithmetic.

Unfortunately, the current formula of $\mathcal{L}$ for continuous reachability has quadratic size rather than linear size, hence it has not yet been possible to harness its power in practice. Nonetheless, it can be exploited to derive complexity bounds.

For example, *structural cyclicity*, which asks whether $\mathbf{0} \xrightarrow{+} \mathbf{0}$, where "+" stands for any nonempty sequence, can be solved in polynomial time [Drewes and Leroux 2015]. This result can be recast in the logical framework of continuous reachability.

Indeed, it can be shown that $\mathbf{0} \xrightarrow{+} \mathbf{0} \iff \mathbf{0} \xrightarrow{+}\!\!\!\!\rightarrow \mathbf{0}$, and the latter amounts to this formula from $\mathcal{L}$:

$$\exists \boldsymbol{x} \in \mathbb{R}_+^T : \varphi_{\text{mark-eq}}(\mathbf{0}, \boldsymbol{x}, \mathbf{0}) \land \varphi_{\twoheadrightarrow}(\mathbf{0}, \boldsymbol{x}) \land \varphi_{\twoheadleftarrow}(\boldsymbol{x}, \mathbf{0}) \land \bigvee_{t \in T} \boldsymbol{x}(t) > 0.$$

We will see another application of $\mathcal{L}$ in the forthcoming Section 4.3.

3.4. Pseudo-reachability with control-states

As discussed so far, continuous reachability keeps some information on the order in which transition can be fired, whereas pseudo-reachability does not. However, there are settings where the latter can preserve some information as well.

Indeed, the set of places of a Petri net $\mathcal{N}$ can often be viewed as a partition $P = Q \cup C$ where *exactly one* token appears in Q in any reachable marking, due to a control graph structure of $\mathcal{N}$ over Q. Figure 4 depicts such a *Petri net with control-states*.[4] Such structures arise naturally from modeling some types of concurrent systems, *e.g.* if each place of Q represents a valuation from variables to a finite domain (such as Boolean variables), and if C counts the number of threads at some program locations.

Since any marking has the form $[q\colon 1] + \boldsymbol{v}$, where $q \in Q$ and $\boldsymbol{v} \in \mathbb{N}^C$, we can shorten the notation to $q(\boldsymbol{v})$. For example, Figure 4 (left) illustrates marking $q_1(c_1\colon 2, c_2\colon 1)$.

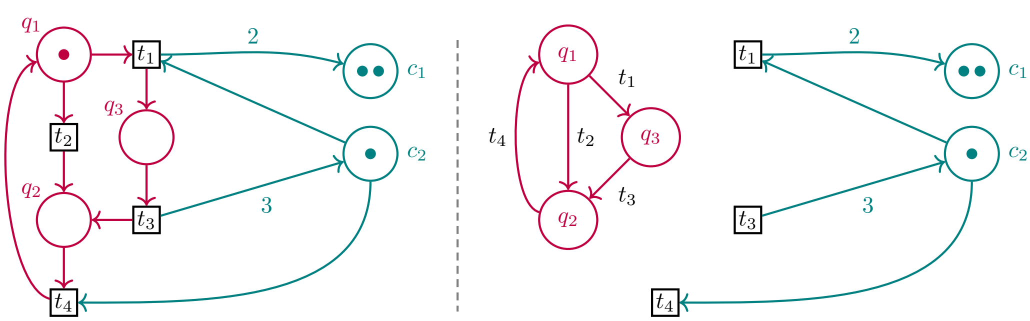

Fig. 4. *Left*: Petri net with control-states $\mathcal{N}$. *Right*: underlying *control graph* $\mathcal{N}|_Q$ and *controlless net* $\mathcal{N}|_C$.

The notion of pseudo-reachability can be refined for Petri nets with control-states: only places from C are allowed to become negative. Alternatively, this can be seen as synchronizing a control graph with a Petri net equipped with pseudo-reachability, or as a variant of so-called blind multicounter machines [Greibach 1978]. It is well-known that pseudo-reachability for Petri nets with control-states is NP-complete (*e.g.* [Haase and Halfon 2014]). The NP-hardness of pseudo-reachability follows from a reduction from a variant of the subset sum problem, while membership to NP results from this characterization:

[4]Perhaps more commonly known as a *vector addition systems with state (VASS)* [Hopcroft and Pansiot 1979].

THEOREM 3.2. *It is the case that $p(\boldsymbol{u}) \overset{*}{\dashrightarrow} q(\boldsymbol{v})$ iff there exists $\boldsymbol{x} \in \mathbb{N}^T$ such that:*

(a) the marking equation of $\mathcal{N}|_C$ is satisfied by $\boldsymbol{x}$; and

(b) $\mathcal{N}|_Q$ has a Eulerian path from p to q if each edge t is replaced by $\boldsymbol{x}(t)$ parallel edges.

By Theorem 3.2 and a result of [Verma et al. 2005], the pseudo-reachability relation of a Petri net with control-states translates into a *linear-size* Presburger formula:

$$\psi_{\overset{*}{\dashrightarrow}}(p(\boldsymbol{u}), q(\boldsymbol{v})) \coloneqq \exists \boldsymbol{x} \in \mathbb{N}^T : \varphi_{\text{mark-eq}}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{v}) \land \varphi_{\text{Euler}}(p, \boldsymbol{x}, q).$$

Hence, the NP-completeness is overcome in practice using efficient SMT solvers, *e.g.* it has been used by [Athanasiou et al. 2016] for unbounded-thread program verification.

It is worth noting that, beyond its practical relevance, pseudo-reachability plays a role in theoretical results such as the decidability of Petri net reachability [Mayr 1981; Kosaraju 1982] (see [Lasota 2018] for a modern presentation), and the flattening of Petri nets with control-states and two places [Leroux and Sutre 2004]. In the specific case where $|C|$ is constant and arc weights are specified in unary, pseudo-reachability with control-states becomes NL-complete [Blondin et al. 2015]; this has been used, *e.g.*, in the context of path queries for graph databases [Michaliszyn et al. 2017].

3.5. Combining continuous reachability and control-states

Continuous reachability also extends to Petri nets with control-states, *i.e.* the continuous relaxation applies only to places from C. This relation translates into a formula of existential linear real arithmetic, which yields an NP-complete complexity [Blondin and Haase 2017]. This can be obtained as follows:

— Cyclic pseudo-reachability, *i.e.* of the form $q(\boldsymbol{u}) \overset{\sigma}{\dashrightarrow\!\!\!\rightarrow} q(\boldsymbol{v})$, allows to define operation $✂_c(\sigma)$ presented in Section 3.2. Indeed, since both endpoints of σ match, the sequence can be scaled and repeated multiple times.

— It is possible to derive a formula for cyclic reachability, *i.e.* of the form $q(\boldsymbol{u}) \xrightarrow{\sigma}\!\!\!\rightarrow q(\boldsymbol{v})$, by carefully combining and adapting Theorem 3.1 and Theorem 3.2.

— Any path of the control graph must alternate between linearly many simple paths and (potentially very long) cycles (see Figure 5). Hence, a linear number of formulas can be "stitched together."

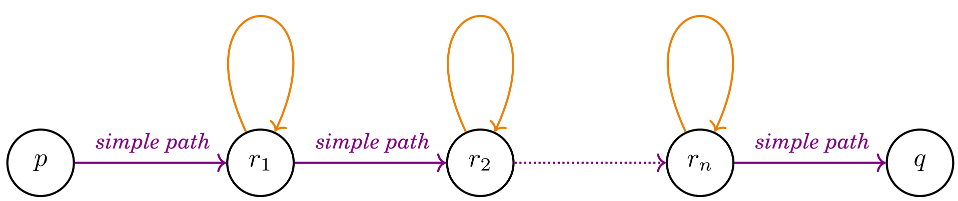

Fig. 5. Decomposition of a path σ that alternates between simple paths and cycles. The cycle from control-state r_i to itself corresponds to the first and last occurrences of r_i in σ, hence $n \leq |Q|$.

4. BEYOND STANDARD PETRI NETS

Petri nets have been extended in various ways in the literature to increase their modeling power. These extensions suffer from TOWER-hard to (possibly) undecidable reachability problems. We discuss relaxations, as presented in the previous section, for Petri nets extended with either affine transformations, branching rules, or colored tokens.

4.1. Affine transformations

Firing transition t of a Petri net from a marking $\boldsymbol{v}$ adds $\boldsymbol{\Delta}(t)$ to $\boldsymbol{v}$. This corresponds to applying the affine transformation $\mathbf{I} \cdot \boldsymbol{v} + \boldsymbol{\Delta}(t)$, where $\mathbf{I}$ is the identity matrix. The model extends naturally to *affine transformations* of the form $\mathrm{M}(t) \cdot \boldsymbol{v} + \boldsymbol{\Delta}(t)$, where $\mathrm{M}(t) \in \mathbb{Z}^{P \times P}$ is a matrix associated to transition t (*e.g.* [Valk 1978]).

This encompasses operations encountered in the literature for modeling more complex concurrent systems, such as:

operation	***description***	***transformation***
reset	empties the contents of a place p	$p \leftarrow 0$
swap	swaps the contents of two places p and q	$p \leftrightarrow q$
transfer	empties the contents of a place q onto a place p	$p \leftarrow p + q;\ q \leftarrow 0$
copy	copies the contents of a place q to a place p	$p \leftarrow q$
doubling	doubles the contents of a place p	$p \leftarrow 2 \cdot p$

For example, resets were used for validating business processes [Wynn et al. 2009] and generating program loop invariants [Silverman and Kincaid 2019], and transfers were employed to verify multithreaded C and JAVA program skeletons with broadcast communication primitives [Kaiser et al. 2014; Delzanno et al. 2002].

It is known that reachability is undecidable for Petri nets extended with affine transformations. In particular, this holds for resets, transfers, copies and doubling, as they can weakly simulate Minsky machines which are Turing-complete (*e.g.* [Araki and Kasami 1976; Dufourd et al. 1998]). In fact, undecidability holds for *any class* of affine transformations beyond permutations, such as swaps [Blondin and Raskin 2020].

Unfortunately, undecidability remains for continuous reachability, even for mild extensions such as resets. For example, resets allow to detect whether any transition of a sequence is ever scaled by some factor $\lambda < 1$, which constrains markings to always remain discrete. The idea is to add two places p and q, and to replace each transition t by two transitions t_p and t_q implementing the following high-level description:[5]

$$t_p \coloneqq \text{if } p > 0 \text{ and } t \text{ is firable, then fire } t;\ p \leftarrow 0;\ q \leftarrow q + 1,$$
$$t_q \coloneqq \text{if } q > 0 \text{ and } t \text{ is firable, then fire } t;\ q \leftarrow 0;\ p \leftarrow p + 1.$$

If the Petri net initially contains a token in p and none in q, then reachable markings satisfy the invariant $0 < p + q \leq 1$. Hence, transition t can be simulated by alternating between t_p and t_q. Moreover, $p + q < 1$ holds iff any of t_p or t_q is ever scaled by $\lambda < 1$.

Surprisingly, pseudo-reachability is sometimes decidable:

operation	***pseudo-reachability with control-states***
reset	NP-complete [Chistikov et al. 2018]
swap	PSPACE-complete [Blondin et al. 2018; Blondin and Raskin 2020]
transfer	PSPACE-complete [Blondin et al. 2018; Blondin and Raskin 2020]
copy	PSPACE-complete [Blondin et al. 2018; Blondin and Raskin 2020]
doubling	undecidable [Reichert 2015]

A trichotomy on the complexity of pseudo-reachability with control-states was established this year: it is either NP-complete, PSPACE-complete or undecidable for any

[5] I was made aware of this construction a few years ago by Piotr Hofman.

so-called *class* of affine transformations [Blondin and Raskin 2020]. Moreover, decidability only holds for classes of matrices generating a finite monoid (under multiplication). When this finite monoid property holds, but there is no parameterization w.r.t. a class of matrices, then the problem belongs to EXPSPACE [Bumpus et al. 2020].

Although most of these results are currently of theoretical nature, there is hope to employ pseudo-reachability for practical purposes, *e.g.* for the case of transfers and its relation to communication primitives. In fact, [Silverman and Kincaid 2019] recently leveraged continuous pseudo-reachability with resets — whose decidability follows by an adaptation of [Chistikov et al. 2018] — to generate program loop invariants.

4.2. Branching rules

Petri nets are sometimes extended with branching rules, *i.e.* a marking $\boldsymbol{u}$ can either be updated by a standard (unary) transition t, or "split" by a binary transition t into a pair of markings $(\boldsymbol{v}_L, \boldsymbol{v}_R)$ such that $\boldsymbol{u} = \boldsymbol{v}_L + \boldsymbol{v}_R - \boldsymbol{\Delta}(t)$. In this setting, computations are trees rather than sequences, and the goal is to reach a root target marking from a set of initial vectors allowed to appear at the leaves. This model, which essentially generalizes tree automata, has ramifications in program verification [Bouajjani and Emmi 2013; Majumdar and Wang 2013], logic [de Groote et al. 2004; Bojańczyk et al. 2009], computational linguistics [Schmitz 2010] and the formal study of cryptographic protocols [Verma and Goubault-Larrecq 2005]. The TOWER-hardness [Lazić and Schmitz 2015] of reachability was established prior to standard Petri nets, and its decidability remains unknown to this day.

Given the titanic complexity and possible undecidability of reachability, relaxations could be relevant for this model. Few results are known at the moment, but the author and colleagues[6] are currently investigating different relaxations. For example, pseudo-reachability with control-states, so with branching of the form $p(\boldsymbol{u}) \mapsto (q_L(\boldsymbol{v}_L), q_R(\boldsymbol{v}_R))$, belongs to NP. This follows by combining the marking equation with a Presburger formula [Verma et al. 2005] for reachability in communication-free Petri nets [Hirshfeld 1993; Esparza 1997]. The latter allows to express the existence of an appropriate tree.

4.3. Colored tokens

Another extension of Petri nets consists in coloring tokens from a countable domain $\mathbb{D}$, such as $\{$●, ■, ▲, $\ldots\}$, instead of the usual singleton $\{$●$\}$ (*e.g.* [Jensen 1996]). A *colored marking* $\boldsymbol{v}$ associates to each place $p \in P$ a finitely supported vector $\boldsymbol{v}(p) \in \mathbb{N}^{\mathbb{D}}$, instead of a number from $\mathbb{N}$. Such colors can represent, *e.g.*, process identities. Reachability for this model is undecidable even for limited update functions, *e.g.* if $\mathbb{D}$ is linearly ordered, tokens can be compared w.r.t. this order, and fresh colors can be spawned [Lazić et al. 2008]. However, it remains unknown whether (un)decidability holds for so-called *unordered data Petri nets*. In this model, arcs are labelled by weighted variables; and a transition is fired by first assigning distinct colors to variables, and then consuming/producing the corresponding number of tokens for each color (*e.g.*, see Figure 6).

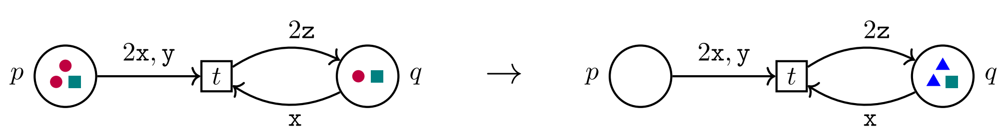

Fig. 6. An unordered data Petri net over $\mathbb{D} = \{$●, ■, ▲, $\ldots\}$. Firing t with valuation x := ●, y := ■ and z := ▲ leads from colored marking $[p\colon$ [●: 2, ■: 1]$, q\colon$ [●: 1, ■: 1]] to colored marking $[q\colon$ [▲: 2, ■: 1]] (left to right).

[6]Filip Mazowiecki and Philip Offtermatt.

Note that it can help to see a colored marking $\boldsymbol{v}$ as a collection of standard markings $\{\boldsymbol{v}|_\star \mid \star \in \mathbb{D}\}$, *e.g.* $\boldsymbol{v}|_\bullet = [p\colon 2, q\colon 1]$, $\boldsymbol{v}|_\blacksquare = [p\colon 1, q\colon 1]$ and $\boldsymbol{v}|_\blacktriangle = \mathbf{0}$ on the left of Figure 6.

Lately, [Hofman et al. 2017] have shown that pseudo-reachability for unordered data Petri nets, the relaxation where colored markings have the form $\boldsymbol{v}\colon P \to \mathbb{Z}^\mathbb{D}$ rather than $\boldsymbol{v}\colon P \to \mathbb{N}^\mathbb{D}$, is NP-complete. Let us summarize their approach.

A *data vector* is a mapping $\boldsymbol{x}\colon \mathbb{D} \to \mathbb{Z}^P$ whose *support* $[\![\boldsymbol{x}]\!] \coloneqq \{\star \in \mathbb{D} \mid \boldsymbol{x}(\star) \neq 0\}$ is finite. Intuitively, such a vector describes the effect $\boldsymbol{\Delta}(t)$ of a transition t. For example, for Figure 6, we could write:

$$\boldsymbol{\Delta}(t) = [\bullet\colon [p\colon -2, q\colon -1], \blacksquare\colon [p\colon -1], \blacktriangle\colon [q\colon 2]]\,. \tag{1}$$

However, the choice of colors in (1) is arbitrary as variables $\{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ could be instantiated otherwise. Hence, $\boldsymbol{\Delta}(t)$ should be seen as a *representative* of the infinite equivalence class obtained by applying any color permutation to (1).

With this in mind, we say that data vector $\boldsymbol{y}$ is a *permutation sum* of a finite set of data vectors V if there exist sequences of data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in V$ and permutations $\pi_1, \pi_2, \ldots, \pi_n\colon \mathbb{D} \to \mathbb{D}$ such that:

$$\boldsymbol{y} = \sum_{1 \leq i \leq n} \boldsymbol{x}_i \circ \pi_i.$$

The marking equation generalizes to unordered data Petri nets as follows:

$$\boldsymbol{u} \xrightarrow{*} \boldsymbol{v} \iff \exists \text{ a permutation sum } \boldsymbol{y} \text{ of } \underbrace{\{\boldsymbol{\Delta}(t) \mid t \in T\}}_{V} \text{ s.t. } \bigwedge_{\star \in [\![\boldsymbol{u}]\!] \cup [\![\boldsymbol{v}]\!] \cup [\![\boldsymbol{y}]\!]} \boldsymbol{v}|_\star - \boldsymbol{u}|_\star = \boldsymbol{y}(\star). \tag{2}$$

In essence, [Hofman et al. 2017] show that the right-hand side of (2) can be written as an integer linear program, by proving that:

(a) there is an integer linear program φ s.t. $\varphi(\boldsymbol{y})$ holds iff $\boldsymbol{y}$ is a permutation sum of V;
(b) if (2) holds, then it does for some $\boldsymbol{y}$ such that $[\![\boldsymbol{y}]\!]$ is of polynomial size in $\sum_{\boldsymbol{x} \in V} |[\![\boldsymbol{x}]\!]|$.

To prove the above, the authors introduce histograms as combinatorial objects that characterize permutation sums. More precisely, a *histogram* of degree k is a matrix $\mathbf{H}\colon \mathbb{D} \times \mathbb{D} \to \mathbb{N}$ such that each row sums to at most k, each column sums to k, and $\mathbf{H}$ has finitely many nonzero columns, *i.e.* $[\![\mathbf{H}]\!]_{\mathrm{col}} \coloneqq \{\star \in \mathbb{D} \mid \mathbf{H}[-, \star] \neq \mathbf{0}\}$ is finite, and consequently $[\![\mathbf{H}]\!]_{\mathrm{row}} \coloneqq \{\star \in \mathbb{D} \mid \mathbf{H}[\star, -] \neq \mathbf{0}\}$ is also finite.

The crux of [Hofman et al. 2017] consists in proving the following:

THEOREM 4.1. *Data vector $\boldsymbol{y}$ is a permutation sum of V iff there exist histograms $\{\mathbf{H}_{\boldsymbol{x}} \mid \boldsymbol{x} \in V, [\![\mathbf{H}_{\boldsymbol{x}}]\!]_{\mathrm{col}} = [\![\boldsymbol{x}]\!]\}$ s.t. $\boldsymbol{y} = \sum_{\boldsymbol{x} \in V} \mathbf{H}_{\boldsymbol{x}} \cdot \boldsymbol{x}$ and $|[\![\mathbf{H}_{\boldsymbol{x}}]\!]_{\mathrm{row}}| \leq 2 \cdot (|[\![\boldsymbol{y}]\!]| + \sum_{\boldsymbol{z} \in V} |[\![\boldsymbol{z}]\!]|)$.*

Note that the NP-complete result extends to pseudo-reachability with *control-states*. Moreover, by exploiting some of these ideas, [Gupta et al. 2019] proved the P-completeness of *continuous* reachability for unordered data Petri nets. More precisely, they:

(a) show that the number of colors for the marking equation is polynomially bounded;
(b) establish a characterization of $\xrightarrow{*}\!\!\!\rightarrow$ reminiscent of Theorem 3.1;
(c) express (b) in the logic $\mathcal{L}$ of [Blondin and Haase 2017] presented in Section 3.3.

Interestingly, pseudo-reachability for the *ordered* data variant of [Lazić et al. 2008] turns out to be much harder: it is equivalent to Petri net reachability.[7] On the other hand, continuous pseudo-reachability for ordered data Petri nets is solvable in polynomial time [Hofman and Lasota 2018] just as in the unordered setting.

[7] Up to an exponential blow-up, but recall that Petri net reachability is TOWER-hard.

ACKNOWLEDGMENTS

I thank Michaël Cadilhac and Christoph Haase for their helpful comments on earlier versions of this column.

REFERENCES

Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. 2000. Algorithmic Analysis of Programs with Well Quasi-ordered Domains. *Information and Computation* 160, 1-2 (2000), 109–127. DOI:http://dx.doi.org/10.1006/inco.1999.2843

David Applegate, William J. Cook, Sanjeeb Dash, and Daniel G. Espinoza. 2007. Exact solutions to linear programming problems. *Operations Research Letters* 35, 6 (2007), 693–699. DOI:http://dx.doi.org/10.1016/j.orl.2006.12.010

Toshiro Araki and Tadao Kasami. 1976. Some Decision Problems Related to the Reachability Problem for Petri Nets. *Theoretical Computer Science* 3, 1 (1976), 85–104. DOI:http://dx.doi.org/10.1016/0304-3975(76)90067-0

Konstantinos Athanasiou, Peizun Liu, and Thomas Wahl. 2016. Unbounded-Thread Program Verification using Thread-State Equations. In *Proc.* 8^{th} *International Joint Conference on Automated Reasoning (IJCAR)*, Vol. 9706. 516–531. DOI:http://dx.doi.org/10.1007/978-3-319-40229-1_35

Mohamed Faouzi Atig, Ahmed Bouajjani, and Shaz Qadeer. 2011. Context-Bounded Analysis For Concurrent Programs With Dynamic Creation of Threads. *Logical Methods in Computer Science (LMCS)* 7, 4 (2011). DOI:http://dx.doi.org/10.2168/LMCS-7(4:4)2011

Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovi'c, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. CVC4. In *Proc.* 23^{rd} *International Conference on Computer Aided Verification (CAV)*. 171–177. DOI:http://dx.doi.org/10.1007/978-3-642-22110-1_14

Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. 2015. Reachability in Two-Dimensional Vector Addition Systems with States Is PSPACE-Complete. In *Proc.* 30^{th} *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 32–43. DOI:http://dx.doi.org/10.1109/LICS.2015.14

Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. 2017. The Logical View on Continuous Petri Nets. *ACM Transactions on Computational Logic (TOCL)* 18, 3 (2017), 24:1–24:28. DOI:http://dx.doi.org/10.1145/3105908

Michael Blondin and Christoph Haase. 2017. Logics for continuous reachability in Petri nets and vector addition systems with states. In *Proc.* 32^{nd} *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–12. DOI:http://dx.doi.org/10.1109/LICS.2017.8005068

Michael Blondin, Christoph Haase, and Filip Mazowiecki. 2018. Affine Extensions of Integer Vector Addition Systems with States. In *Proc.* 29^{th} *International Conference on Concurrency Theory (CONCUR)*. 14:1–14:17. DOI:http://dx.doi.org/10.4230/LIPIcs.CONCUR.2018.14

Michael Blondin and Mikhail Raskin. 2020. The Complexity of Reachability in Affine Vector Addition Systems with States. In *Proc.* 35^{th} *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. DOI:http://dx.doi.org/10.1145/3373718.3394741

Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2009. Two-variable logic on data trees and XML reasoning. *Journal of the ACM* 56, 3 (2009), 13:1–13:48. DOI:http://dx.doi.org/10.1145/1516512.1516515

Ahmed Bouajjani and Michael Emmi. 2013. Analysis of Recursively Parallel Programs. *ACM Transactions on Programming Languages and Systems* 35, 3 (2013), 10:1–10:49. DOI:http://dx.doi.org/10.1145/2518188

Georgina Bumpus, Christoph Haase, Stefan Kiefer, Paul-Ioan Stoienescu, and Jonathan Tanner. 2020. On the Size of Finite Rational Matrix Semigroups. In *Proc.* 47^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*. To appear.

Dmitry Chistikov, Christoph Haase, and Simon Halfon. 2018. Context-free commutative grammars with integer counters and resets. *Theoretical Computer Science* 735 (2018), 147–161. DOI:http://dx.doi.org/10.1016/j.tcs.2016.06.017

Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. 2019. The reachability problem for Petri nets is not elementary. In *Proc.* 51^{st} *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 24–33. DOI:http://dx.doi.org/10.1145/3313276.3316369

René David and Hassane Alla. 1987. Continuous Petri nets. In *Proc. 8th European Workshop on Application and Theory of Petri nets*, Vol. 340. 275–294.

René David and Hassane Alla. 2010. *Discrete, Continuous, and Hybrid Petri nets* (2^{nd} ed.). Springer.

Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. 2004. Vector Addition Tree Automata. In *Proc.* 19^{th} *IEEE Symposium on Logic in Computer Science (LICS)*. 64–73. DOI:http://dx.doi.org/10.1109/LICS.2004.1319601

Leonardo Mendonça de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Proc.* 14^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 337–340. DOI:http://dx.doi.org/10.1007/978-3-540-78800-3_24

Giorgio Delzanno, Jean-François Raskin, and Laurent Van Begin. 2002. Towards the Automated Verification of Multithreaded Java Programs. In *Proc.* 8^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 173–187. DOI:http://dx.doi.org/10.1007/3-540-46002-0_13

Alex Dixon and Ranko Lazić. 2020. KReach: A Tool for Reachability in Petri Nets. In *Proc.* 26^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 405–412. DOI:http://dx.doi.org/10.1007/978-3-030-45190-5_22

Frank Drewes and Jérôme Leroux. 2015. Structurally Cyclic Petri Nets. *Logical Methods in Computer Science (LMCS)* 11, 4 (2015). DOI:http://dx.doi.org/10.2168/LMCS-11(4:15)2015

Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. 1998. Reset Nets Between Decidability and Undecidability. In *Proc.* 25^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*. 103–115. DOI:http://dx.doi.org/10.1007/BFb0055044

Javier Esparza. 1997. Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes. *Fundamenta Informaticae* 31, 1 (1997), 13–25. DOI:http://dx.doi.org/10.3233/FI-1997-3112

Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. 2017. Verification of population protocols. *Acta Informatica* 54, 2 (2017), 191–215. DOI:http://dx.doi.org/10.1007/s00236-016-0272-3

Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Nikšić. 2014. An SMT-Based Approach to Coverability Analysis. In *Proc.* 26^{th} *International Conference on Computer Aided Verification (CAV)*, Vol. 8559. 603–619. DOI:http://dx.doi.org/10.1007/978-3-319-08867-9_40

Estíbaliz Fraca and Serge Haddad. 2015. Complexity Analysis of Continuous Petri Nets. *Fundamenta Informaticae* 137, 1 (2015), 1–28. DOI:http://dx.doi.org/10.3233/FI-2015-1168

Thomas Geffroy, Jérôme Leroux, and Grégoire Sutre. 2018. Occam's Razor applied to the Petri net coverability problem. *Theoretical Computer Science* 750 (2018), 38–52. DOI:http://dx.doi.org/10.1016/j.tcs.2018.04.014

Steven M. German and A. Prasad Sistla. 1992. Reasoning about Systems with Many Processes. *Journal of the ACM* 39, 3 (1992), 675–735. DOI:http://dx.doi.org/10.1145/146637.146681

Sheila A. Greibach. 1978. Remarks on Blind and Partially Blind One-Way Multicounter Machines. *Theoretical Computer Science* 7 (1978), 311–324. DOI:http://dx.doi.org/10.1016/0304-3975(78)90020-8

Utkarsh Gupta, Preey Shah, S. Akshay, and Piotr Hofman. 2019. Continuous Reachability for Unordered Data Petri Nets is in PTime. In *Proc.* 22^{nd} *International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, Vol. 11425. 260–276. DOI:http://dx.doi.org/10.1007/978-3-030-17127-8_15

Christoph Haase and Simon Halfon. 2014. Integer Vector Addition Systems with States. In *Proc.* 8^{th} *International Workshop on Reachability Problems (RP)*. 112–124. DOI:http://dx.doi.org/10.1007/978-3-319-11439-2_9

Monika Heiner, David R. Gilbert, and Robin Donaldson. 2008. Petri Nets for Systems and Synthetic Biology. In *Formal Methods for Computational Systems Biology*. 215–264. DOI:http://dx.doi.org/10.1007/978-3-540-68894-5_7

Yoram Hirshfeld. 1993. Petri Nets and the Equivalence Problem. In *Proc.* 7^{th} *Workshop on Computer Science Logic (CSL)*, Vol. 832. 165–174. DOI:http://dx.doi.org/10.1007/BFb0049331

Piotr Hofman and Slawomir Lasota. 2018. Linear Equations with Ordered Data. In *Proc.* 29^{th} *International Conference on Concurrency Theory (CONCUR)*, Vol. 118. 24:1–24:17. DOI:http://dx.doi.org/10.4230/LIPIcs.CONCUR.2018.24

Piotr Hofman, Jérôme Leroux, and Patrick Totzke. 2017. Linear combinations of unordered data vectors. In *Proc.* 32^{nd} *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–11. DOI:http://dx.doi.org/10.1109/LICS.2017.8005065

John Hopcroft and Jean-Jacques Pansiot. 1979. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science* 8, 2 (1979), 135–159. DOI:http://dx.doi.org/10.1016/0304-3975(79)90041-0

Kurt Jensen. 1996. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use – Volume 1* (2^{nd} ed.). Springer. DOI:http://dx.doi.org/10.1007/978-3-662-03241-1

Alexander Kaiser, Daniel Kroening, and Thomas Wahl. 2014. A Widening Approach to Multithreaded Program Verification. *ACM Trans. on Prog. Languages and Systems* 36, 4 (2014), 14:1–14:29. DOI:http://dx.doi.org/10.1145/2629608

S. Rao Kosaraju. 1982. Decidability of Reachability in Vector Addition Systems (Preliminary Version). In *Proc.* 14^{th} *Symposium on Theory of Computing (STOC)*. 267–281. DOI:http://dx.doi.org/10.1145/800070.802201

Jean-Luc Lambert. 1992. A Structure to Decide Reachability in Petri Nets. *Theoretical Computer Science* 99, 1 (1992), 79–104. DOI:http://dx.doi.org/10.1016/0304-3975(92)90173-D

Sławomir Lasota. 2018. VASS reachability in three steps. *CoRR* abs/1812.11966 (2018).

Ranko Lazić, Thomas Christopher Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. 2008. Nets with Tokens which Carry Data. *Fundamenta Informaticae* 88, 3 (2008), 251–274.

Ranko Lazić and Sylvain Schmitz. 2015. Nonelementary Complexities for Branching VASS, MELL, and Extensions. *ACM Transactions on Computational Logic (TOCL)* 16, 3 (2015), 20:1–20:30. DOI:http://dx.doi.org/10.1145/2733375

Jérôme Leroux. 2012. Vector Addition Systems Reachability Problem (A Simpler Solution). In *Turing-100 – The Alan Turing Centenary*. 214–228.

Jérôme Leroux and Sylvain Schmitz. 2019. Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension. In *Proc.* 34^{th} *Symposium on Logic in Computer Science (LICS)*.

Jérôme Leroux and Grégoire Sutre. 2004. On Flatness for 2-Dimensional Vector Addition Systems with States. In *Proc.* 15^{th} *International Conference on Concurrency Theory (CONCUR)*, Vol. 3170. 402–416. DOI:http://dx.doi.org/10.1007/978-3-540-28644-8_26

Richard J. Lipton. 1976. *The Reachability Problem Requires Exponential Space*. Technical Report 63. Department of Computer Science, Yale University.

Rupak Majumdar and Zilong Wang. 2013. Expand, Enlarge, and Check for Branching Vector Addition Systems. In *Proc.* 24^{th} *International Conference on Concurrency Theory (CONCUR)*, Vol. 8052. 152–166. DOI:http://dx.doi.org/10.1007/978-3-642-40184-8_12

Ernst W. Mayr. 1981. An Algorithm for the General Petri Net Reachability Problem. In *Proc.* 13^{th} *Symposium on Theory of Computing (STOC)*. 238–246. DOI:http://dx.doi.org/10.1145/800076.802477

Jakub Michaliszyn, Jan Otop, and Piotr Wieczorek. 2017. Querying Best Paths in Graph Databases. In *Proc.* 37^{th} *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Vol. 93. 43:1–43:15. DOI:http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2017.43

Julien Reichert. 2015. *Reachability games with counters: decidability and algorithms*. Ph.D. Dissertation. École normale supérieure de Cachan, France.

Sylvain Schmitz. 2010. On the Computational Complexity of Dominance Links in Grammatical Formalisms. In *Proc.* 48^{th} *Annual Meeting of the Association for Computational Linguistics (ACL)*. 514–524.

Jake Silverman and Zachary Kincaid. 2019. Loop Summarization with Rational Vector Addition Systems. In *Proc.* 31^{st} *International Conference on Computer Aided Verification (CAV)*. 97–115. DOI:http://dx.doi.org/10.1007/978-3-030-25543-5_7

Rüdiger Valk. 1978. Self-Modifying Nets, a Natural Extension of Petri Nets. In *Proc. Fifth Colloquium on Automata, Languages and Programming (ICALP)*. 464–476. DOI:http://dx.doi.org/10.1007/3-540-08860-1_35

Wil van der Aalst. 1998. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers* 8, 1 (1998), 21–66. DOI:http://dx.doi.org/10.1142/S0218126698000043

Kumar Neeraj Verma and Jean Goubault-Larrecq. 2005. Karp-Miller Trees for a Branching Extension of VASS. *Discrete Mathematics & Theoretical Computer Science* 7 (2005), 217–230.

Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. 2005. On the Complexity of Equational Horn Clauses. In *Proc.* 20^{th} *International Conference on Automated Deduction on Automated Deduction*. 337–352. DOI:http://dx.doi.org/10.1007/11532231_25

Moe Thandar Wynn, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. 2009. Synchronization and Cancelation in Workflows Based on Reset Nets. *International Journal of Cooperative Information Systems* 18, 1 (2009), 63–114. DOI:http://dx.doi.org/10.1142/S0218843009002002