On the Complexity of Proving Polyhedral Reductions

Nicolas Amat, Silvano Dal Zilio, Didier Le Botlan

Published in Fundamenta Informaticae 192(3-4) (2024) – DOI 10.3233/FI-242197
and as *arXiv:2306.01466v4*

On the Complexity of Proving Polyhedral Reductions

Nicolas Amat*
IMDEA Software Institute
Madrid, Spain
nicolas.amat@imdea.org

Silvano Dal Zilio, Didier Le Botlan
LAAS-CNRS
Université de Toulouse, INSA, CNRS
Toulouse, France

Abstract. We propose an automated procedure to prove polyhedral abstractions (also known as polyhedral reductions) for Petri nets. Polyhedral abstraction is a new type of state space equivalence, between Petri nets, based on the use of linear integer constraints between the marking of places. In addition to defining an automated proof method, this paper aims to better characterize polyhedral reductions, and to give an overview of their application to reachability problems.

Our approach relies on encoding the equivalence problem into a set of SMT formulas whose satisfaction implies that the equivalence holds. The difficulty, in this context, arises from the fact that we need to handle infinite-state systems. For completeness, we exploit a connection with a class of Petri nets, called flat nets, that have Presburger-definable reachability sets. We have implemented our procedure, and we illustrate its use on several examples.

Keywords: Automated reasoning, Abstraction techniques, Reachability problems, Petri nets

1. Introduction

Our work is related with a new abstraction technique for Petri nets [1, 2] based on a combination of structural reductions [3, 4] with the use of linear constraints between the marking of places. The idea

*Address for correspondence: IMDEA Software Institute Campus Montegancedo s/n, 28223 Pozuelo de Alarcon, Madrid, Spain

is to compute reductions of the form (N, E, N'), where: N is an initial net (that we want to analyze); N' is a residual net (hopefully much simpler than N); and E is a Presburger predicate. The idea is to preserve enough information in E to rebuild the reachable markings of N, knowing only the ones of N'. We refined this concept into a new abstraction, called *polyhedral abstraction* [5, 6], in reference to "polyhedral models" used in program optimization and static analysis [7, 8]. Indeed, like in these works, we propose an algebraic representation of state spaces using solutions to linear constraints. We use the term *abstraction* to refer to the fact that we exploit an equivalence relation that provides a mapping between an initial and a target model. We shall also often use the name *polyhedral reduction* for the same concept, interchangeably, due to the fact that many of the abstraction rules we use in practice derives from (Petri net) structural reductions.

We implemented our approach into two independent symbolic model-checkers developed by our team: `Tedd`, a tool based on Hierarchical Set Decision Diagrams (SDD) [9], part of the `Tina` toolbox [10]; and `SMPT` [11, 12], an SMT-based model-checker focused on reachability problems [13]. Both tools demonstrated the effectiveness of polyhedral reductions by achieving good rankings in both the StateSpace and Reachability examinations of the Model Checking Contest [14], an international competition for model-checking tools.

Our approach has several positive features. In particular, it does not impose restrictions on the syntax of nets, such as constraints on the weights of arcs, and it can be transparently applied to unbounded nets. In practice, we can often reduce a Petri net N with n places (from a high dimensional space) into a residual net N' with far fewer places, say n' (in a lower-dimensional space). More formally, with our approach, we can represent the state space of N as the "inverse image", by the Presburger predicate E, of the state space of N' (a subset of vectors in dimension n'), which can result in a very compact representation of the reachability set. This problem shares some similarities with the question of whether we can precisely characterize the reachability set of a net using a formula in Presburger arithmetic. A connection we will further develop. An important distinction is that we use "Presburger relations" to relate the reachability set of two nets, as an equivalence, rather than to abstract a single state space. One of the goals of our work is to give decidability results about this equivalence, and to find ways to automatically check when an equivalence judgment is true.

We define this notion of equivalence using a new relation, $N \equiv_E N'$, called *polyhedral abstraction equivalence* (or just *polyhedral equivalence* for short). We should also often use the term *E-abstraction equivalence* to emphasize the importance of the linear predicate E. This equivalence plays a central role in many of our results, as well as it provides the basis to formally define polyhedral reductions.

We prove that deciding the correctness of our original notion of equivalence, see Sect. 2.2, is undecidable (Theorem 2.2). This decidability result is not surprising since most equivalence problems on Petri nets are undecidable [15, 16]. Indeed, polyhedral equivalence is by essence related to the *marking equivalence* problem, which amounts to deciding if two Petri nets with the same set of places have the same reachable markings; a problem proved undecidable by Hack [17]. Also, polyhedral equivalence (such as marking equivalence) entails trace equivalence, another well-known undecidable equivalence problem when we consider general Petri nets [18, 17].

Although this may appear contradictory, we prove that the equivalence problem becomes decidable when we consider a slightly different, and in some sense more general, equivalence relation between

parametric Petri nets. In this context, we use the term *parametric* to stress the fact that we manipulate semilinear sets of markings, meaning sets that can be defined using a Presburger arithmetic formula C. In particular, we reason about parametric nets (N, C), instead of marked nets (N, m_0), with the intended meaning that all markings satisfying C are potential initial markings of N. We also define an extended notion of polyhedral equivalence between parametric nets, denoted $(N_1, C_1) \approxeq_E (N_2, C_2)$, whereas our original definition [19, 5] was between marked nets only (see Definition 2.1).

We show that given a valid equivalence statement $(N_1, C_1) \approxeq_E (N_2, C_2)$, it is possible to derive a Presburger formula, in a constructive way, whose satisfaction implies that the equivalence holds. We implemented this procedure on top of an SMT-solver for Linear Integer Arithmetic (LIA) and show that our approach is applicable in practice (Sect. 10). Even if we prove that this problem is decidable (see Theorem 6.1), our implementation is only a semi-decision procedure since we rely on the external tool `FAST`, which may not terminate if the equivalence does not hold. If anything, it makes the fact that we may translate our problem into Presburger arithmetic quite remarkable.

Description of our approach

Our approach can be summarized as follows. We start from an initial net (N_1, C_1) and derive a polyhedral equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$ by applying a set of *abstraction laws* in an iterative and compositional way. Finally, we solve a reachability problem about N_1 by transforming it into a reachability problem about net N_2, which should hopefully be easier to check. A large number of the laws we implement in our tools derive from structural reduction rules [4], or are based on the elimination of redundant places and transitions, with the goal to obtain a "reduced" net N_2 that is smaller than N_1.

We also implement several other kinds of abstraction rules—often subtler to use and harder to prove correct—which explains why we want machine checkable proofs of equivalence. For instance, some of our rules are based on the identification of Petri nets subclasses in which the set of reachable markings equals the set of potentially reachable ones, a property we call the PR-R equality in [20, 21]. We use this kind of rules in the example of the "SwimmingPool" model of Fig. 10, a classical example of Petri net often used in case studies (see e.g. [22]).

We give an example of a basic abstraction law in Fig. 1, with an instance of rule (CONCAT) that allows us to fuse two places connected by a direct, silent transition. We give another example with (MAGIC), in Fig. 2, which illustrates a more complex agglomeration rule, and refer to other examples in Sect. 10.

The parametric net (N_1, C_1) (left of Fig. 1) has a condition which entails that place y_2 should be empty initially ($y_2 = 0$), whereas net (N_2, C_2) has a trivial constraint, which can be interpreted as simply $x \geqslant 0$. We can show (see Sect. 3) that nets N_1 and N_2 are E-equivalent, which amounts to prove that any marking $(y_1 : k_1,\ y_2 : k_2)$ of N_1, reachable by firing a transition sequence σ, can be associated with the marking $(x : k_1 + k_2)$ of N_2, also reachable by the same firing sequence. Actually, we prove that this equivalence is sound when no transition can input a token directly into place y_2 of N_1. This means that the rule is correct in the absence of the "dashed" transition (with label d), but that our procedure should flag the rule as unsound when transition d is present.

The results presented in this paper provide an automated technique for proving the correctness of

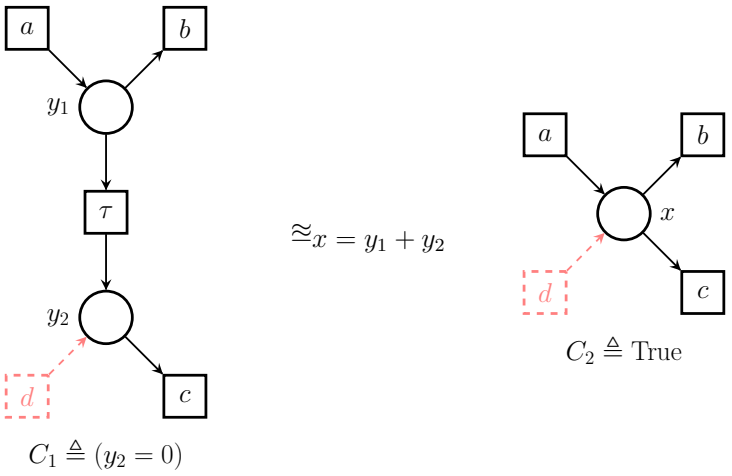


Figure 1: Equivalence rule (CONCAT), $(N_1, C_1) \cong_E (N_2, C_2)$, between nets N_1 (left) and N_2 (right), for the relation $E \triangleq (x = y_1 + y_2)$.

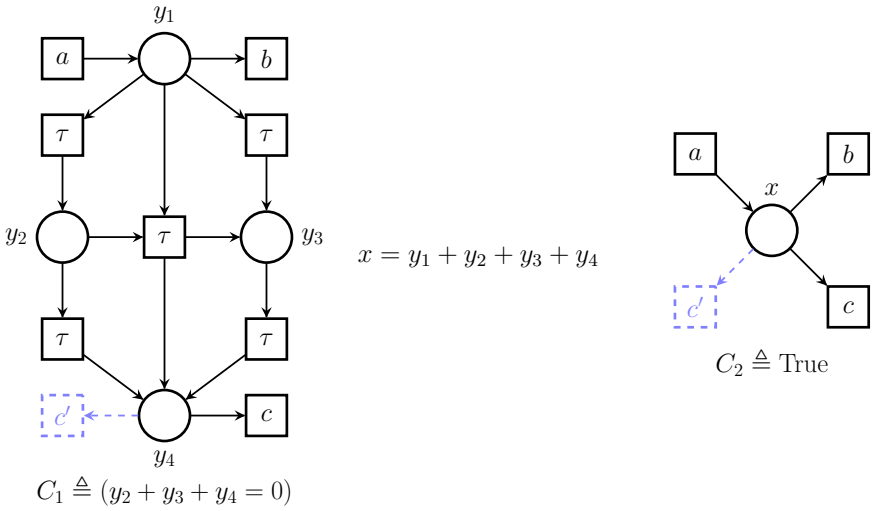


Figure 2: Equivalence rule (MAGIC).

polyhedral abstraction laws. This helps us gain more confidence on the correctness of our tools and is also useful if we want to add new abstraction rules. Indeed, up until now, all our rules where proven using "manual theorem proving", which can be tedious and error-prone.

Incidentally, the theory we developed for this paper also helped us gain a better understanding of the constraints necessary when designing new abstraction laws. A critical part of our approach relies on the ability, given a Presburger predicate C, to encode the set of markings reachable from C by firing only silent transitions, that we denote $\tau_C^\star$ in the following. Our approach draws a connection with previous works [23, 24, 25] that study the class of Petri nets that have Presburger-definable reachability sets; also called *flat nets*. We should also make use of a tool implemented by the same

authors, called FAST, which provides a method for representing the reachable set of flat nets. Basically, we gain the insight that polyhedral reductions provide a way to abstract away (or collapse) the subparts of a net that are flat. Note that our approach may work even though the reachability set of the whole net is not semilinear, since only the part that is abstracted must be flat. We also prove that when $(N_1, C_1) \approxeq_E (N_2, C_2)$ then necessarily the sets $\tau^\star_{C_1}$ and $\tau^\star_{C_2}$ are semilinear.

Outline and contributions

The paper is organized as follows. We define our central notion of *parametric polyhedral abstraction* in Sect. 3 and prove several of its properties in Sect. 7. In particular, we prove that polyhedral abstraction is a congruence, and that it is preserved when "duplicating labeled transitions". These properties mean that every abstraction law we prove can be safely applied in every context, and that each law can be used as a "rule schema". Our definition relies on a former notion of polyhedral equivalence, that we recall in Sect. 2, together with a quick overview of our notations. We describe our proof procedure in Sect. 4, which is defined as the construction of a set of four *core requirements*, each expressed as separate quantified LIA formulas. A key ingredient in this translation is to build a predicate, $\tau^\star_C$, which encodes the markings reachable by firing only the silent transitions of a net. We defer the definition of this predicate until Sect. 5, where we show how it can be obtained using the output of the FAST tool. From this procedure, we prove that our problem is decidable in Sect. 6, and we extend our automated procedure in Sect. 8 to the check of state space partition that is prerequisite for model counting. We conclude by presenting the results obtained with a new tool implementing our approach, called Reductron, on some concrete examples. First, in Sect. 9 by showing how our tool can be used to "debug" incorrect reduction rules, and in Sect. 10 by providing quantitative information on its performance for our set of reduction rules.

Many results and definitions have already been presented in a shorter version of the paper [26]. This extended version contains several additions. First, we have added the full proofs of all the results given in our work and added a new fundamental result, namely the decidability of checking parametric equivalence (Sect. 6). We also give a more precise proof for the undecidability of checking "regular" polyhedral equivalence. This paper also contains new theoretical results, such as an automatic method for checking when the equivalence defines a partition of the state space (Sect. 8). Finally, we added a new section with experimental results about the performance of our tool (Sect. 10) and its use for debugging problematic reduction rules (Sect. 9).

2. Petri nets and polyhedral abstraction

In this section, we briefly introduce Petri nets and our concept of polyhedral reduction. While we tried to make the presentation as self-contained as possible, we assume that the reader has some familiarity with basic Petri net theory.

2.1. Petri nets

A Petri net is a tuple $(P, T, \mathrm{Pre}, \mathrm{Post})$, where $P \triangleq \{p_1, \ldots, p_n\}$ is a finite set of places, $T \triangleq \{t_1, \ldots, t_k\}$ is a finite set of transitions (disjoint from P), and $\mathrm{Pre} : T \to (P \to \mathbb{N})$ and $\mathrm{Post} : T \to (P \to \mathbb{N})$ are the pre- and post-condition functions (also known as the flow functions of the net). A state of a net, also called a marking, is a mapping $m : P \to \mathbb{N}$ (also denoted $\mathbb{N}^P$) that assigns a number of tokens, $m(p)$, to each place p in P. A marked net (N, m_0) is a pair consisting of a net, N, and an initial marking, m_0. In the following, we will often consider that each transition is labeled with a symbol from an alphabet Σ. In this case, we assume that a net is associated with a labeling function $l : T \to \Sigma \cup \{\tau\}$, where τ is a special symbol for the silent action. Every net has a default labeling function, l_N, such that $\Sigma = T$ and $l_N(t) \triangleq t$ for every transition $t \in T$.

A transition $t \in T$ is enabled at a marking $m \in \mathbb{N}^P$ if $m(p) \geqslant \mathrm{Pre}(t, p)$ for all places $p \in P$, which we also write $m \geqslant \mathrm{Pre}(t)$, where $\geqslant$ represents component-wise comparison of the markings. A marking $m' \in \mathbb{N}^P$ is reachable from a marking $m \in \mathbb{N}^P$ by firing transition t, denoted $(N, m) \xrightarrow{t} (N, m')$ or simply $m \xrightarrow{t} m'$ when N is obvious from the context, if: (1) transition t is enabled at m, and (2) $m' = m - \mathrm{Pre}(t) + \mathrm{Post}(t)$. A firing sequence $\varrho \triangleq t_1, \ldots, t_n \in T^*$ can be fired from m, denoted $(N, m) \xrightarrow{\varrho} (N, m')$ or simply $m \xrightarrow{\varrho} m'$, if there exist markings $m_0, \ldots, m_n$ such that $m = m_0$, $m' = m_n$, and $m_i \xrightarrow{t_{i+1}} m_{i+1}$ for all $i < n$. We denote $R(N, m_0)$ the set of markings reachable from m_0 in N.

Observable sequences

We can lift any labeling function $l : T \to \Sigma \cup \{\tau\}$ to a mapping of sequences from T^* to Σ^*. Specifically, we define inductively $l(\varrho.t) \triangleq l(\varrho)$ if $l(t) = \tau$ and $l(\varrho.t) \triangleq l(\varrho).l(t)$ otherwise, where the dot operator (.) stands for concatenation, and $l(\epsilon) \triangleq \epsilon$, where ϵ is the empty sequence, verifying $\epsilon.\sigma = \sigma.\epsilon = \sigma$ for any $\sigma \in \Sigma^*$. Given a sequence of labels $\sigma \in \Sigma^*$, we write $(N, m) \overset{\sigma}{\Rightarrow} (N, m')$ if there exists a firing sequence $\varrho \in T^*$ such that $(N, m) \overset{\varrho}{\Rightarrow} (N, m')$ and $\sigma = l(\varrho)$. In this case, σ is referred to as an *observable sequence* of the marked net (N, m). In some cases, we have to consider firing sequences that must not finish with τ transitions. Hence, we define a relation $(N, m) \overset{\sigma\rangle}{\Rightarrow} (N, m')$, written simply $m \overset{\sigma\rangle}{\Rightarrow} m'$, as follows:

- $(N, m) \overset{\epsilon\rangle}{\Rightarrow} (N, m)$ holds for all marking m.
- $(N, m) \overset{\sigma.a\rangle}{\Longrightarrow} (N, m')$ holds for any markings m, m' and $a, \sigma \in \Sigma \times \Sigma^*$, if there exists a marking m'' and a transition t such that $l(t) = a$ and $(N, m) \overset{\sigma}{\Rightarrow} (N, m'') \xrightarrow{t} (N, m')$.

It is immediate that $m \overset{\sigma\rangle}{\Rightarrow} m'$ implies $m \overset{\sigma}{\Rightarrow} m'$. Note the difference between $m \overset{\epsilon}{\Rightarrow} m'$, which stands for any sequence of τ transitions, and $m \overset{\epsilon\rangle}{\Rightarrow} m'$, which implies $m = m'$ (the sequence is empty).

We use the standard graphical notation for nets, where places are depicted as circles and transitions as squares such as the nets displayed in Fig. 1.

2.2. Polyhedral abstraction

We define an equivalence relation that can be used to describe a linear dependence between the markings of two different nets, N_1 and N_2. Assume V is a set of places $p_1, \ldots, p_n$, considered as variables,

and let m be a mapping in $V \to \mathbb{N}$. We define $\underline{m}$ as a linear formula, whose unique model in $\mathbb{N}^V$ is m, defined as $\underline{m} \triangleq \bigwedge\{x = m(x) \mid x \in V\}$. By extension, given a Presburger formula E, we say that m is a (partial) solution of E if the formula $E \wedge \underline{m}$ is satisfiable. Equivalently, we can view $\underline{m}$ as a substitution, where each variable $x \in V$ is substituted by $m(x)$. Indeed, the formula $F\{m\}$ (the substitution $\underline{m}$ applied to F) and $F \wedge \underline{m}$ admit the same models.

Equivalence Between Markings

Given two mappings $m_1 \in \mathbb{N}^{V_1}$ and $m_2 \in \mathbb{N}^{V_2}$, we say that m_1 and m_2 are *compatible* when they have equal values on their shared domain: $m_1(x) = m_2(x)$ for all x in $V_1 \cap V_2$. This is a necessary and sufficient condition for the system $\underline{m_1} \wedge \underline{m_2}$ to be satisfiable. Finally, if V is the set of free variables of $\underline{m_1}$, $\underline{m_2}$, and the free variables of E are included in V, we say that m_1 and m_2 are related up-to E, denoted $m_1 \equiv_E m_2$, when $E \wedge \underline{m_1} \wedge \underline{m_2}$ is satisfiable.

$$m_1 \equiv_E m_2 \quad \Leftrightarrow \quad \exists m \in \mathbb{N}^V \,.\, m \models E \wedge \underline{m_1} \wedge \underline{m_2} \tag{1}$$

Equivalence Between Nets

The previous relation defines an equivalence between markings of two different nets ($\equiv_E \subseteq \mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$) and, by extension, can be used to define an equivalence between nets themselves, that is called *polyhedral equivalence* in [5, 27], where all reachable markings of N_1 are related to reachable markings of N_2 (and conversely), as explained next.

Definition 2.1. (E-Abstraction)
Assume $N_1 \triangleq (P_1, T_1, \mathrm{Pre}_1, \mathrm{Post}_1)$ and $N_2 \triangleq (P_2, T_2, \mathrm{Pre}_2, \mathrm{Post}_2)$ are two Petri nets, and E a Presburger formula whose free variables are included in $P_1 \cup P_2$. We say that the marked net (N_2, m_2) is an E-abstraction of (N_1, m_1), denoted $(N_1, m_1) \sqsubseteq_E (N_2, m_2)$, if and only if:

(A1) the initial markings are related up-to E, meaning $m_1 \equiv_E m_2$;

(A2) for all observable sequences $(N_1, m_1) \stackrel{\sigma}{\Rightarrow} (N_1, m_1')$ in N_1, there is at least one marking m_2' over P_2 such that $m_1' \equiv_E m_2'$, and for all markings m_2' over P_2 such that $m_1' \equiv_E m_2'$ we have $(N_2, m_2) \stackrel{\sigma}{\Rightarrow} (N_2, m_2')$.

We say that (N_1, m_1) is E-equivalent to (N_2, m_2), denoted $(N_1, m_1) \equiv_E (N_2, m_2)$, when we have both $(N_1, m_1) \sqsubseteq_E (N_2, m_2)$ and $(N_2, m_2) \sqsubseteq_E (N_1, m_1)$.

By definition, given an equivalence statement $(N_1, m_1) \equiv_E (N_2, m_2)$, then for every marking m_2' reachable in N_2, the set of markings of N_1 consistent with $E \wedge \underline{m_2'}$ is non-empty (condition (A2)). In practice, this defines a partition of the reachable markings of (N_1, m_1) into a union of "convex sets"—hence the name polyhedral abstraction—each associated to one (at least) reachable marking in N_2.

Although E-abstraction looks like a simulation, it is not, since the pair of reachable markings m_1', m_2' from the definition does not satisfy $(N_1, m_1') \sqsubseteq_E (N_2, m_2')$ in general. This relation $\sqsubseteq_E$ is therefore broader than a simulation, but suffices for our primary goal, that is Petri net reduction. Of course, $\equiv_E$ is not a bisimulation either.

Undecidability of the Equivalence Checking

It is also quite simple to show that checking E-abstraction equivalence is undecidable in general.

Theorem 2.2. (Undecidability of the E-Equivalence Checking)
The problem of checking whether a statement $(N_1, m_1) \equiv_E (N_2, m_2)$ is valid is undecidable.

Proof:
By contradiction, we assume there exists some algorithm, say $\mathcal{A}$, that checks the E-abstraction equivalence problem. More precisely, the input of $\mathcal{A}$ consists in two marked nets (N_1, m_1) and (N_2, m_2), as well as a Presburger formula E with free variables in the places of N_1 and N_2. The output of $\mathcal{A}$ is a Boolean, indicating whether $(N_1, m_1) \equiv_E (N_2, m_2)$ holds or not.

Let us consider another problem: given any pair of marked nets (N_1, m_1) and (N_2, m_2) with the same set of places, and equal initial markings (i.e., $m_1 = m_2$), check the marking equivalence of both nets, that is check if $R(N_1, m_1) = R(N_2, m_2)$ holds. This problem is known to be undecidable*. Yet, we will show that algorithm $\mathcal{A}$ is always able to answer to this problem, hence the contradiction.

Take any pair of marked nets (N_1, m_1) and (N_2, m_2) with the same set of places and $m_1 = m_2$. We equip each net with a labeling function l_1 (resp. l_2) such that $l_1(t) = \tau$ (resp. $l_2(t) = \tau$) for all transition t of N_1 (resp. N_2). Let us show first that: $(N_1, m_1) \sqsubseteq_E (N_2, m_2)$ with the trivial constraint $E \triangleq \text{True}$ is equivalent to $R(N_1, m_1) \subseteq R(N_2, m_2)$.

Condition (A1) trivially holds since $m_1 = m_2$. We now show that condition (A2) is necessary and sufficient for $R(N_1, m_1) \subseteq R(N_2, m_2)$:

- Assume condition (A2) holds and take a marking m'_1 in $R(N_1, m_1)$. We have $m'_1 \equiv_E m'_1$. Then, by condition (A2) we get $m'_1 \in R(N_2, m_2)$, and so $R(N_1, m_1) \subseteq R(N_2, m_2)$.
- Assume $R(N_1, m_1) \subseteq R(N_2, m_2)$ and take a firing sequence $(N_1, m_1) \xrightarrow{\varrho_1} (N_1, m'_1)$. Since all transitions are silent we have $l_1(\varrho_1) = \epsilon$. Both nets share the same sets of places, thus m'_1 satisfies $m'_1 \equiv_E m'_1$ (and no other marking $m'_2 \neq m'_1$ satisfies the condition $m'_1 \equiv_E m'_2$). By assumption, $m'_1 \in R(N_2, m_2)$, meaning $(N_2, m_2) \xrightarrow{\varrho_2} (N_2, m'_1)$ for some firing sequence ϱ_2 such that $l_2(\varrho_2) = \epsilon$, and so, condition (A2) holds.

The statement above is proved. By immediate symmetry, we get that $R(N_1, m_1) = R(N_2, m_2)$ is equivalent to $(N_1, m_1) \equiv_E (N_2, m_2)$. As a consequence, checking the marking equivalence problem is equivalent to checking the E-equivalence problem on (N_1, m_1) and (N_2, m_2), with E the trivial constraint. Since algorithm $\mathcal{A}$ is supposed to answer to the latter, it equivalently answers to the former, which is a contradiction.

* Hack proved the undecidability of the marking equivalence between two subparts of nets N_1, N_2 given a pair of initial markings not necessary equal [17]. However, his proof's construction leads to the same results when initial markings are equal. □

2.3. Basic properties of polyhedral reduction

We proved in [19, 5] that we can use E-equivalence to check the reachable markings of N_1 simply by looking at the reachable markings of N_2. We give a first property that is useful when we

try to find a counter-example to a property by looking at firing sequences with increasing length. Our second property is useful for checking invariants. Both results are at the basis of our model checker SMPT.

Lemma 2.3. (Reachability Checking [19, 5])
Assume $(N_1, m_1) \equiv_E (N_2, m_2)$. Then for all pairs of markings m'_1, m'_2 of N_1, N_2 such that $m'_1 \equiv_E m'_2$ and $m'_2 \in R(N_2, m_2)$ it is the case that $m'_1 \in R(N_1, m_1)$.

Lemma 2.3 (see Fig. 3) can be used to find a counter-example m'_1 to some property F in N_1 (where F is a formula whose variables are in P_1), just by looking at the reachable markings of N_2. Indeed, it is enough to find a marking m'_2 reachable in N_2 such that $m'_2 \models E \wedge \neg F$.

Our second property can be used to prove that every reachable marking of N_1 can be traced back to at least one marking of N_2 using the reduction constraints. (While this mapping is surjective, it is not a function since a state in N_2 could be associated with multiple states in N_1.)

Lemma 2.4. (Invariance Checking [19, 5])
Assume $(N_1, m_1) \equiv_E (N_2, m_2)$. Then for all m'_1 in $R(N_1, m_1)$ there is m'_2 in $R(N_2, m_2)$ such that $m'_1 \equiv_E m'_2$.

Using Lemma 2.4 (see Fig. 4), we can easily extract an invariant on N_1 from an invariant on N_2. If property $E \wedge \neg F$ is not reachable on N_2, then we can prove that $\neg F$ is not reachable on N_1, meaning F is an invariant. This property (the *invariant conservation* theorem of Sect. 2.3) ensures the soundness of the model checking technique implemented in our tool.

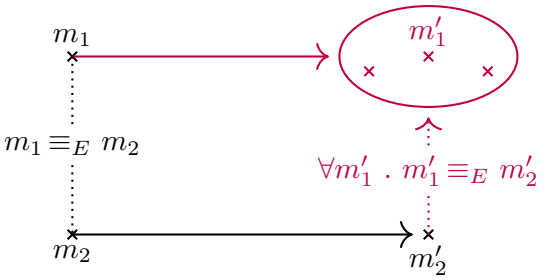


Figure 3: Illustration of Lemma 2.3.

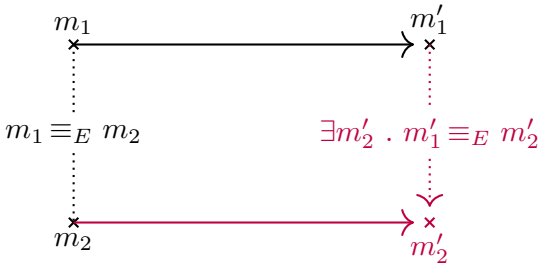


Figure 4: Illustration of Lemma 2.4.

Straightforward application

We now recall from [19, 5] a general method for combining polyhedral reductions with SMT-based procedures. Assume we have $(N_1, m_1) \equiv_E (N_2, m_2)$, where the nets N_1, N_2 have sets of places P_1, P_2 respectively. In the following, we use $\boldsymbol{p_1} \triangleq (p^1_1, \ldots, p^1_k)$ and $\boldsymbol{p_2} \triangleq (p^2_1, \ldots, p^2_l)$ for the places in P_1 and P_2. We also consider (disjoint) sequences of variables, $\boldsymbol{x}$ and $\boldsymbol{y}$, ranging over (the places of) N_1 and N_2. With these notations, we denote $\tilde{E}(\boldsymbol{x}, \boldsymbol{y})$ the formula obtained from E where place names in N_1 are replaced with variables in $\boldsymbol{x}$, and place names in N_2 are replaced with variables in $\boldsymbol{y}$. When we have the same place in both nets, say $p^1_i = p^2_j$, we also add the constraint $(x_i = y_j)$ to

$\tilde{E}$ in order to avoid shadowing variables. (Remark that $\tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2})$ is equivalent to E, since equalities $x_i = y_j$ become tautologies in this case.)

$$\tilde{E}(\boldsymbol{x}, \boldsymbol{y}) \triangleq E\{\boldsymbol{p_1} \leftarrow \boldsymbol{x}\}\{\boldsymbol{p_2} \leftarrow \boldsymbol{y}\} \wedge \bigwedge_{\{(i,j)|p_i^1 = p_j^2\}} (x_i = y_j) \tag{2}$$

Assume F_1 is a property that we want to study on N_1, such that $\mathrm{FV}(F_1) \subseteq P_1$. We construct an equivalent formula F_2, to study on N_2, which we call the E-transform formula of F_1.

Definition 2.5. (E-Transform Formula)
Assume $(N_1, m_1) \equiv_E (N_2, m_2)$ and take F_1 a property with variables in P_1, i.e. $\mathrm{FV}(F_1) \subseteq P_1$. Formula $F_2(\boldsymbol{y}) \triangleq \exists \boldsymbol{x} \,.\, \tilde{E}(\boldsymbol{x}, \boldsymbol{y}) \wedge F_1(\boldsymbol{x})$ is the E-transform of F_1.

The following property states that, to check F_1 reachable in N_1, it is enough to check the corresponding E-transform formula F_2 on N_2.

Theorem 2.6. (Reachability Conservation [19, 5])
Assume $(N_1, m_1) \equiv_E (N_2, m_2)$ and that F_2 is the E-transform of formula F_1 on N_1. Then, formula F_1 is reachable in N_1 if and only if F_2 is reachable in N_2.

Since F_1 invariant on N_1 is equivalent to $\neg F_1$ not reachable, we can directly infer an equivalent conservation theorem for invariance:

Corollary 2.7. (Invariant Conservation)
Assume $(N_1, m_1) \equiv_E (N_2, m_2)$ and that F_2 is the E-transform of formula $\neg F_1$ on N_1. Then F_1 is an invariant on N_1 if and only if $\neg F_2$ is an invariant on N_2.

Negating the E-transform formula, as done in Corollary 2.7, introduces universally quantified variables that may impact the solver performance since we require the "full" LIA theory instead of only the quantifier-free fragment. We showed a pragmatic solution to get around this problem in [28], where we propose a quantifier elimination procedure specific to the particular structure of constraints that occur with structural reductions.

3. Parametric reduction rules and equivalence

E-abstraction is defined on marked nets (Definition 2.1), thus the reduction rules defined in [19, 5], which are E-abstraction equivalences, mention marked nets as well. Their soundness was proven manually, using constrained parameters for initial markings. Such constraints on markings are called *coherency constraints*.

3.1. Coherency constraints

We define a notion of *coherency constraint*, C, that must hold not only in the initial state, but also in a sufficiently large subset of reachable markings, as formalized next. We have already seen an example with the constraint $C_1 \triangleq (y_2 = 0)$ used in rule (CONCAT). Without the use of C_1, rule

(CONCAT) would be unsound since net N_2 (right of Fig. 1) could fire transition b more often than its counterpart, N_1.

Since C is a predicate on markings, we equivalently consider it as a subset of markings or as a logic formula, so that we may equivalently write $m \models C$ or $m \in C$ to indicate that $C(m)$ is true.

Definition 3.1. (Coherent Net)
Given a Petri net N and a predicate C on markings, we say that N satisfies the coherency constraint C, or equivalently, that (N, C) is a coherent net, if and only if for all firing sequences $m \stackrel{\sigma}{\Rightarrow} m'$ with $m \in C$, we have

$$\exists m'' \in C \,.\, m \stackrel{\sigma\rangle}{\Longrightarrow} m'' \wedge m'' \stackrel{\epsilon}{\Rightarrow} m'$$

Intuitively, if we consider that all τ transitions are irreversible choices, then we can define a partial order on markings with $m < m'$ whenever $m \stackrel{\tau}{\rightarrow} m'$ holds. Then, markings satisfying the coherency constraint C must be minimal with respect to this partial order.

In this paper, we wish to prove automatically the soundness of a given reduction rule. A reduction rule basically consists of two nets with their coherency constraints, and a Presburger relation between markings.

Definition 3.2. (Parametric Reduction Rule)
A parametric reduction rule is written $(N_1, C_1) >_E (N_2, C_2)$, where (N_1, C_1) and (N_2, C_2) are both coherent nets, and C_1, C_2, and E are Presburger formulas whose free variables are in $P_1 \cup P_2$.

A given reduction rule $(N_1, C_1) >_E (N_2, C_2)$ is a candidate, which we will analyze to prove its soundness: is it an E-abstraction equivalence?

Our analysis relies on a richer definition of E-abstraction, namely parametric E-abstraction (Definition 3.3, next), which includes the coherency constraints C_1, C_2. Parametric E-abstraction entails E-abstraction for each instance of its parameters (Theorem 3.4, below). Essentially, for any sequence $m_1 \stackrel{\sigma}{\Rightarrow} m_1'$ with $m_1 \in C_1$, there exists a marking m_2' such that $m_1' \equiv_E m_2'$; and for every marking $m_2 \in C_2$ compatible with m_1, i.e., $m_1 \equiv_E m_2$, all markings m_2' compatible with m_1' (i.e., $m_1' \equiv_E m_2'$) can be reached from m_2 by the same observable sequence σ. To ease the presentation, we define the notation

$$m_1 \langle C_1 E C_2 \rangle m_2 \triangleq m_1 \models C_1 \wedge m_1 \equiv_E m_2 \wedge m_2 \models C_2 \tag{3}$$

Definition 3.3. (Parametric E-Abstraction)
Assume $(N_1, C_1) >_E (N_2, C_2)$ is a parametric reduction rule. We say that (N_2, C_2) is a parametric E-abstraction of (N_1, C_1), denoted $(N_1, C_1) \preceq_E (N_2, C_2)$ if and only if:

(S1) for all markings m_1 satisfying C_1 there exists a marking m_2 such that $m_1 \langle C_1 E C_2 \rangle m_2$;

(S2) for all firing sequences $m_1 \stackrel{\epsilon}{\Rightarrow} m_1'$ and all markings m_2, we have $m_1 \equiv_E m_2$ implies $m_1' \equiv_E m_2$;

(S3) for all firing sequences $m_1 \stackrel{\sigma}{\Rightarrow} m_1'$ and all marking pairs m_2, m_2', if $m_1 \langle C_1 E C_2 \rangle m_2$ and $m_1' \equiv_E m_2'$ then we have $m_2 \stackrel{\sigma}{\Rightarrow} m_2'$.

We say that (N_1, C_1) and (N_2, C_2) are in parametric E-equivalence, denoted $(N_1, C_1) \approxeq_E (N_2, C_2)$, when we have both $(N_1, C_1) \preceq_E (N_2, C_2)$ and $(N_2, C_2) \preceq_E (N_1, C_1)$.

Condition (S1) corresponds to the solvability of the Presburger formula E with respect to the marking predicates C_1 and C_2. Condition (S2) ensures that silent transitions of N_1 are abstracted away by the formula E, and are therefore invisible to N_2. Condition (S3) follows closely condition (A2) of the standard E-abstraction equivalence.

Note that equivalence $\approxeq$ is not a bisimulation, in the same way that $\equiv$ from Definition 2.1. It is defined only for observable sequences starting from states satisfying the coherency constraints C_1 of N_1 or C_2 of N_2, and so this relation is usually not true on every pair of equivalent markings $m_1 \equiv_E m_2$.

3.2. Instantiation law

Parametric E-abstraction implies E-abstraction for every instance pair satisfying the coherency constraints C_1, C_2.

Theorem 3.4. (Parametric E-Abstraction Instantiation)
Assume $(N_1, C_1) \preceq_E (N_2, C_2)$ is a parametric E-abstraction. Then for every pair of markings m_1, m_2 we have $m_1 \langle C_1 E C_2 \rangle m_2$ implies $(N_1, m_1) \sqsubseteq_E (N_2, m_2)$.

Proof:
Consider $(N_1, C_1) \preceq_E (N_2, C_2)$, a parametric E-abstraction, and m_1, m_2 such that $m_1 \langle C_1 E C_2 \rangle m_2$ holds. By definition of $m_1 \langle C_1 E C_2 \rangle m_2$ (that is Equation (3)), condition (A1) of Definition 2.1 is immediately satisfied. We show (A2) by considering an observable sequence $(N_1, m_1) \stackrel{\sigma}{\Rightarrow} (N_1, m_1')$. Since m_1 satisfies the coherency constraint C_1, we get from Definition 3.1 a marking $m_1'' \in C_1$ such that $m_1 \stackrel{\sigma}{\Rightarrow} m_1'' \stackrel{\epsilon}{\Rightarrow} m_1'$ holds. By applying (S1) to m_1'', we get a marking m_2' such that $m_1'' \langle C_1 E C_2 \rangle m_2'$ holds, which implies $m_1'' \equiv_E m_2'$. Then, by applying (S2) to $m_1'' \stackrel{\epsilon}{\Rightarrow} m_1'$, we obtain the expected result $m_1' \equiv_E m_2'$. Finally, for all markings m_2' such that $m_1' \equiv_E m_2'$, we conclude $m_2 \stackrel{\sigma}{\Rightarrow} m_2'$ from (S3). Condition (A2) is proved, hence $(N_1, m_1) \sqsubseteq_E (N_2, m_2)$ holds. □

4. Automated proof procedure

Our automated proof procedure receives a candidate reduction rule (Definition 3.2) as input, and has three possible outcomes: (i) the candidate is proven sound, congratulations you have established a new parametric E-abstraction equivalence; (ii) the candidate is proven unsound, try to understand why and fix it (see examples in Sect 9); or (iii) we cannot conclude, because part of our procedure relies on a semi-algorithm for expressing the set of reachable markings of a flat subnet as a linear constraint (even if the problem is decidable; see Sects. 5 and 6).

Given the candidate reduction rule, the procedure generates SMT queries, which we call *core requirements* (defined in Sect. 4.2) that are solvable if and only if the candidate is a parametric E-abstraction (Theorems 4.8 and 4.9, Sect. 4.3). We express these constraints into Presburger predicates, so it is enough to use solvers for the theory of formulas on Linear Integer Arithmetic, what is known

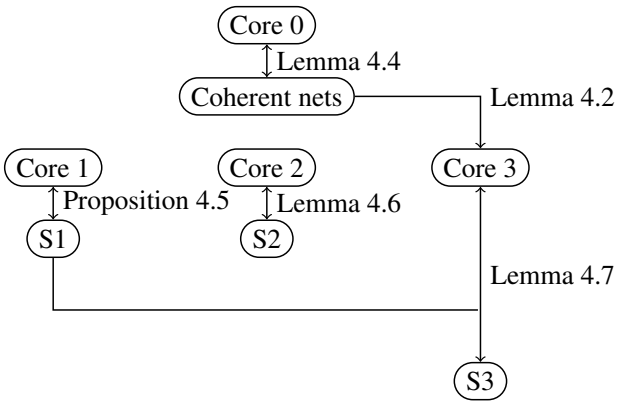


Figure 5: Detailed dependency relations.

as LIA in SMT-LIB [29]. We illustrate the results given in this section using a diagram (Fig. 5) that describe the dependency relations between conditions (S1), (S2), (S3) and their encoding as core requirements.

4.1. Presburger encoding of Petri net semantics

We start by defining a few formulas that ease the subsequent expression of core requirements. This will help with the most delicate point of our encoding, which relies on how to encode sequences of transitions. Note that the coherency constraints of reduction rules are already defined as Presburger formulas.

In the following, we use $\boldsymbol{x}$ for the vector of variables $(x_1, \ldots, x_n)$, corresponding to the places $p_1, \ldots, p_n$ of P, and $F(\boldsymbol{x})$ for a formula whose variables are included in $\boldsymbol{x}$. We say that a mapping m of $\mathbb{N}^P$ is a *model* of F, denoted $m \models F$, if the ground formula $F(m) \triangleq F(m(p_1), \ldots, m(p_n))$ is true. Hence, we can also interpret F as a predicate over markings. Finally, we define the semantics of F as the set $[\![F]\!] \triangleq \{m \in \mathbb{N}^P \mid m \models F\}$. As usual, we say that a predicate F is *valid*, denoted $\models F$, when all its interpretations are true ($[\![F]\!] = \mathbb{N}^P$). In order to keep track of fired transitions in our encoding, and without any loss of generality we assume that our alphabet of labels Σ is a subset of the natural numbers ($\Sigma \subset \mathbb{N}^*$), except 0 that is reserved for τ.

We define next a few Presburger formulas that express properties on markings of a net N. For instance, Equation (4) below defines the predicate ENBL_t, for a given transition t, which corresponds exactly to the markings that enable t. We also define a linear predicate $\mathrm{T}(\boldsymbol{x}, \boldsymbol{x'}, a)$ that describes the relation between the markings before ($\boldsymbol{x}$) and after ($\boldsymbol{x'}$) firing a transition with label a. With this convention, formula $\mathrm{T}(m, m', a)$ holds if and only if $m \xrightarrow{t} m'$ holds for some transition t such that $l(t) = a$ (which implies $a \neq 0$).

$$\mathrm{ENBL}_t(\boldsymbol{x}) \triangleq \textstyle\bigwedge_{i \in 1..n} (x_i \geqslant \mathrm{Pre}(t, p_i)) \tag{4}$$

$$\Delta_t(\boldsymbol{x}, \boldsymbol{x'}) \triangleq \textstyle\bigwedge_{i \in 1..n} (x'_i = x_i + \mathrm{Post}(t, p_i) - \mathrm{Pre}(t, p_i)) \tag{5}$$

$$\mathrm{T}(\boldsymbol{x}, \boldsymbol{x'}, a) \triangleq \textstyle\bigvee_{t \in T} (\mathrm{ENBL}_t(\boldsymbol{x}) \wedge \Delta_t(\boldsymbol{x}, \boldsymbol{x'}) \wedge a = l(t)) \tag{6}$$

We admit the following, for all markings m, m' and label a:

$$\models \mathrm{T}(m, m', a) \iff \exists t \,.\, m \xrightarrow{t} m' \wedge l(t) = a \tag{7}$$

In order to define the core requirements, we additionally require a predicate $\tau_C^*(\boldsymbol{x}, \boldsymbol{x'})$ encoding the markings reachable by firing any sequence of silent transitions from a state satisfying the coherency constraints C. And so, the following constraint must hold:

$$\models m \in C \implies (\tau_C^*(m, m') \iff m \overset{\epsilon}{\Rightarrow} m') \tag{8}$$

Since $m \overset{\epsilon}{\Rightarrow} m'$ may fire an arbitrary number of silent transitions τ, the predicate τ_C is not guaranteed to be expressible as a Presburger formula in the general case. Yet, in Sect. 5, we characterize the Petri nets for which τ_C can be expressed in Presburger arithmetic, which include all the polyhedral reductions that we meet in practice (we explain why).

Thanks to this predicate, we define the formula $\acute{\mathrm{T}}_C(\boldsymbol{x}, \boldsymbol{x'}, a)$ encoding the reachable markings from a marking satisfying the coherency constraint C, by firing any number of silent transitions, followed by a transition labeled with a. Then, we define $\hat{\mathrm{T}}$ which extends $\acute{\mathrm{T}}$ with any number of silent transitions after a and also allows for only silent transitions (no transition a).

$$\acute{\mathrm{T}}_C(\boldsymbol{x}, \boldsymbol{x'}, a) \triangleq \exists \boldsymbol{y} \,.\, \tau_C^*(\boldsymbol{x}, \boldsymbol{y}) \wedge \mathrm{T}(\boldsymbol{y}, \boldsymbol{x'}, a) \tag{9}$$

$$\hat{\mathrm{T}}_C(\boldsymbol{x}, \boldsymbol{x'}, a) \triangleq \Big(\exists \boldsymbol{z} \,.\, \acute{\mathrm{T}}_C(\boldsymbol{x}, \boldsymbol{z}, a) \wedge C(\boldsymbol{z}) \wedge \tau_C^*(\boldsymbol{z}, \boldsymbol{x'}))\Big) \tag{10}$$

$$\vee \big(a = 0 \wedge \tau_C^*(\boldsymbol{x}, \boldsymbol{x'})\big) \tag{11}$$

Lemma 4.1. For any markings m, m' and label a such that $m \in C$, we have $\models \acute{\mathrm{T}}_C(m, m', a)$ if and only if $m \overset{a\rangle}{\Rightarrow} m'$ holds.

Proof:
We show both directions separately.

- Assume $m \overset{a\rangle}{\Rightarrow} m'$. By definition, this implies that there exists m'' and a transition t such that $l(t) = a$ and $m \overset{\epsilon}{\Rightarrow} m'' \xrightarrow{t} m'$. Therefore, $\tau_C^*(m, m'')$ is valid by Equation (8), and $\mathrm{T}(m'', m', a)$ is valid by Equation (7), hence the expected result $\models \acute{\mathrm{T}}_C(m, m', a)$.
- Conversely, assume $\acute{\mathrm{T}}_C(m, m', a)$ is valid. Then, by Equation (9) there exists a marking m'' such that both $\tau_C^*(m, m'')$ and $T(m'', m', a)$ are valid. From Equation (8), we get $m \overset{\epsilon}{\Rightarrow} m''$, and Equation (7) implies $\exists t \,.\, m'' \xrightarrow{t} m' \wedge l(t) = a$. Thus, $m \overset{\epsilon}{\Rightarrow} m'' \xrightarrow{t} m'$, that is the expected result $m \overset{a\rangle}{\Rightarrow} m'$.

□

Lemma 4.2. Given a coherent net (N, C), for any markings m, m' such that $m \in C$ and $a \in \Sigma \cup \{0\}$, we have $\models \hat{\mathrm{T}}_C(m, m', a)$ if and only if either $m \overset{\epsilon}{\Rightarrow} m'$ and $a = 0$, or $m \overset{a}{\Rightarrow} m'$.

Proof:
We show both directions separately.

- Assume $m \overset{\epsilon}{\Rightarrow} m'$ and $a = 0$, then $\tau_C^*(m, m')$ is valid by Equation (8), hence the expected result $\models \hat{\mathrm{T}}_C(m, m', a)$ from Equation (11).
- Assume $m \overset{a}{\Rightarrow} m'$. From Definition 3.1 (coherent net), there exists $m'' \in C$ such that $m \overset{a\rangle}{\Rightarrow} m'' \overset{\epsilon}{\Rightarrow} m'$. Then, we get $\models \acute{\mathrm{T}}_C(m, m'', a)$ from Lemma 4.1, and $\models \tau_C^*(m'', m')$ from Equation (8). Consequently, $\hat{\mathrm{T}}_C(m, m', a)$ is valid from Equation (10).
- Conversely, assume $\hat{\mathrm{T}}_C(m, m', a)$ holds by Equation (11), then $a = 0$ and $\models \tau_C^*(m, m')$, which implies $m \overset{\epsilon}{\Rightarrow} m'$ by Equation (8). This is the expected result.
- Finally, assume $\hat{\mathrm{T}}_C(m, m', a)$ holds by Equation (10), then there exists a marking $m'' \in C$ such that $\models \acute{\mathrm{T}}_C(m, m'', a)$ and $\models \tau_C^*(m'', m')$. This implies $m \overset{a\rangle}{\Rightarrow} m'' \overset{\epsilon}{\Rightarrow} m'$ from Lemma 4.1 and Equation (8). This implies the expected result $m \overset{a}{\Rightarrow} m'$. □

As with the E-transform formula in Sect. 2.5, we denote $\tilde{E}(\boldsymbol{x}, \boldsymbol{y})$ the formula obtained from E where free variables are substituted as follows: place names in N_1 are replaced with variables in $\boldsymbol{x}$, and place names in N_2 are replaced with variables in $\boldsymbol{y}$ (making sure that bound variables of E are renamed to avoid interference). When the same place occurs in both nets, say $p_i^1 = p_j^2$, we also add the equality constraint $(x_i = y_j)$ to $\tilde{E}$ in order to preserve this equality constraint.

4.2. Core requirements: parametric E-abstraction encoding

In order to check conditions (S1)–(S3) of parametric E-abstraction (Definition 3.3), we define a set of Presburger formulas, called *core requirements*, to be verified using an external SMT solver ((Core 1) to (Core 3)). You will find an illustration of these requirements in Figs. 6–9. The satisfaction of these requirements entail the parametric E-abstraction relation. We have deliberately stressed the notations to prove that (N_2, C_2) is a parametric E-abstraction of (N_1, C_1). Of course, each constraint must be checked in both directions to obtain the equivalence. Also, in order not to overload the notations, we assume that the transition relations are clear in the context if they belong to N_1 or N_2.

4.2.1. Verifying that a net is coherent.

The first step consists in verifying that both nets N_1 and N_2 satisfy their coherency constraints C_1 and C_2 (the coherency constraint is depicted in Figure 6). We recall Definition 3.1:

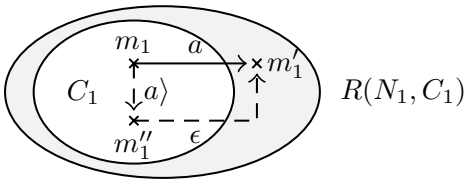


Figure 6: Illustration of (Core 0).

Definition 3.1 ***(Coherent Net)***
For all firing sequence $m \stackrel{\sigma}{\Rightarrow} m'$ with $m \in C$, there exists a marking m'' satisfying C such that $m \stackrel{\sigma\rangle}{\Longrightarrow} m''$ and $m'' \stackrel{\epsilon}{\Rightarrow} m'$.

We encode a simpler relation, below, with sequences σ of size 1. This relies on the following result:

Lemma 4.3. (N, C) is coherent if and only if for all firing sequence $m \stackrel{a\rangle}{\Longrightarrow} m'$ with $m \in C$ and $a \in \Sigma$, we have $\exists m'' \in C \,.\, m \stackrel{a\rangle}{\Longrightarrow} m'' \wedge m'' \stackrel{\epsilon}{\Rightarrow} m'$.

We deliberately consider a firing sequence $m \stackrel{a\rangle}{\Longrightarrow} m'$ (and not $m \stackrel{a}{\Rightarrow} m'$), since the encoding relies only on $\acute{\mathrm{T}}_C$ (that is, $\stackrel{a\rangle}{\Longrightarrow}$), not on $\hat{\mathrm{T}}_C$ (that is, $\stackrel{a}{\Rightarrow}$).

Proof:
The "only if" part is immediate, as a particular case of Definition 3.1 and noting that $m \stackrel{a\rangle}{\Longrightarrow} m'$ implies $m \stackrel{a}{\Rightarrow} m'$. Conversely, assume the property stated in the lemma is true. Then, we show by induction on the size of σ, that Definition 3.1 holds for any σ. Note that the base case $\sigma = \epsilon$ always holds, for any net, by taking $m'' = m$. Now, consider a non-empty sequence $\sigma = \sigma'.a$ and $m \stackrel{\sigma'.a}{\Longrightarrow} m'$ with $m \in C$. By definition, there exists m_1 and m_2 such that $m \stackrel{\sigma'}{\Rightarrow} m_1 \stackrel{a\rangle}{\Longrightarrow} m_2 \stackrel{\epsilon}{\Rightarrow} m'$. By induction hypothesis, on $m \stackrel{\sigma'}{\Rightarrow} m_1$, there exists $m_3 \in C$ such that $m \stackrel{\sigma'\rangle}{\Longrightarrow} m_3 \stackrel{\epsilon}{\Rightarrow} m_1$. Therefore, we have $m \stackrel{\sigma'\rangle}{\Longrightarrow} m_3 \stackrel{\epsilon}{\Rightarrow} m_1 \stackrel{a\rangle}{\Longrightarrow} m_2 \stackrel{\epsilon}{\Rightarrow} m'$, which can simply be written $m \stackrel{\sigma'\rangle}{\Longrightarrow} m_3 \stackrel{a\rangle}{\Longrightarrow} m_2 \stackrel{\epsilon}{\Rightarrow} m'$. Using the property stated in the lemma on $m_3 \stackrel{a\rangle}{\Longrightarrow} m_2$, we get a marking $m_4 \in C$ such that $m_3 \stackrel{a\rangle}{\Longrightarrow} m_4 \stackrel{\epsilon}{\Rightarrow} m_2$. Hence, $m \stackrel{\sigma'\rangle}{\Longrightarrow} m_3 \stackrel{a\rangle}{\Longrightarrow} m_4 \stackrel{\epsilon}{\Rightarrow} m_2 \stackrel{\epsilon}{\Rightarrow} m'$ holds, which can be simplified as $m \stackrel{\sigma'.a\rangle}{\Longrightarrow} m_4 \stackrel{\epsilon}{\Rightarrow} m'$. This is the expected result. □

Therefore, we can encode Definition 3.1 using the following formula:

$$\forall \boldsymbol{p}, \boldsymbol{p'}, a \,.\, C(\boldsymbol{p}) \wedge \acute{\mathrm{T}}_C(\boldsymbol{p}, \boldsymbol{p'}, a) \implies \exists \boldsymbol{p''} \,.\, C(\boldsymbol{p''}) \wedge \acute{\mathrm{T}}_C(\boldsymbol{p}, \boldsymbol{p''}, a) \wedge \tau_C^*(\boldsymbol{p''}, \boldsymbol{p'}) \qquad \text{(Core 0)}$$

Lemma 4.4. Given a Petri net N, the constraint (Core 0) is valid if and only if the net satisfies the coherency constraint C.

Proof:
Constraint (Core 0) is an immediate translation of the property stated in Lemma 4.3. □

Given a net N, a constraint C expressed as a Presburger formula, and a formula τ_C^* that captures $\stackrel{\epsilon}{\Rightarrow}$ transitions (as obtained in Sect. 5), we are now able to check automatically that a net (N, C) is coherent. Thus, from now on, we assume that the considered nets (N_1, C_1) and (N_2, C_2) are indeed coherent.

4.2.2. Coherent solvability

The first requirement of the parametric E-abstraction relates to the solvability of formula E with regard to the coherency constraints C_1, and is encoded by (Core 1). This requirement ensures that every marking of N_1 satisfying C_1 can be associated to at least one marking of N_2 satisfying C_2. Let us recall (S1), taken from Definition 3.3:

Definition 3.3 ***(S1)***
For all markings m_1 satisfying C_1 there exists a marking m_2 such that $m_1 \langle C_1 E C_2 \rangle m_2$.

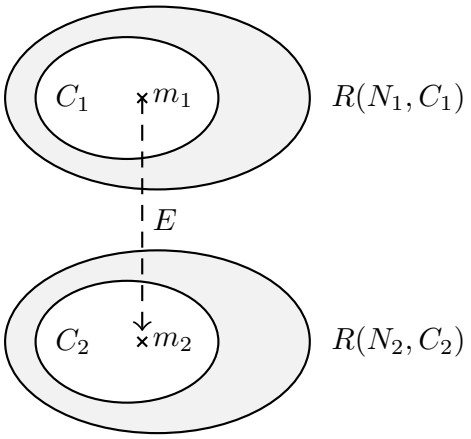


Figure 7: Illustration of (Core 1).

Condition (S1) is depicted in Figure 7. We propose to encode it by the following Presburger formula:

$$\forall \boldsymbol{x} \,.\, C_1(\boldsymbol{x}) \implies \exists \boldsymbol{y} \,.\, \tilde{E}(\boldsymbol{x}, \boldsymbol{y}) \wedge C_2(\boldsymbol{y}) \qquad \text{(Core 1)}$$

Since the encoding is immediate, we admit this proposition:

Proposition 4.5. The constraint (Core 1) is valid if and only if (S1) holds.

4.2.3. Silent constraints

So far, we have focused on the specific case of coherent nets, which refers to intermediate coherent markings. Another notable feature of parametric E-abstractions is the ability to fire any number of silent transitions without altering the solutions of E. In other words, if two markings, m_1 and m_2, are solutions of E, then firing any silent sequence from m_1 (or m_2) will always lead to a solution of $E \wedge m_2$ (or $E \wedge m_1$). This means that silent transitions must be invisible to the other net.

Let us recall (S2), taken from Definition 3.3:

Definition 3.3 ***(S2)***
For all firing sequences $m_1 \overset{\epsilon}{\Rightarrow} m_1'$ and all markings m_2, we have $m_1 \equiv_E m_2$ implies $m_1' \equiv_E m_2$.

It actually suffices to show the result for each silent transition $t \in T_1$ taken separately:

Lemma 4.6. Condition (S2) holds if and only if, for all markings m_1, m_2 such that $m_1 \equiv_E m_2$, and for all $t_1 \in T_1$ such that $l_1(t_1) = \tau$, we have $m_1 \xrightarrow{t_1} m_1' \implies m_1' \equiv_E m_2$.

Proof:
The "only if" way is only a particular case of (S2) with a single silent transition t_1. For the "if" way, (S2) is shown from the given property by transitivity. □

Thanks to this result, we encode (S2) by the following core requirement:

$$\forall \boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p'_1} \,.\, \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tau(\boldsymbol{p_1}, \boldsymbol{p'_1}) \implies \tilde{E}(\boldsymbol{p'_1}, \boldsymbol{p_2}) \qquad \text{(Core 2)}$$

where $\tau(\boldsymbol{x}, \boldsymbol{x'})$ is defined as $\tau(\boldsymbol{x}, \boldsymbol{x'}) \triangleq \bigvee_{t \in T | l(t) = \tau} (\mathrm{ENBL}_t(\boldsymbol{x}) \wedge \Delta_t(\boldsymbol{x}, \boldsymbol{x'}))$

4.2.4. Reachability

Let us recall the definition of (S3), taken from Definition 3.3:

> **Definition 3.3** ***(S3)***
> For all firing sequences $m_1 \stackrel{\sigma}{\Rightarrow} m'_1$ and all marking pairs m_2, m'_2, if $m_1 \langle C_1 E C_2 \rangle m_2$ and $m'_1 \equiv_E m'_2$ then we have $m_2 \stackrel{\sigma}{\Rightarrow} m'_2$.

Condition (S3) mentions sequences σ of arbitrary length. We encode it with a formula dealing only with sequences of length at most 1, thanks to the following result:

Lemma 4.7. Given a parametric reduction rule $(N_1, C_1) >_E (N_2, C_2)$ which satisfies condition (S1), then condition (S3) holds if and only if for all firing sequence $m_1 \stackrel{\sigma}{\Rightarrow} m'_1$ with $\sigma = \epsilon$ or $\sigma = a$ with $a \in \Sigma$, and all markings m_2, m'_2, we have $m_1 \langle C_1 E C_2 \rangle m_2 \wedge m'_1 \equiv_E m'_2 \implies m_2 \stackrel{\sigma}{\Rightarrow} m'_2$.

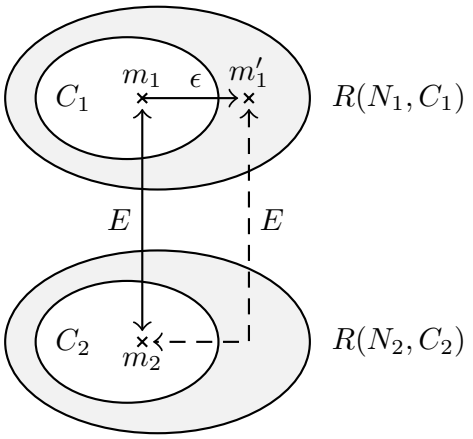


Figure 8: Illustration of (Core 2).

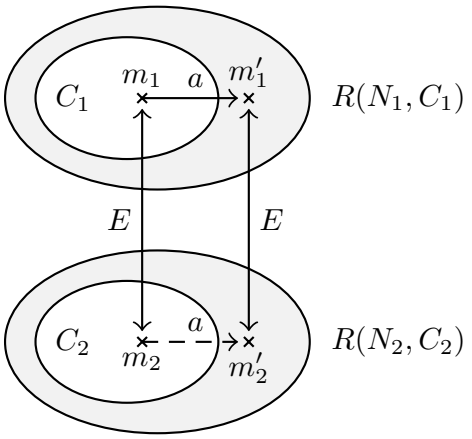


Figure 9: Illustration of (Core 3).

Proof:
The given property is necessary as a particular case of (S3) taking $\sigma = a$ or $\sigma = \epsilon$. Conversely, assume the given property holds. We show by induction on the size of σ that (S3) holds for any sequence σ.

The base cases $\sigma = a$ and $\sigma = \epsilon$ are ensured by hypothesis. Now, consider a non-empty sequence $\sigma = \sigma'.a$, and $m_1 \overset{\sigma}{\Rightarrow} m_1'$ (i), as well as markings m_2, m_2' such that $m_1 \langle C_1 E C_2 \rangle m_2$ and $m_1' \equiv_E m_2'$ holds. We have to show $m_2 \overset{\sigma}{\Rightarrow} m_2'$. From (i), we have $m_1 \overset{\sigma'.a}{\Longrightarrow} m_1'$, that is, there exists a marking u_1 such that $m_1 \overset{\sigma'}{\Rightarrow} u_1 \overset{a}{\Rightarrow} m_1'$ (ii). By Definition 3.1, there exists $u_1' \in C_1$ such that $m_1 \overset{\sigma'}{\Longrightarrow} u_1' \overset{\epsilon}{\Rightarrow} u_1$ (iii). Also, by condition (S1), there exists a marking u_2' of N_2 such that $u_1' \langle C_1 E C_2 \rangle u_2'$, which implies $u_1' \equiv_E u_2'$ (iv). Hence, by induction hypothesis on $m_1 \overset{\sigma'}{\Rightarrow} u_1'$, we have $m_2 \overset{\sigma'}{\Rightarrow} u_2'$ (α) From (iii) and (ii), we get $u_1' \overset{a}{\Rightarrow} m_1'$ (v). Applying the property of the lemma on (iv) and (v), we get $u_2' \overset{a}{\Rightarrow} m_2'$ (β). Combining (α) and (β) leads to $m_2 \overset{\sigma'.a}{\Longrightarrow} m_2'$, that is the expected result $m_2 \overset{\sigma}{\Rightarrow} m_2'$. □

Thanks to Lemma 4.7, we can encode (S3) by the following formula:

$$\forall \boldsymbol{p_1}, \boldsymbol{p_2}, a, \boldsymbol{p_1'}, \boldsymbol{p_2'} \;.\; \langle C_1 E C_2 \rangle(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \hat{\mathrm{T}}_{C_1}(\boldsymbol{p_1}, \boldsymbol{p_1'}) \wedge \tilde{E}(\boldsymbol{p_1'}, \boldsymbol{p_2'}) \implies \hat{\mathrm{T}}_{C_2}(\boldsymbol{p_2}, \boldsymbol{p_2'}) \quad \text{(Core 3)}$$

4.3. Global procedure

In this section, we consider the full process for proving parametric E-abstraction. We demonstrate that verifying constraints (Core 0) to (Core 3) is sufficient for obtaining a sound abstraction (Theorem 4.8). We also prove that these conditions are necessary (Theorem 4.9).

Theorem 4.8. (Soundness)
Given two nets N_1, N_2 and constraints C_1, C_2 expressed as Presburger formulas, if core requirement (Core 0) holds for both (N_1, C_1) and (N_2, C_2), and if core requirements (Core 1), (Core 2), and (Core 3) are valid, then the rule is a parametric E-abstraction: $(N_1, C_1) \preceq_E (N_2, C_2)$.

Proof:
If (Core 0) holds for (N_1, C_1), then (N_1, C_1) is a coherent net by Lemma 4.4. Similarly for (N_2, C_2). Hence, $(N_1, C_1) >_E (N_2, C_2)$ is a parametric reduction rule. By Proposition 4.5, and since (Core 1) is valid, we get (S1) from Definition 3.3. Similarly, by Lemma 4.6, and since (Core 2) is valid, we get (S2). Finally, (S3) holds by Lemma 4.7 since (Core 3) is valid and since (S1) is known to hold. (S1), (S2), (S3) entail $(N_1, C_1) \preceq_E (N_2, C_2)$ by Definition 3.3. □

The converse also holds:

Theorem 4.9. (Completeness)
Given a parametric E-abstraction $(N_1, C_1) \preceq_E (N_2, C_2)$, then core requirements (Core 1), (Core 2), and (Core 3) are valid, and (Core 0) holds for both (N_1, C_1) and (N_2, C_2).

Proof:
By hypothesis, conditions (S1), (S2) and (S3) hold and (N_1, C_1) and (N_2, C_2) are coherent nets. Then, Lemma 4.4 implies that (Core 0) holds for both nets. Besides, Proposition 4.5 and Lemmas 4.6 and 4.7 ensure that (Core 1), (Core 2), and (Core 3) are valid. □

Consequently, checking E-abstraction equivalence amounts to check that SMT formulas (Core 0)-(Core 3) are valid on both nets.

Our approach relies on our ability to express (arbitrarily long) sequences $m \overset{\epsilon}{\Rightarrow} m'$ thanks to a formula $\tau_C^*(\boldsymbol{x}, \boldsymbol{x'})$. This is addressed in the next section.

5. Accelerating the silent transition relation

The previous results, including Theorems 4.8 and 4.9, rely on our ability to express the reachability set of silent transitions as a Presburger predicate, denoted τ_C^*. Finding a finite formula τ_C^* that captures an infinite state space is not granted, since τ-sequences may be of arbitrary length. However, we now show that, since τ transitions must be abstracted away by E in order to define a valid parametric E-equivalence (condition (S2)), and since E is itself a Presburger formula, this implies that τ_C^* corresponds to the reachability set of a *flat* subnet [25], which is expressible as a Presburger formula too.

We define the *silent reachability set* of a net N from a coherent constraint C as $R_\tau(N, C) \triangleq \{m' \mid m \models C \wedge m \stackrel{\epsilon}{\Rightarrow} m'\}$. We now want to find a predicate $\tau_C^*(\boldsymbol{p}, \boldsymbol{p'})$ that satisfies the relation:

$$R_\tau(N, C) \triangleq \{m' \mid m' \models \exists \boldsymbol{x} \,.\, C(\boldsymbol{x}) \wedge \tau_C^*(\boldsymbol{x}, \boldsymbol{x'})\} \tag{8}$$

In order to express the formula τ_C^*, we first use the tool `FAST` [23], designed for the analysis of infinite systems, and that permits to compute the reachability set of a given Vector Addition System with States (VASS). Note that a Petri net can be transformed to an equivalent VASS with the same reachability set, so the formal presentation of VASS can be skipped. The algorithm implemented in `FAST` is a semi-procedure, for which we have some termination guarantees whenever the net is flat [24], i.e. its corresponding VASS can be unfolded into a VASS without nested cycles, called a flat VASS. Equivalently, a net N is flat for some coherent constraint C if its language is flat, that is, there exists some finite sequence $\varrho_1 \dots \varrho_k \in T^*$ such that for every initial marking $m \models C$ and reachable marking m' there is a sequence $\varrho \in \varrho_1^* \dots \varrho_k^*$ such that $m \stackrel{\varrho}{\Rightarrow} m'$. In short, all reachable markings can be reached by simple sequences, belonging to the language: $\varrho_1^* \dots \varrho_k^*$. Last but not least, the authors stated in Theorem XI.2 from [25] that a net is flat if and only if its reachability set is Presburger-definable:

Theorem XI.2 ***(from [25])***
The class of flatable VAS coincides with the class of Presburger VAS.

As a consequence, `FAST`'s algorithm terminates when its input is Presburger-definable. We show in Theorem 5.1 that given a parametric E-abstraction equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$, the silent reachability sets for both nets N_1 and N_2 with their coherency constraints C_1 and C_2 are indeed Presburger-definable—we can even provide the expected formulas. Yet, our computation is complete only if the candidate reduction rule is a parametric E-abstraction equivalence (then, we are able to compute the τ_C^* relation), otherwise `FAST`, and therefore our procedure too, may not terminate.

Theorem 5.1. (Silent State Spaces are Presburger-Definable)
Given a parametric E-abstraction equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$, the silent reachability set $R_\tau(N_1, C_1)$ is Presburger-definable.

Proof:
We prove only the result for (N_1, C_1), the proof for (N_2, C_2) is similar since $\approxeq$ is a symmetric relation. We first propose an expression that computes $R_\tau(N_1, m_1)$ for any marking m_1 satisfying

C_1. Consider an initial marking m_1 in C_1. From condition (S1) (solvability of E), there exists a compatible marking m_2 satisfying C_2, meaning $m_1 \langle C_1 E C_2 \rangle m_2$ holds. Take a silent sequence $m_1 \overset{\epsilon}{\Rightarrow} m_1'$. From condition (S2) (silent stability), we have $m_1' \equiv_E m_2$. Hence, $R_\tau(N_1, m_1) \subseteq \{m_1' \mid \exists m_2 \,.\, C_2(m_2) \wedge \tilde{E}(m_1, m_2) \wedge \tilde{E}(m_1', m_2)\}$. Conversely, we show that all m_1' solution of $\tilde{E}(m_1', m_2)$ are reachable from m_1. Take m_1' such that $m_1' \equiv_E m_2$. Since we have $m_2 \overset{\epsilon}{\Rightarrow} m_2$, by condition (S3) we must have $m_1 \overset{\epsilon}{\Rightarrow} m_1'$. And finally we obtain $R_\tau(N_1, m_1) = \{m_1' \mid m_1' \models \exists \boldsymbol{p_1}, \boldsymbol{p_2} \,.\, \underline{m_1}(\boldsymbol{p_1}) \wedge C_2(\boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p_1'}, \boldsymbol{p_2})\}$.

We can generalize this reachability set for all coherent markings satisfying C_1. We first recall its definition, $R_\tau(N_1, C_1) = \{m_1' \mid \exists m_1 \,.\, m_1 \models C_1 \wedge m_1 \overset{\epsilon}{\Rightarrow} m_1'\}$. From condition (S1), we can rewrite this set as $\{m_1' \mid \exists m_1, m_2 \,.\, m_1 \langle C_1 E C_2 \rangle m_2 \wedge m_1 \overset{\epsilon}{\Rightarrow} m_1'\}$ without losing any marking. Finally, thanks to the previous result we get $R_\tau(N_1, C_1) = \{m_1' \mid m_1' \models P\}$ with $P = \exists \boldsymbol{p_1}, \boldsymbol{p_2} \,.\, \langle C_1 E C_2 \rangle(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p_1'}, \boldsymbol{p_2})$ a Presburger formula. Because of the E-abstraction equivalence, (S1) holds in both directions, which gives $\forall \boldsymbol{p_2} \,.\, C_2(\boldsymbol{p_2}) \implies \exists \boldsymbol{p_1} \,.\, \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge C_1(\boldsymbol{p_1})$. Hence, P can be simplified into $\exists \boldsymbol{p_2} \,.\, C_2(\boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p_1'}, \boldsymbol{p_2})$.

Note that this expression of $R_\tau(N, C)$ relies on the fact that the equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$ already holds. Thus, we cannot conclude that a candidate rule is an E-abstraction equivalence using this formula at once without the extra validation of `FAST`. □

Verifying FAST Results.

We have shown that `FAST` terminates in case of a correct parametric E-abstraction. We now show that it is possible to check that the predicates $\tau^*_{C_1}$ and $\tau^*_{C_2}$, computed from the result of `FAST` (see Theorem 5.1) are indeed correct.

Assume τ^*_C is, according to `FAST`, equivalent to the language $\varrho_1^* \dots \varrho_n^*$ with $\varrho_i \in T^*$. We encode this language with the following Presburger predicate (similar to the one presented in [13]), which uses the formulas $H(\sigma^{k_i})$ and $\Delta(\sigma^{k_i})$ defined below:

$$\tau^*_C(\boldsymbol{p^1}, \boldsymbol{p^{n+1}}) \triangleq \exists k_1 \dots k_n, \boldsymbol{p^2} \dots \boldsymbol{p^{n-1}} \,.\, \bigwedge_{i \in 1..n} \Big((\boldsymbol{p^i} \geqslant H(\varrho_i^{k_i})) \wedge \Delta(\varrho_i^{k_i})(\boldsymbol{p^i}, \boldsymbol{p^{i+1}}) \Big) \tag{12}$$

This definition introduces acceleration variables k_i, encoding the number of times we fire the sequence ϱ_i. The hurdle and delta of the sequence of transitions ϱ_i^k, which depends on k, are written $H(\sigma^{k_i})$ and $\Delta(\sigma^{k_i})$, respectively. Their formulas are given in Equations (15) and (16) below. Let us explain how we obtain them.

First, we define the notion of hurdle $H(\varrho)$ and delta $\Delta(\varrho)$ of an arbitrary sequence ϱ, such that $m \overset{\varrho}{\Rightarrow} m'$ holds if and only if (1) $m \geqslant H(\varrho)$ (the sequence ϱ is fireable), and (2) $m' = m + \Delta(\varrho)$. This is an extension of the hurdle and delta of a single transition t, already used in Formulas (4) and (5). The definition of H and Δ is inductive:

$$H(\epsilon) \triangleq \mathbf{0},\ H(t) \triangleq \mathrm{Pre}(t) \ \text{ and } \ H(\varrho_1.\varrho_2) \triangleq \max\left(H(\varrho_1), H(\varrho_2) - \Delta(\varrho_1)\right) \tag{13}$$

$$\Delta(\epsilon) \triangleq \mathbf{0},\ \Delta(t) \triangleq \mathrm{Post}(t) - \mathrm{Pre}(t) \ \text{ and } \ \Delta(\varrho_1.\varrho_2) \triangleq \Delta(\varrho_1) + \Delta(\varrho_2) \tag{14}$$

where $\max$ is the component-wise max operator. The careful reader will check by herself that the definitions of $H(\varrho_1.\varrho_2)$ and $\Delta(\varrho_1.\varrho_2)$ do not depend on the way the sequence $\varrho_1.\varrho_2$ is split.

From these, we are able to characterize a necessary and sufficient condition for firing the sequence ϱ^k, meaning firing the same sequence k times. Given $\Delta(\varrho)$, a place p with a negative displacement (say $-d$) means that d tokens are consumed each time we fire ϱ. Hence, we should budget d tokens in p for each new iteration, and this suffices to enable the $k-1$ more iterations following the first transition ϱ. Therefore, we have $m \xRightarrow{\varrho^k} m'$ if and only if (1) $m \models m \geqslant \mathbb{1}_{>0}(k) \times (H(\varrho) + (k-1) \times \max(\mathbf{0}, -\Delta(\varrho)))$, with $\mathbb{1}_{>0}(k) = 1$ if and only if $k > 0$, and 0 otherwise, and (2) $m' = m + k \times \Delta(\varrho)$. Concerning the token displacement of this sequence ϱ^k, it is k times the one of the non-accelerated sequence ϱ. Equivalently, if we denote by m^+ the "positive" part of a mapping Δ, such that $\Delta^+(p) \triangleq 0$ when $\Delta(p) \leqslant 0$ and $\Delta^+(p) \triangleq \Delta(p)$ when $\Delta(p) > 0$, we get:

$$H(\varrho^k) \triangleq \mathbb{1}_{>0}(k) \times (H(\varrho) + (k-1) \times (-\Delta(\varrho))^+) \tag{15}$$

$$\Delta(\varrho^k) \triangleq k \times \Delta(\varrho) \tag{16}$$

Finally, given a parametric rule $(N_1, C_1) >_E (N_2, C_2)$ we can now check that the reachability expression $\tau^*_{C_1}$ provided by `FAST`, and encoded as explained above, corresponds to the solutions of $\exists \boldsymbol{p_2} \,.\, \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2})$ using the following additional SMT query:

$$\forall \boldsymbol{p_1}, \boldsymbol{p'_1} \,.\, C_1(\boldsymbol{p_1}) \implies (\exists \boldsymbol{p_2} \,.\, \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p'_1}, \boldsymbol{p_2}) \iff \tau^*_{C_1}(\boldsymbol{p_1}, \boldsymbol{p'_1})) \tag{17}$$

(and similarly for $\tau^*_{C_2}$).

Once the equivalence (17) above has been validated by a solver, it is in practice way more efficient to use the formula $(\exists \boldsymbol{p_2} \,.\, \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p'_1}, \boldsymbol{p_2}))$ inside the core requirements, rather than the formula $\tau^*_{C_1}(\boldsymbol{p_1}, \boldsymbol{p'_1})$ given by `FAST`, since the latter introduces many new acceleration variables.

6. Decidability

Even if our method may not terminate, since `FAST` is only a semi-decision procedure, we can prove that checking the correctness of parametric E-abstraction is decidable.

Theorem 6.1. (Checking Parametric E-abstraction is Decidable)
Given two nets N_1, N_2 and constraints C_1, C_2 expressed as Presburger formulas. The problem of deciding whether the statement $(N_1, C_1) \approxeq_E (N_2, C_2)$ holds is decidable.

Proof:
We proved in Theorems 4.8 and 4.9 that the statement $(N_1, C_1) \approxeq_E (N_2, C_2)$ holds if and only if (Core 0) is valid for both nets (N_1, C_1) and (N_2, C_2) and core requirements (Core 1), (Core 2), and (Core 3) are valid (in both ways). Furthermore, checking the truth of Presburger formulas is decidable [30].

We are left to prove that we can construct these formulas. The crux relies on the computation of predicates $\tau^*_{C_1}$ and $\tau^*_{C_2}$. We proved in Theorem 5.1 a necessary condition to have a correct equivalence, that is, $R_\tau(N_1, C_1)$ and $R_\tau(N_2, C_2)$ must be Presburger-definable. The problem of deciding

if the reachability set of a general Petri net from an initial Presburger set of markings is Presburger (equivalently semilinear [31]) is decidable [32, 33]. Then, if either $R_\tau(N_1, C_1)$ or $R_\tau(N_2, C_2)$ is not Presburger-definable we can assert that the equivalence does not hold; without constructing the core requirements. Otherwise, the net is flat [25]; and computing $\tau^*_{C_1}$ and $\tau^*_{C_2}$ is also decidable [34].

Hence, we proposed a theoretical procedure to answer the problem of deciding a parametric E-equivalence holds, where all steps are decidable. □

7. Generalizing equivalence rules

In this section we discuss some results related with the *genericity* and *generalisability* of our abstraction rules. We consider several "dimensions" in which a rule can be generalized. A first dimension is related with the parametricity of the initial marking, which is taken into account by our use of a parametric equivalence, $\approxeq$ instead of $\equiv$, see Theorem 3.4. Next, we show that we can infer an infinite number of equivalences from a single abstraction rule using compositionality, transitivity, and structural modifications involving labels. Therefore, each abstraction law can be interpreted as a schema for several equivalence rules.

Definition 7.1. (Transition Operations)
Given a Petri net $N \triangleq (P, T, \mathrm{Pre}, \mathrm{Post})$ and its labeling function $l : T \to \Sigma \cup \{\tau\}$, we define two operations: T^-, for removing, and T^+, for duplicating transitions. Let a and b be labels in Σ.

- $T^-(a)$ is a net $(P, T', \mathrm{Pre}', \mathrm{Post}')$, where $T' \triangleq T \setminus l^{-1}(a)$, and Pre' (resp. Post') is the projection of Pre (resp. Post) to the domain T'.
- $T^+(a, b)$ is a net $(P, T', \mathrm{Pre}', \mathrm{Post}')$, where T' is a subset of $T \times \{0, 1\}$ defined by $T' \triangleq T \times \{0\} \cup l^{-1}(a) \times \{1\}$. Additionally, we define $\mathrm{Pre}'(t, i) \triangleq \mathrm{Pre}(t)$ and $\mathrm{Post}'(t, i) \triangleq \mathrm{Post}(t)$ for all $t \in T$ and $i \in \{0, 1\}$. Finally, the labeling function l' is defined with $l'(t, 0) \triangleq l(t)$ and $l'(t, 1) = b$ for all $t \in T$.

The operation $T^-(a)$ removes transitions labeled by a, while $T^+(a, b)$ duplicates all transitions labeled by a and labels the copies with b. We illustrated T^+ in the nets of rule (MAGIC), in Fig. 2, where the "dashed" transition c' can be interpreted has the result of applying operation $T^+(c, c')$. Note that these operations only involve labeled transitions. Silent transitions are kept untouched—up-to some injection.

Theorem 7.2. (Preservation by Transition Operations)
Assume we have a parametric E-equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$, a and b are labels in Σ. Then,

- $T_i^-(a)$ and $T_i^+(a, b)$ satisfy the coherency constraint C_i, for $i = 1, 2$.
- $(T_1^-(a), C_1) \approxeq_E (T_2^-(a), C_2)$.
- $(T_1^+(a, b), C_1) \approxeq_E (T_2^+(a, b), C_2)$.

where T_i^-, T_i^+ is (respectively) the operation T^-, T^+ on N_i.

Proof:
We assume $(N_1, C_1) \approxeq_E (N_2, C_2)$ (i) holds, which implies that N_1 satisfies the coherency constraint C_1 (resp., N_2 satisfies C_2). For each operation T^-, T^+, we show that the transformed nets N_1' and N_2' still satisfy the coherency constraints and that the conditions (S1), (S2), (S3) of definition 3.3 still hold. Conditions (S1) and (S2) do not involve labeled transitions, so they immediately hold in N_1' and N_2'. (S3) is proven by considering each operation separately.

- Case $T^-(a)$: N_1' (resp. N_2') is N_1 (resp. N_2) without transitions labeled by a. Assume $(N_1', m_1) \stackrel{\sigma}{\Rightarrow} (N_1', m_1')$ holds (hence, $a \notin \sigma$). From (i), for all markings m_2, m_2', such that $m_1 \langle C_1 E C_2 \rangle m_2 \wedge m_1' \equiv_E m_2'$, we have $(N_2, m_2) \stackrel{\sigma}{\Rightarrow} (N_2, m_2')$. Hence, $(N_2', m_2) \stackrel{\sigma}{\Rightarrow} (N_2', m_2')$ holds since $a \notin \sigma$.

- Case $T^+(a, b)$: N_1' (resp. N_2') is N_1 (resp. N_2) with transitions labeled by a duplicated and duplicates are labeled by b. Assume $(N_1', m_1) \stackrel{\sigma}{\Rightarrow} (N_1', m_1')$ holds. Let σ_a be $\sigma\{b \leftarrow a\}$. Then, we have $(N_1, m_1) \stackrel{\sigma_a}{\Rightarrow} (N_1, m_1')$. From (i), for all markings m_2, m_2', such that $m_1 \langle C_1 E C_2 \rangle m_2 \wedge m_1' \equiv_E m_2'$, we have $(N_2, m_2) \stackrel{\sigma_a}{\Rightarrow} (N_2, m_2')$. Then, $(N_2', m_2) \stackrel{\sigma_a}{\Rightarrow} (N_2', m_2')$ holds since transitions of N_2 are included in those of N_2'. In N_2', each transition labeled by a is identical to a twin transition labeled by b. Hence, any such transition can be freely replaced by its twin. Therefore, $(N_2', m_2) \stackrel{\sigma}{\Rightarrow} (N_2', m_2')$ also holds. This concludes the case.

The proof that net N_1' (resp. N_2') still satisfies the coherency constraint C_1 (resp. C_2) is also done by considering each operation separately, and is actually very similar to the above cases (we omit the details). The three conditions (S1), (S2), (S3) hold on N_1' and N_2', thus $(N_1', C_1) \approxeq_E (N_2', C_2)$ is shown. □

Finally, we recall a previous result from [19, 5] (Theorem 7.3), which states that equivalence rules can be combined together using synchronous composition, relabeling, and chaining. Note that, in order to avoid inconsistencies that could emerge if we inadvertently reuse the same variable in different reduction equations (variable escaping its scope), we require that conditions can be safely composed: the equivalence statements $(N_1, m_1) \equiv_E (N_2, m_2)$ and $(N_2, m_2) \equiv_{E'} (N_3, m_3)$ are *compatible* if and only if $P_1 \cap P_3 = P_2 \cap P_3$. We also rely on classical operations for relabeling a net, and for synchronous product, $N_1 \parallel N_2$, which are defined in [5] for instance.

Theorem 7.3. (E-Equivalence is a Congruence [19, 5])
Assume we have two compatible equivalence statements $(N_1, m_1) \equiv_E (N_2, m_2)$ and $(N_2, m_2) \equiv_{E'} (N_3, m_3)$, and that M is a Petri net such that $N_1 \parallel M$ and $N_2 \parallel M$ are defined, then

- $(N_1, m_1) \parallel (M, m) \equiv_E (N_2, m_2) \parallel (M, m)$.
- $(N_1, m_1) \equiv_{\exists P_2 \setminus (P_1 \cup P_3).E \wedge E'} (N_3, m_3)$.
- $(N_1[a/b], m_1) \equiv_E (N_2[a/b], m_2)$ for any $a \in \Sigma$ and $b \in \Sigma \cup \{\tau\}$.

8. Checking the state space partition

We finally propose to check whether a statement provides a state space partition—that is not entailed by the parametric E-equivalence—since the relation is symmetric—by verifying two additional core requirements ((Core 4) and (Core 5)) on the reduced net N_2. It is important to emphasize that the state space partition is not a prerequisite for solving reachability problems mentioned in Sect. 2.3. Nevertheless, it is a requirement for some model counting methods, for which polyhedral reduction were initially developed [1, 2].

Given a marking m'_2 of the reduced net N_2, we define $\mathrm{Inv}_E(m'_2)$ as the set of markings of the initial net N_1 related to m'_2.

$$\mathrm{Inv}_E(m'_2) \triangleq \{m'_1 \mid m'_1 \equiv_E m'_2\} \tag{18}$$

Definition 8.1. (Equivalence Preserves Partitioning)
Given a parametric equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$, we say that it preserves partitioning if and only if the family of sets $S \triangleq \{\mathrm{Inv}_E(m'_2) \mid m'_2 \in R(N_2, C_2)\}$ is a partition of $R(N_1, C_1)$.

Note that, although the equivalence $\approxeq_E$ is symmetric, the partitioning property is not. That is, although $(N_1, C_1) \approxeq_E (N_2, C_2)$ and $(N_2, C_2) \approxeq_E (N_1, C_1)$ both hold, in general at most one of these relations preserves partitioning.

Here are the formulas that we use to check if an equivalence preserves partitioning:

$$\forall \boldsymbol{p_2}, \boldsymbol{p'_2} \,.\, C_2(\boldsymbol{p_2}) \wedge \tau(\boldsymbol{p_2}, \boldsymbol{p'_2}) \implies \mathrm{EQ}(\boldsymbol{p_2}, \boldsymbol{p'_2}) \tag{Core 4}$$

$$\forall \boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p'_2} \,.\, C_2(\boldsymbol{p_2}) \wedge C_2(\boldsymbol{p'_2}) \wedge \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p_2}) \wedge \tilde{E}(\boldsymbol{p_1}, \boldsymbol{p'_2}) \implies \mathrm{EQ}(\boldsymbol{p_2}, \boldsymbol{p'_2}) \tag{Core 5}$$

Theorem 8.2. (Checking State Space Partition)
The equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$ preserves partitioning if and only if (Core 4) and (Core 5) are valid.

Proof:
The set S, as defined in Definition 8.1 is a partition as a consequence of the following points:

No empty set in S. For any marking m'_2 in $R(N_2, C_2)$ there exists some marking m_2 and sequence σ such that $m_2 \models C_2$ and $m_2 \overset{\sigma}{\Rightarrow} m'_2$. By condition (S1) of the parametric E-abstraction, there is some marking m_1 such that $m_1 \langle C_1 E C_2 \rangle m_2$. From Theorem 3.4, we have $(N_1, m_1) \equiv_E (N_2, m_2)$. Now, by condition (A2) of the E-abstraction (Definition 2.1), there is some m'_1 such that $m'_1 \equiv_E m'_2$. Thus, $\mathrm{Inv}_E(m'_2)$ is not empty. This implies $\emptyset \notin S$.

The union $\cup_{A \in S} A$ is equal to $R(N_1, C_1)$. We prove both inclusions separately.

- Take a marking m'_1 in $R(N_1, C_1)$. As previously, we still have some markings $m_1 \models C_1$ and $m_2 \models C_2$ such that $(N_1, m_1) \equiv_E (N_2, m_2)$ and $m'_1 \in R(N_1, m_1)$ (by condition (S1) and Theorem 3.4). By condition (A2) of the E-abstraction, there is some marking m'_2 such that $m'_2 \in R(N_2, m_2)$ and $m'_1 \equiv_E m'_2$. Hence, there is some set $A \in S$ such that $m'_1 \in A$ and so $R(N_1, C_1) \subseteq \cup_{A \in S} A$.

- Now take a set A in S and a marking $m_1' \in A$. By construction, there is some marking m_2' in $R(N_2, C_2)$ such that $m_1' \equiv_E m_2'$. By condition (S1) and Theorem 3.4, there is $(N_1, m_1) \equiv_E (N_2, m_2)$ such that $m_1 \models C_1$, $m_2 \models C_2$ and $m_2' \in R(N_2, m_2)$. By condition (A2) of Definition 2.1 we have $m_1' \in R(N_1, m_1)$. Hence, $m_1' \in R(N_1, C_1)$ and so $\cup_{A \in S} A \subseteq R(N_1, C_1)$.

Pairwise disjoint. Take two different markings m_2' and m_2'' in $R(N_2, C_2)$. Since (N_2, C_2) is a coherent net, we can find some initial and intermediate markings such that $m_2^{(1)} \Rightarrow m_2^{(2)} \stackrel{\epsilon}{\Rightarrow} m_2'$ and $m_2^{(3)} \Rightarrow m_2^{(4)} \stackrel{\epsilon}{\Rightarrow} m_2''$ with $m_2^{(i)} \models C_2$ for all i in 1..4. And since (Core 4) is valid, we have $m_2' = m_2^{(2)}$ and $m_2'' = m_2^{(4)}$ (firing silent transitions from a coherent state do not change the marking). Hence, we get $m_2' \models C_2$ (i) and $m_2'' \models C_2$ (ii).

Now, we prove by contradiction that $\mathrm{Inv}_E(m_2') \cap \mathrm{Inv}_E(m_2'') = \emptyset$. Assume $\mathrm{Inv}_E(m_2') \cap \mathrm{Inv}_E(m_2'')$ is not empty and take a marking m_1 from it. Hence, $m_1 \equiv_E m_2'$ and $m_1 \equiv_E m_2''$. From (i) and (ii), and the hypothesis $m_2' \not\equiv m_2''$, we contradict the validity of (Core 5).

We are left to prove that the validity of (Core 4) and (Core 5) is a necessary condition to obtain such partition. Assume S is a partition of $R(N_1, C_1)$.

- Assume that (Core 4) is not valid. Then, there is a pair of different markings m_2, m_2' of N_2 such that $m_2 \models C_2$ and $m_2 \stackrel{\epsilon}{\Rightarrow} m_2'$. From condition (S1) there is some marking m_1 such that $m_1 \equiv_E m_2$, and by condition (S2) we also have $m_1 \equiv_E m_2'$. Then, there are two sets A and A' in S such that $m_1 \in A$ and $m_1 \in A'$, which contradicts that sets in S are pairwise disjoint.

- Assume that (Core 5) is not valid. Then, there are some markings m_1 of N_1 and m_2, m_2' of N_2 such that $m_2 \models C_2$, $m_2' \models C_2$, $m_1 \equiv_E m_2$, $m_1 \equiv_E m_2'$ and $m_2 \not\equiv m_2'$. By construction of S, we can find some sets A, A' in S, such that $m_1 \in A$ and $m_1 \in A'$, which also contradicts that sets in S are pairwise disjoint. □

From the proof of Theorem 8.2 we can derive an interesting characterization on the partitioning of parametric equivalences:

Lemma 8.3. A parametric equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$ preserves partitioning if and only if for all $m_2 \in R(N_2, C_2)$ and for all m_1, m_2' in $\mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$,

$$m_1 \equiv_E m_2 \wedge m_1 \equiv_E m_2' \implies m_2 = m_2'$$

Proof:
Assume the equivalence $(N_1, C_1) \approxeq_E (N_2, C_2)$ preserves partitioning. Take $m_2 \in R(N_2, C_2)$ and m_1, m_2 in $\mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$, such that $m_1 \equiv_E m_2 \wedge m_1 \equiv_E m_2'$. From $m_1 \equiv_E m_2$ and $m_2 \in R(N_2, C_2)$, we get $m_1 \in R(N_1, C_1)$ as a consequence of Definition 3.3. Then, similarly from $m_1 \equiv_E m_2'$, we get $m_2' \in R(N_2, C_2)$. Hence, $m_1 \in \mathrm{Inv}_E(m_2)$ and $m_1 \in \mathrm{Inv}_E(m_2')$. Since S (from Definition 8.1) is a partition, m_1 can only belong to exactly one of the sets that constitute S. Hence, necessarily $m_2 = m_2'$ (otherwise, two sets of S would contain m_1).

Conversely, let us assume that for all $m_2 \in R(N_2, C_2)$ and for all m_1, m_2 in $\mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$, $m_1 \equiv_E m_2 \wedge m_1 \equiv_E m_2' \implies m_2 = m_2'$ (i). Let $S \triangleq \{\mathrm{Inv}_E(m_2') \mid m_2' \in R(N_2, C_2)\}$. We have to show that S is a partition of $R(N_1, C_1)$.

No empty set in S. Same proof than in Theorem 8.2.

The union $\cup_{A\in S}A$ is equal to $R(N_1, C_1)$. Same proof than in Theorem 8.2.

Pairwise disjoint. Take two different markings m_2 and m_2' in $R(N_2, C_2)$. We get the expected result $\mathrm{Inv}_E(m_2) \cap \mathrm{Inv}_E(m_2') = \emptyset$ as an immediate consequence of implication (i).

Consequently, S is a partition of $R(N_1, C_1)$. □

Once an equivalence is shown to preserve partitioning, it can be freely composed with other partitioning equivalences. More precisely,

Lemma 8.4. If the equivalences $(N_1, C_1) \cong_E (N_2, C_2)$ and $(N_2, C_2) \cong_{E'} (N_3, C_3)$ are compatible and both preserve partitioning, then $(N_1, C_1) \cong_{\exists P_2 \setminus (P_1 \cup P_3)\, E \wedge E'} (N_3, C_3)$ holds and preserves partitioning too.

Proof:
We assume both equivalences $(N_1, C_1) \cong_E (N_2, C_2)$ and $(N_2, C_2) \cong_{E'} (N_3, C_3)$ are compatible and preserve partitioning. Let E'' be $\exists P_2 \setminus (P_1 \cup P_3)\, E \wedge E'$. By virtue of Theorem 7.3, $(N_1, C_1) \cong_{E''} (N_3, C_3)$ holds. It remains to show that it preserves partitioning. Following Lemma 8.3, let us take $m_3 \in R(N_3, C_3)$ and $m_1, m_3' \in \mathbb{N}^{P_1} \times \mathbb{N}^{P_3}$, and assume $m_1 \equiv_{E''} m_3$ and $m_1 \equiv_{E''} m_3'$ both hold. Notice that $m_1 \in R(N_1, C_1)$ and $m_3' \in R(N_3, C_3)$ (as already shown in the proof of Lemma 8.3). By definition of $m_1 \equiv_{E''} m_3$, and by removing the existential, we get $\exists m \in \mathbb{N}^{P_1 \cup P_2 \cup P_3}\ .\ m \models E \wedge E' \wedge \underline{m_1} \wedge \underline{m_3}$. By projecting m on P_2, we get a marking $m_2 \in \mathbb{N}^{P_2}$ such that $m_1 \equiv_E m_2$ and $m_2 \equiv_{E'} m_3$. Similarly, there exists $m_2' \in \mathbb{N}^{P_2}$ such that $m_1 \equiv_E m_2'$ and $m_2' \equiv_{E'} m_3'$. Since $(N_1, C_1) \cong_E (N_2, C_2)$ preserves partitioning, and by Lemma 8.3, we get $m_2 = m_2'$. Then again, since $(N_2, C_2) \cong_{E'} (N_3, C_3)$ preserves partitioning, and by Lemma 8.3, we get $m_3 = m_3'$, which is the expected result. □

9. Debugging reduction rules

An interesting feature of `Reductron` is to return which core requirements failed when a rule is unsound. It allows us to pinpoint the problematic condition and, if necessary, fix it. In this section, we give concrete examples of how this information can be used to debug a problematic equivalence rule. All our examples are obtained by mutating one of the reduction rules used in our toolchain.

Modified Equivalence Rule (CONCAT).

As we already mentioned, the equivalence rule (CONCAT) becomes incorrect by adding the "red dashed" transition with a label d. Our method will output that (N_1, C_1) is not a coherent net (i.e., (Core 0) fail). In fact, from a coherent state $m_1 \models C_1$, meaning $m_1(y_2) = 0$, it is not possible to fire the transition d and then reach a coherent state. This rule can be fixed by adding a τ transition from y_2 to y_1. This new rule is called (AGG) in our reduction framework.

An equivalent incorrect version of the rule (CONCAT), is the one in Fig.1, without transitions d, but with the coherency constraint $C_1 \triangleq \mathrm{True}$. Our tool, asserts that the expression $\tau^*_{C_1}$ returned by `FAST`

does not correspond to what is expected to obtain a correct equivalence (see Theorem 5.1). In fact, the tokens initially contained in y_2 cannot be transferred to y_1. As previously, one possible modification to ensure the equivalence is correct when $C_1 \triangleq \mathrm{True}$ is to add a silent transition from y_2 to y_1.

Modified Equivalence Rule (MAGIC)

Now we consider an incorrect version of the (MAGIC) rule depicted in Fig. 2, with $E \triangleq x = y_1 + y_2 + y_3$ (we forget y_4). Of course, again the preliminary check of the predicate $\tau^*_{C_1}$ obtained using `FAST` result fail. But, we also obtain that the requirement (Core 2) does not hold. A counter-example is $m_2 \triangleq (x = 1)$, $m_1 \triangleq (y_2 = 1) \wedge (y_1 + y_3 + y_4 = 0)$ and $m'_1 \triangleq (y'_4 = 1) \wedge (y'_1 + y'_2 + y'_3 = 0)$. This indicates that there is a problem with the definition of E.

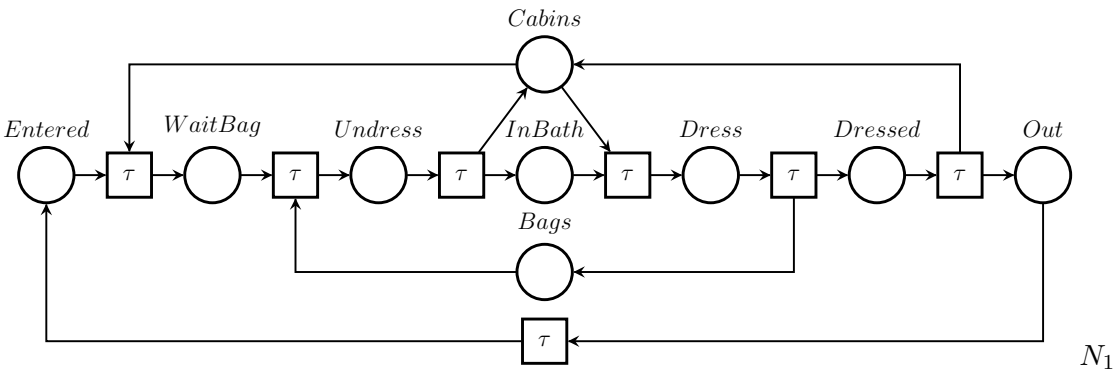
10. Experimental validation

We have implemented our automated procedure in a new tool called `Reductron`. The tool is open-source, under the GPLv3 license, and is freely available on GitHub [35]. The repository contains a subdirectory, `rules`, that provides examples of equivalence rules that can be checked using our approach. Each test contains two Petri nets, one for N_1 (called `initial.net`) and another for N_2 (called `reduced.net`), defined using the syntax of `Tina` [36]. These nets also include declarations for constraints, C_1 and C_2, and for the equation system E. Our list contains examples of laws that are implemented in `Tedd` and `SMPT`, such as rule (CONCAT) depicted in Fig. 1, but also some examples of unsound equivalences rules. For instance, we provide example (FAKE_CONCAT), which corresponds to the example of Fig. 1 with transition d added.

Table 1: Computation times (time in seconds).

Rule	`FAST`	`z3`	Total
(AGG)	0.09	0.18	0.28
(BUFFER)	0.45	0.44	0.89
(CONCAT)	0.08	0.14	0.23
(MAGIC)	0.17	0.32	0.51
(RED)	0.00	0.14	0.14
(SINK)	0.08	0.14	0.23
(SWIMMINGPOOL)	1.92	12.11	14.03

We performed some experimentation using `z3` [37] (version 4.8) as our target SMT solver, and `FAST` (version 2.1). We display the computation times obtained on all our examples of sound rules in Table 1. In each case, we give the total time, and how much time was spent in the two main steps of the procedure, for `FAST` and `z3`. Our results show that our reduction rules can all be checked in a few seconds.



$$C_1 \triangleq Cabins = 10 \wedge Out = 20 \wedge Bags = 15 \wedge$$
$$Entered + WaitingBag + Undress + Dresse + Inbath + Dressed = 0$$

$$E \triangleq \begin{cases} Cabins + Dress + Dressed + Undress + WaitBag = 10 \\ Dress + Dressed + Entered + InBath + Out + Undress + WaitBag = 20 \\ Bags + Dress + InBath + Undress = 15 \end{cases}$$

Figure 10: A Petri net modeling users in a swimming pool, see e.g. [22].

Although we focus on the automatic verification of abstraction laws, we have also tested our tool on moderate-sized nets, such as the swimming pool example given in Fig. 10. In this context, we use the fact that an equivalence of the form $(N, C) \approxeq_E (\emptyset, \mathrm{True})$, between N and a net containing an empty set of places, entails that the reachability set of (N, C) must be equal to the solution set of E. In this case, also, results are almost immediate.

These very good results depend largely on the continuous improvements made by SMT solvers. Indeed, we generate very large LIA formulas, with sometimes hundreds of quantified variables, and a moderate amount of quantifier alternation (formulas of the form $\forall\exists\forall$). For instance, experiments performed with older versions of `z3` (such as 4.4.1, October 2015) exhibit significantly degraded performances. We also rely on the very good performances exhibited by the tool `FAST`, which is essential in the implementation of `Reductron`.

11. Conclusion

This work aims to improve the safety of our polyhedral reduction framework using automated reasoning techniques. But the result we find the most interesting is the fact that it enhances our understanding of the theoretical underpinnings of polyhedral equivalence and its close relationship with the notion of flat nets. It also underlines the importance of coherency constraints, which takes a central role in our definition of a parametric version of polyhedral equivalence. We also hope that it helps better understand how to construct new reduction rules in the future.

There is still ample room to study polyhedral reduction. For instance, we are interested in characterizing Petri nets that are *fully reducible*, but where E is a "convex" predicate (to ensure that the equivalence defines a partition of the state space). This defines an interesting and non-trivial subset of flat nets.

Finally, we exhibited a concrete use case for the problem of deciding whether the state space of a given Petri net is Presburger-definable. This result can be found in two different works [32, 33], with proofs that do not easily translate into practical algorithms. We believe that it would be worthwhile to revisit this problem.

Acknowledgements

We would like to thanks Jérôme Leroux for his support during our experimentation with `FAST`.

References

[1] Berthomieu B, Le Botlan D, Dal Zilio S. Petri net Reductions for Counting Markings. In: Model Checking Software (SPIN), volume 10869 of *Lecture Notes in Computer Science*. Springer, 2018 doi:10.1007/978-3-319-94111-0_4.

[2] Berthomieu B, Le Botlan D, Dal Zilio S. Counting Petri net markings from reduction equations. *International Journal on Software Tools for Technology Transfer*, 2019. **22**(2):163–181. doi:10.1007/s10009-019-00519-1.

[3] Berthelot G, Lri-Iie. Checking properties of nets using transformations. In: Advances in Petri Nets (APN), volume 222 of *Lecture Notes in Computer Science*. Springer, 1985 doi:10.1007/BFb0016204.

[4] Berthelot G. Transformations and Decompositions of Nets. In: Petri Nets: Central Models and Their Properties (ACPN), volume 254 of *Lecture Notes in Computer Science*. Springer, 1987 doi:10.1007/ 978-3-540-47919-2_13.

[5] Amat N, Berthomieu B, Dal Zilio S. A Polyhedral Abstraction for Petri Nets and its Application to SMT-Based Model Checking. *Fundamenta Informaticae*, 2022. **187**(2-4):103–138. doi:10.3233/FI-222134.

[6] Amat N, Dal Zilio S, Le Botlan D. Leveraging polyhedral reductions for solving Petri net reachability problems. *International Journal on Software Tools for Technology Transfer*, 2023. **25**(1):95–114. doi:10.1007/s10009-022-00694-8.

[7] Besson F, Jensen T, Talpin JP. Polyhedral Analysis for Synchronous Languages. In: Static Analysis (SAS), volume 1694 of *Lecture Notes in Computer Science*. Springer, 1999 doi:10.1007/3-540-48294-6_4.

[8] Feautrier P. Automatic parallelization in the polytope model. In: The Data Parallel Programming Model, volume 1132 of *Lecture Notes in Computer Science*. Springer, 1996. doi:10.1007/3-540-61736-1_44.

[9] Thierry-Mieg Y, Poitrenaud D, Hamez A, Kordon F. Hierarchical Set Decision Diagrams and Regular Models. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 5505 of *Lecture Notes in Computer Science*. Springer, 2009 doi:10.1007/978-3-642-00768-2_1.

[10] LAAS-CNRS. Tina Toolbox, 2023. URL `http://projects.laas.fr/tina`.

[11] Amat N. SMPT: The Satisfiability Modulo Petri Nets Model Checker. An SMT-based model checker for Petri nets focused on reachability problems that takes advantage of polyhedral reduction., 2020. URL `https://github.com/nicolasAmat/SMPT`.

[12] Amat N, Dal Zilio S. SMPT: A Testbed for Reachabilty Methods in Generalized Petri Nets. In: Formal Methods (FM), volume 14000 of *Lecture Notes in Computer Science*. Springer, 2023 doi:10.1007/978-3-031-27481-7_25.

[13] Amat N, Dal Zilio S, Hujsa T. Property Directed Reachability for Generalized Petri Nets. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 13243 of *Lecture Notes in Computer Science*. Springer, 2022 doi:10.1007/978-3-030-99524-9_28.

[14] Amparore E, Berthomieu B, Ciardo G, Dal Zilio S, Gallà F, Hillah LM, Hulin-Hubard F, Jensen PG, Jezequel L, Kordon F, Le Botlan D, Liebke T, Meijer J, Miner A, Paviot-Adet E, Srba J, Thierry-Mieg Y, van Dijk T, Wolf K. Presentation of the 9th Edition of the Model Checking Contest. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS. Springer, 2019 doi:10.1007/978-3-662-58381-4_9.

[15] Esparza J, Nielsen M. Decidability issues for Petri nets. *BRICS Report Series*, 1994. **1**(8). doi:10.7146/brics.v1i8.21662.

[16] Esparza J. Decidability and complexity of Petri net problems — An introduction. In: Lectures on Petri Nets I: Basic Models (ACPN), volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998 doi:10.1007/3-540-65306-6_20.

[17] Hack MHT. Decidability Questions for Petri Nets. PhD Thesis, 1976.

[18] Hirshfeld Y. Petri nets and the equivalence problem. In: Computer Science Logic (CSL), volume 832 of *Lecture Notes in Computer Science*. Springer, 1994 doi:10.1007/BFb0049331.

[19] Amat N, Berthomieu B, Dal Zilio S. On the Combination of Polyhedral Abstraction and SMT-Based Model Checking for Petri Nets. In: Application and Theory of Petri Nets and Concurrency (PETRI NETS), volume 12734 of *Lecture Notes in Computer Science*. Springer, 2021 doi:10.1007/978-3-030-76983-3_9.

[20] Hujsa T, Berthomieu B, Dal Zilio S, Le Botlan D. Checking marking reachability with the state equation in Petri net subclasses. *CoRR*, 2020. **abs/2006.05600**.

[21] Hujsa T, Berthomieu B, Dal Zilio S, Le Botlan D. On the Petri Nets with a Single Shared Place and Beyond. *CoRR*, 2020. **abs/2005.04818**. `2005.04818`.

[22] Bérard B, Fribourg L. Reachability Analysis of (Timed) Petri Nets Using Real Arithmetic. In: Concurrency Theory (CONCUR), volume 1664 of *Lecture Notes in Computer Science*. Springer, 1999 doi:10.1007/3-540-48320-9_14.

[23] Bardin S, Finkel A, Leroux J, Petrucci L. FAST: Fast Acceleration of Symbolic Transition Systems. In: Computer Aided Verification (CAV), volume 2725 of *Lecture Notes in Computer Science*. Springer, 2003 doi:10.1007/978-3-540-45069-6_12.

[24] Bardin S, Finkel A, Leroux J, Petrucci L. FAST: acceleration from theory to practice. *International Journal on Software Tools for Technology Transfer*, 2008. **10**(5):401–424. doi:10.1007/s10009-008-0064-3.

[25] Leroux J. Presburger Vector Addition Systems. In: Logic in Computer Science (LICS). IEEE, 2013 doi:10.1109/LICS.2013.7.

[26] Amat N, Dal Zilio S, Le Botlan D. Automated Polyhedral Abstraction Proving. In: Application and Theory of Petri Nets and Concurrency (PETRI NETS), volume 13929 of *Lecture Notes in Computer Science*. Springer, 2023 doi:10.1007/978-3-031-33620-1_18.

[27] Amat N, Dal Zilio S, Le Botlan D. Accelerating the Computation of Dead and Concurrent Places Using Reductions. In: Model Checking Software (SPIN), volume 12864 of *Lecture Notes in Computer Science*. Springer, 2021 doi:10.1007/978-3-030-84629-9_3.

[28] Amat N, Dal Zilio S, Le Botlan D. Project and Conquer: Fast Quantifier Elimination for Checking Petri Nets Reachability. In: Verification, Model Checking, and Abstract Interpretation (VMCAI), Lecture Notes in Computer Science. Springer, 2024 doi:10.1007/978-3-031-50524-9_5.

[29] Barrett C, Fontaine P, Tinelli C. The SMT-LIB Standard: Version 2.6. Standard, University of Iowa, 2017.

[30] Presburger M, Jacquette D. On the completeness of a certain system of arithmetic of whole numbers in which addition occurs as the only operation. *History and Philosophy of Logic*, 1991. **12**(2):225–233. doi:10.1080/014453409108837187.

[31] Ginsburg S, Spanier E. Semigroups, Presburger formulas, and languages. *Pacific journal of Mathematics*, 1966. **16**(2):285–296. doi:10.2140/pjm.1966.16.285.

[32] Hauschildt D. Semilinearity of the reachability set is decidable for Petri nets. PhD Thesis, University of Hamburg, Germany, 1990.

[33] Lambert JL. Vector addition systems and semi-linearity. Université Paris-Nord. Centre Scientifique et Polytechnique [CSP], 1990.

[34] Finkel A, Leroux J. How to Compose Presburger-Accelerations: Applications to Broadcast Protocols. In: Foundations of Software Technology and Theoretical Computer Science (FSTTCS), volume 2556 of *Lecture Notes in Computer Science*. Springer, 2002 doi:10.1007/3-540-36206-1_14.

[35] Amat N. Reductron: The Polyhedral Abstraction Prover. A tool to automatically prove the correctness of polyhedral equivalences for Petri nets., 2023. URL `https://github.com/nicolasAmat/Reductron`.

[36] LAAS-CNRS. File formats of the Tina Toolbox. URL `http://projects.laas.fr/tina//manuals/formats.html/`.

[37] De Moura L, Bjørner N. Z3: An Efficient SMT Solver. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 4963 of *Lecture Notes in Computer Science*. Springer, 2008 doi:10.1007/978-3-540-78800-3_24.