

Petri Net Invariant Synthesis

Peter Chini¹ and Florian Furbach²

¹ TU Braunschweig, p.chini@tu-braunschweig.de

² Uppsala University, florian.furbach@it.uu.se

Abstract. We study the synthesis of inductive half spaces (IHS). These are linear inequalities that form inductive invariants for Petri nets, capable of disproving reachability or coverability. IHS generalize classic notions of invariants like traps or siphons. Their synthesis is desirable for disproving reachability or coverability where traditional invariants may fail.

We formulate a CEGAR-loop for the synthesis of IHS. The first step is to establish a structure theory of IHS. We analyze the space of IHS with methods from discrete mathematics and derive a linear constraint system closely over-approximating the space. To discard false positives, we provide an algorithm that decides whether a given half space is indeed inductive, a problem that we prove to be **coNP**-complete. We implemented the CEGAR-loop in the tool **INEQUALIZER** and our experiments show that it is competitive against state-of-the-art techniques.

1 Introduction

A major task of today’s program verification is to formulate and prove safety properties. Such a property describes the desirable and undesirable behavior of a program, often expressed in terms of safe and unsafe states. A safety property is satisfied if all executions of a program explore only safe states. Phrased differently, it is violated if an unsafe state is reachable via an execution. Testing reachability is usually a rather complex problem and often undecidable [7,27,52,50].

To restore decidability, the behavior of a program is often over-approximated. Intuitively, an over-approximation describes a property that holds for all reachable states but fails for unsafe states. Hence, over-approximations act like a separator between reachable and unsafe states and therefore provide a proof for the non-reachability of the latter. Computing over-approximations is often achieved by generating some type of invariant [17,28,4,49]. The challenge is to find a type that admits an efficient generation and that is expressive enough to separate reachable from unsafe states. Inductive invariants are a prominent example [3,9,25]. If an inductive invariant holds for some state, then it also holds for any successor after a step of an execution. Hence, if an inductive invariant is satisfied initially, it holds for all reachable states.

We generate inductive invariants for Petri nets, a well-established model of concurrent programs [44,42]. Here, safety verification is usually expressed in terms of the Petri net reachability or coverability problem. The former is known

to be ACKERMANN-complete [11,35,10,36], the latter is EXPSPACE-complete [46,37,6]. Despite the ongoing algorithmic development, in particular for coverability [30,29,53,22,47], computational requirements of solving both problems often exceeds practical limits. This has led to the development of classic Petri net invariants like *traps*, *siphons*, or *place invariants* [44] that may help to solve both problems more efficiently. Typically, these invariants are based on linear dependencies of places or transitions and can be synthesized easily by incorporating tools and solvers from linear programming.

The trade-off for the efficient synthesis of these classic invariants is that their expressiveness is limited and often not sufficient to prove non-reachability of a marking. We study *inductive half spaces* (IHS) [48,51], a type of invariants with increased expressiveness. These consist of a tuple (k, c) , where k is a vector over the places of the Petri net and c is an integer. The corresponding *half space* is a subset of the space of markings, containing all markings m that satisfy the inequality $k \cdot m \geq c$. It is called *inductive* if the markings that are in the half space do not leave it after firing a transition. Inductive half spaces generalize many of the classical Petri net invariants [51] and preserve their *linear nature*. However, the synthesis of IHS remained an open problem.

Our contribution is a method for the synthesis of inductive half spaces. More precise, we compute IHS that separate an initial marking m_0 from a final marking m_f , proving the latter non-reachable. This task is formalized in the *linear safety verification problem* LSV(R). Given m_0 and m_f , it asks for an IHS (k, c) such that $k \cdot m_0 \geq c$ and $k \cdot m_f < c$. The problem was first considered in [48] for continuous Petri nets. The synthesis of IHS is much easier in the continuous case. In fact, an entire subclass we call *non-trivial inductive half spaces* does not occur in this setting. So far, LSV(R) has not been considered in its full generality and its decidability is still unknown.

We provide a semi-decision procedure for LSV(R) using *counter example guided abstraction refinement* (CEGAR) [8], a state-of-the-art technique in program verification. We illustrate the approach in Fig. 1. Suppose we are given a Petri net N , an initial marking m_0 , and a marking m_f for which we want to disprove reachability from m_0 . Our approach attempts to synthesize an IHS that separates m_f from the reachable markings of N . It begins by constructing a formula ϕ of linear constraints from the given information and passes it to an SMT-solver. Roughly, ϕ describes necessary conditions for solutions of LSV(R). For each solution (k, c) of LSV(R), the vector k is a solution of ϕ . If the SMT-solver does not find a solution to ϕ , then no separating IHS exists. Otherwise we find a vector k of a half space candidate. We then determine whether there exists a $c \in \mathbb{Z}$ such that (k, c) is indeed inductive. In order to synthesize such a c , we developed a *constant generation algorithm* (CGA). If CGA is suc-

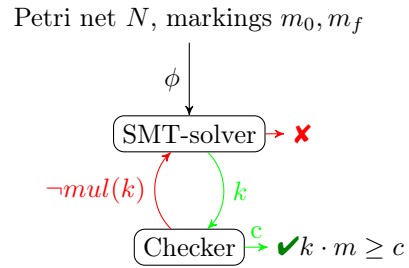


Fig. 1: The CEGAR loop.

cessful, we have found a separating IHS (k, c) . Otherwise, k does not admit a suitable constant c and we apply a refinement. We set $\phi = \phi \wedge \neg \text{mul}(k)$, where $\neg \text{mul}(k)$ is a linear constraint that excludes all multiples of k and repeat the above process. Note that the loop may not terminate. But if it does, we obtain an answer to LSV(R).

To realize the CEGAR-loop, we make the following main contributions.

- We develop a structure theory of inductive half spaces. It decomposes the space of IHS into *trivial* and *non-trivial* half spaces. While the synthesis of trivial IHS is simple, the synthesis of non-trivial ones is challenging. By employing techniques from discrete mathematics, we can determine necessary conditions for non-trivial IHS and construct the required formula ϕ .
- We present two algorithms: the *inductivity checker* (ICA) and the *constant generation algorithm* (CGA). The former determines whether a given half space is inductive, a problem that we prove coNP-complete. This answers an open question from [51]. ICA combines structural properties of IHS with dynamic programming. The algorithm CGA synthesizes a constant c for a solution k of ϕ . CGA is an instrumentation of ICA. As termination argument, it uses an interesting connection between IHS and the *Frobenius number*.
- We implemented the CEGAR-loop in the tool INEQUALIZER. Employing it, we disproved reachability and coverability for a benchmark of widely used concurrent programs. The results are compared to algorithms implemented in MIST [43] and show INEQUALIZER to be competitive.

Related Work The reachability problem of Petri nets is a central problem in theoretical computer science. Its complexity was finally resolved after 45 years and proven to be ACKERMANN-complete. The upper bound is due to Leroux and Schmitz [36]. The authors refined several classical algorithms for reachability like the one by Kosaraju [32], Mayr [38,39], and Lambert [33]. Hardness was first considered by Lipton [37]. He proved reachability EXPSPACE-hard. Czerwinski et al [10] improved the lower bound to non-elementary. A new result due to Leroux, Czerwinski, and Orlikowski [35,11] closes the gap completely.

Many safety verification tasks can be phrased in terms of coverability. The EXPSPACE-completeness of the problem was determined by Rackoff [46] and Lipton [37]. Despite this, efficient algorithms keep getting developed [30]. Modern approaches are based on forward or backward state space exploration [20,23,29,31,53]. A method that has drawn interest are so-called unfoldings [14,40,15,34]. Notably, Abdulla et al [1] solve coverability by constructing an unfolding that represents backwards reachable states. They analyze it using an SMT-formula.

Profiting from advances in SMT-solving, deriving program properties by constraint solving has become popular [26,1,13,25]. In [45], ranking functions are synthesized by solving linear inequalities. Synthesis methods for Petri nets involving SMT-solving often simplify the task by using continuous values. In [13], Esparza et al generate inductive invariants disproving co-linear properties. Sankaranarayanan et al [48] synthesize IHS over continuous Petri nets. Compared to the

latter, we generate a larger class of invariants: *non-trivial* IHS do not occur in [48] but can be necessary for discrete nets (see Fig. 2). The structure of IHS was first considered by Triebel and Sürmeli [51]. The authors show that IHS generalize notions like traps, siphons, and place invariants.

Outline In Section 2, we introduce the necessary notions around Petri nets. The structure of IHS is examined in Section 3. In Section 4, we formulate the SMT-formula in the CEGAR loop. The algorithms ICA and CGA are given in Section 5. Experimental results are presented in Section 6. For brevity, we omit a number of formal proofs. They can be found in the appendix.

2 Linear Safety Verification

We introduce the linear safety verification problems for Petri nets. They formalize the question of whether there exists an inductive half space which disproves reachability or coverability of a certain marking. To this end, we formally introduce half spaces and the necessary notions around Petri nets.

Petri Nets A *Petri net* is a tuple $N = (P, T, F)$, where P is a finite set of *places*, T is a finite set of *transitions*, and $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is a *flow function*. We denote the number of places $|P|$ by n . The places are numbered. For convenience, we use a place p_i and their numeric value i interchangeably: Given a vector $x \in \mathbb{N}^n$, we denote its i -th component as both $x(i)$ and $x(p_i)$. For a transition $t \in T$, we define vectors $t^-, t^+ \in \mathbb{N}^n$. The i -th component of t^- , with $p_i \in P$, is defined to be $F(p_i, t)$, written $t^-(i) = t^-(p_i) := F(p_i, t)$. Similarly, $t^+(i) = t^+(p_i) := F(t, p_i)$. The vector t^Δ captures the difference $t^\Delta := t^+ - t^-$.

The semantics of a Petri net N is defined in terms of markings. A *marking* m is a vector in \mathbb{N}^n . Intuitively, it puts a number of *tokens* in each place. A marking is said to *enable* a transition t if $m(p) \geq t^-(p)$ for each place $p \in P$, written $m \geq t^-$. The set of all markings that enable t is called the *activation space* of t and is denoted by $\text{Act}(t)$. Note that $\text{Act}(t) = \{t^- + v \mid v \geq 0\}$. If $m \in \text{Act}(t)$, then t can be *fired*, resulting in the new marking $m' = m + t^\Delta$. This constitutes the *firing relation*, written as $m[t]m'$. We lift the relation to sequences of transitions $\sigma = t_1 \dots t_k \in T^*$ where convenient, writing $m[\sigma]m'$. A marking m_f is called *reachable* from a marking m_0 if there is a sequence of transitions σ such that $m_0[\sigma]m_f$. We use $\text{post}^*(m_0)$ to denote the markings reachable from m_0 and $\text{pre}^*(m_f)$ are the markings from which m_f is reachable. The *upward closure* of m_f is $\uparrow m_f = \{m \in \mathbb{N}^n \mid m \geq m_f\}$. A marking m_f is *coverable* from m_0 , if there is a sequence of transitions σ and an $m \in \uparrow m_f$ such that $m_0[\sigma]m$.

(Inductive) Half Spaces We describe sets of markings by means of half spaces. Let $N = (P, T, F)$ be a Petri net, $k \in \mathbb{Z}^n$ a vector, and $c \in \mathbb{Z}$ an integer. The *half space* defined by k and c is $\text{Sol}(k, c) = \{m \in \mathbb{Z}^n \mid k \cdot m \geq c\}$. Here, $k \cdot m = \sum_{p \in P} k(p) \cdot m(p)$ is the usual scalar product. We also refer to the tuple

(k, c) as half space. Note that we could also define half spaces via $k \cdot m \leq c$. This is of course equivalent since $k \cdot m \geq c$ if and only if $-k \cdot M \leq -c$. We are interested in half spaces that are inductive in the sense that they cannot be left by firing transitions. A half space (k, c) is *t-inductive* if for any $m \in \text{Act}(t) \cap \text{Sol}(k, c)$ we have $m + t^\Delta \in \text{Sol}(k, c)$. A half space (k, c) is *inductive* if it is *t-inductive* for all $t \in T$. We use IHS as a shorthand for inductive half space.

A half space (k, c) is not *t-inductive* if and only if it contains a marking m with $k \cdot m \geq c$ that enables t , i.e. $m \geq t^-$, and from which we leave the half space by firing t : $k \cdot (m + t^\Delta) < c$. Since $m \geq t^-$ if and only if there is an $x \in \mathbb{N}^n$ with $m = t^- + x$, we can state inductivity in terms of an infeasibility requirement:

Theorem 1. *A half space (k, c) is t-inductive iff there is no vector $x \in \mathbb{N}^n$ with*

$$c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta.$$

Theorem 1 provides a way of disproving inductivity of a half space by finding a suitable vector x . It is a key ingredient of our further development.

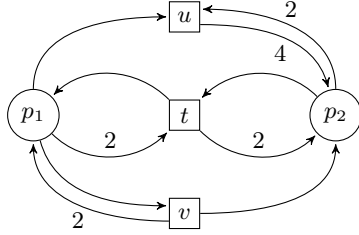


Fig. 2: Petri net with places p_1, p_2 , transitions u, t, v . Edges are entries of the flow function F . We omit the label if it is 1.

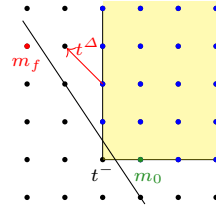


Fig. 3: Geometric interpretation of the half space (k, c) in \mathbb{Z}^2 . It is inductive and separates m_0 from m_f .

Example We provide some geometric intuition. Consider the Petri net in Fig. 2. Focus on transition t . The vectors describing t are $t^- = (2, 1)$ (incoming edges), $t^+ = (1, 2)$ (outgoing edges), and $t^\Delta = (-1, 1)$. The activation space of t is $\text{Act}(t) = \{(2, 1) + (x, y) \mid x, y \in \mathbb{N}\}$. It is visualized by the yellow area in Fig. 3. Let $m_0 = (3, 1)$ and $m_f = (0, 4)$. Consider the half space defined by $k = (3, 2)$ and $c = 9$. In Fig. 3, it is indicated by the diagonal line $k \cdot x = c, x \in \mathbb{R}^2$. The set of integer vectors above it is $\text{Sol}(k, c)$. Clearly, $m_0 \in \text{Sol}(k, c)$ and $m_f \notin \text{Sol}(k, c)$, the half space separates the markings. The markings in $\text{Act}(t) \cap \text{Sol}(k, c)$ are colored blue in Fig. 3. The half space is *t-inductive*: if $m \in \text{Act}(t) \cap \text{Sol}(k, c)$, firing t does not lead to a marking below the line. As we will see in Section 3, (k, c) is also *u* and *v*-inductive. Hence, it proves non-reachability of m_f from m_0 .

Linear Safety Verification Our goal is to find inductive half spaces that disprove reachability or coverability. Given a Petri net N and two markings m_0, m_f , we study two corresponding algorithmic problems: the *linear safety verification problem* LSV(R) for reachability and its coverability variant LSV(C).

LSV(R): Is there an IHS (k, c) with $m_0 \in \text{Sol}(k, c)$ and $m_f \notin \text{Sol}(k, c)$?

LSV(C): Is there an IHS (k, c) with $m_0 \in \text{Sol}(k, c)$ and $\uparrow m_f \cap \text{Sol}(k, c) = \emptyset$?

The reader familiar with separability will note that disproving reachability of m_f from m_0 amounts to finding a separator between $\text{post}^*(m_0)$ and $\text{pre}^*(m_f)$. A separator is a set $S \subseteq \mathbb{N}^n$ so that $\text{post}^*(m_0) \subseteq S$ and $S \cap \text{pre}^*(m_f) = \emptyset$. The difference between separability and linear safety verification is that separators are neither required to be half spaces nor required to be inductive.

The choice for half spaces and inductivity is motivated by the constraint-based approach to safety verification that we pursue. Half spaces can be given in terms of (k, c) , a format that is computable by a solver. Inductivity yields a local check for separation. Indeed, if (k, c) is inductive and $m_0 \in \text{Sol}(k, c)$, we already have $\text{post}^*(m_0) \subseteq \text{Sol}(k, c)$. Similarly, if (k, c) is inductive and $m_f \notin \text{Sol}(k, c)$, then $\text{pre}^*(m_f) \cap \text{Sol}(k, c) = \emptyset$. Hence, $\text{Sol}(k, c)$ is indeed a separator. But there are separators that are neither half spaces nor inductive. To see the latter, consider a transition that is not enabled in $\text{post}^*(m_0)$ but in a separator S . Firing the transition may lead to a marking outside of S and violate inductivity.

While reachability and coverability are decidable for Petri nets, decidability of LSV(R) and LSV(C) is unknown. Our approach semi-decides both problems.

3 Half Spaces

In order to synthesize inductive half spaces, we consider the structure of the space of IHS in more detail. Our goal is to derive a linear constraint system that closely approximates the structure of the space. The system can then be passed to an SMT-solver to synthesize candidates of half spaces.

Since IHS require inductivity for all transitions, their structure can be convoluted. Therefore, we do not immediately consider the space of all IHS. Instead, we first focus on half spaces that are inductive for a single transition. We derive linear constraints describing these half spaces. They are combined in Section 4 in order to obtain the desired SMT-formula for the space of all IHS.

The set of half spaces that are inductive for a given transition splits into two parts: the *trivial* half spaces and the *non-trivial* ones. We first focus on the former. Trivial half spaces were already described in [48,51]. They satisfy one of three conditions that immediately imply inductivity and can be easily synthesized. We provide a formal definition below.

The first condition for triviality describes the fact that the vector k and the transition t point into the same direction. The half space (k, c) is *oriented towards transition t* if $k \cdot t^\Delta \geq 0$. Since the scalar product provides information about the angle between k and t^Δ , the condition means that firing transition t moves a marking in the half space further away from the border. To give an

example, consider the half space (k, c) with $k = (3, 2)$ from Fig. 3. It is oriented towards transitions u and v . We have $u^\Delta = (-1, 2)$ and $v^\Delta = (1, 1)$, hence $k \cdot u^\Delta$ and $k \cdot v^\Delta$ are both non-negative. The half space is not oriented towards t since $t^\Delta = (-1, 1)$. Firing t means moving closer to the border of the half space.

It is easy to see that a half space which is oriented towards a transition t is actually t -inductive. This observation is a first step in the synthesis of IHS. In fact, note that generating a half space (k, c) that separates two markings m_0 and m_f and that is oriented towards t amounts to finding a solution (k, c) of the linear constraint system $k \cdot m_0 \geq c \wedge k \cdot m_f < c \wedge k \cdot t^\Delta \geq 0$.

The second condition for triviality uses the fact that for $k \geq 0$, the function $k \cdot m$ is monotone on markings. We call a half space (k, c) *monotone for transition t* if $k \geq 0$ and $k \cdot (t^- + t^\Delta) \geq c$. Note that with larger markings, $k \cdot m$ grows. This means if the smallest marking in the half space enabling t , namely t^- , stays within the half space after firing t , the same holds for all larger markings. The requirement is captured in the inequality $k \cdot (t^- + t^\Delta) \geq c$. Hence, monotone half spaces are inductive and can be synthesized as solutions of $k \geq 0 \wedge k \cdot (t^- + t^\Delta) \geq c$.

The last condition is dual to monotonicity. A half space (k, c) is *antitone for transition t* if $k \leq 0$ and $k \cdot t^- < c$. The latter requirement describes that t^- does not lie in the half space. Since $k \leq 0$ this means that $\text{Act}(t) \cap \text{Sol}(k, c) = \emptyset$. Hence, antitone half spaces are inductive. Moreover, they can be generated as solutions to the linear constraints $k \leq 0 \wedge k \cdot t^- < c$. We summarize:

Definition 1. *A half space (k, c) is trivial wrt. t if one of the following holds: (k, c) is oriented towards t , (k, c) is monotone for t , or (k, c) is antitone for t .*

Theorem 2. ([51]) *If (k, c) is trivial with respect to t then it is t -inductive.*

Non-trivial half spaces are not automatically t -inductive. As an example, consider the half space from Fig. 3. Recall that $k = (3, 2)$ and $c = 9$. If we replace c by $c' = 8$, we get that (k, c') is a non-trivial half space that is not t -inductive. We have $k \cdot t^- = 8 = c'$ but $k \cdot (t^- + t^\Delta) = 7 < c'$. Hence, when firing t from t^- , we leave (k, c') . This has two implications. First, we need an algorithm to test whether a non-trivial half space is indeed inductive. Second, we cannot hope for a simple synthesis as for trivial half spaces. The former is resolved by the algorithm ICA which we show in Section 5. For the latter, we develop an independent structure theory in the subsequent section.

3.1 Non-Trivial Half Spaces

We consider half-spaces that are non-trivial but inductive. These are neither oriented towards the transition of interest, nor monotone, nor antitone. Our first insight is a structural theorem which strongly impacts the synthesis of non-trivial IHS. In fact, we show that a half space (k, c) which is not oriented towards a transition t but t -inductive cannot have positive and negative entries in k . This means we can restrict to $k \geq 0$ or $k \leq 0$ when synthesizing non-trivial IHS.

Theorem 3. *Let (k, c) be a half space that not oriented towards a transition t but t -inductive. Then, we have $k \geq 0$ or $k \leq 0$.*

The proof of the theorem relies on the notion of *syzygies* known from commutative algebra [24]. We adapt it to our setting. A *syzygy* of k is a vector $s \in \mathbb{Z}^n$ with $k \cdot s = 0$. This means that adding a syzygy to a marking m does not change the scalar product with k . We have $k \cdot m = k \cdot (m + s)$. Hence, if $m \in \text{Sol}(k, c)$, we get that $m + s \in \text{Sol}(k, c)$ for all syzygies s of k . We proceed with the proof.

Proof. Assume (k, c) is t -inductive and not oriented towards t but there are $i \neq j$ with $k(i) > 0$ and $k(j) < 0$. We show that (k, c) cannot be t -inductive which contradicts the assumption. The idea is as follows. We set $u(i) = \lceil \frac{c}{k(i)} \rceil$ and $u(\ell) = 0$ for $\ell \neq i$. Note that $u \in \text{Sol}(k, c)$. From u , we construct a vector $v \in \mathbb{Z}^n$ that lies in $\text{Sol}(k, c)$ but $v + t^\Delta \notin \text{Sol}(k, c)$. Note that v might not be a proper marking. By adding non-negative syzygies to v , we obtain a marking $m \in \text{Act}(t) \cap \text{Sol}(k, c)$ with $m + t^\Delta \notin \text{Sol}(k, c)$. Hence, (k, c) is not t -inductive.

The vector v is defined by $v = u + \lfloor \frac{c - k \cdot u}{k \cdot t^\Delta} \rfloor \cdot t^\Delta \in \mathbb{Z}^n$. Since (k, c) is not oriented towards t , we have $k \cdot t^\Delta < 0$. Hence, v is well-defined. By $\lfloor x \rfloor \geq x - 1$, we obtain the following inequality showing that $v \in \text{Sol}(k, c)$:

$$k \cdot v \geq k \cdot u + \left(\frac{c - k \cdot u}{k \cdot t^\Delta} - 1 \right) \cdot k \cdot t^\Delta = c - k \cdot t^\Delta \geq c.$$

Similarly, by $\lfloor x \rfloor \leq x$, we obtain that $v + t^\Delta \notin \text{Sol}(k, c)$:

$$k \cdot (v + t^\Delta) \leq k \cdot u + \frac{c - k \cdot u}{k \cdot t^\Delta} \cdot k \cdot t^\Delta + k \cdot t^\Delta = c + k \cdot t^\Delta < c.$$

Note that v is not yet a counter example for t -inductivity. Indeed, we cannot ensure that v is a marking that enables t . But we can construct such a marking by adding syzygies to v . For a place $p \in P$ let e_p denote the p -th unit vector. This means $e_p(p) = 1$ and $e_p(q) = 0$ for $q \neq p$. For any place p , we construct a syzygy s_p defined as follows. If $k(p) > 0$, we set $s_p = -k(j) \cdot e_p + k(p) \cdot e_j$. If $k(p) < 0$, we set $s_p = -k(p) \cdot e_i + k(i) \cdot e_p$. For the case $k(p) = 0$, we simply set $s_p = e_p$. Note that for all places p , we have $s_p \geq 0$ and $k \cdot s_p = 0$.

The syzygies s_p allow for adding non-negative values to each component of v without changing the scalar product with k . Hence, there exist $\mu_p \in \mathbb{N}$ such that $v + \sum_{p \in P} \mu_p \cdot s_p \geq t^-$. By setting $m = v + \sum_{p \in P} \mu_p \cdot s_p$, we get a marking in $\text{Act}(t)$ that satisfies $k \cdot m = k \cdot v \geq c$ and $k \cdot (m + t^\Delta) = k \cdot (v + t^\Delta) < c$. Hence, m contradicts t -inductivity of (k, c) and we obtain the desired contradiction. \square

The theorem allows us to assume $k \geq 0$ or $k \leq 0$ when synthesizing non-trivial inductive half spaces. However, we cannot hope for a compact linear constraint system like we have for trivial half spaces. The reason is as follows. Assume we have a constraint system $L(k, c)$ of polynomial size describing the space of t -inductive non-trivial half spaces. Each solution of $L(k, c)$ corresponds to such a half space and vice versa. We can then decide, in polynomial time, whether a given half space (k, c) is t -inductive. Indeed, an algorithm would first decide whether (k, c) is trivial or non-trivial. In the former case, t -inductivity immediately follows. In the latter case, the algorithm checks if (k, c) is a solution

to $L(k, c)$. All these steps can clearly be carried out in polynomial time. However, the algorithm would contradict the coNP-hardness of checking t -inductivity, which we prove in Section 5. Hence, the system $L(k, c)$ of polynomial size cannot exist.

Although a concise constraint system for the space of non-trivial IHS seems out of reach, we can give a close linear approximation. To this end, we derive two necessary conditions for non-trivial IHS that can be formulated in terms of linear constraints. The first one is given in the following lemma. The proof follows from Theorem 3 and from inverting the constraints for trivial half spaces.

Lemma 1. *A t -inductive half space (k, c) that is non-trivial for t either satisfies (a) $k \geq 0$ and $k \cdot t^- < c - k \cdot t^\Delta$ or (b) $k \leq 0$ and $k \cdot t^- \geq c$.*

The lemma provides geometric intuition to separate non-trivial from trivial half spaces. If (k, c) is non-trivial, $\text{Sol}(k, t) \cap \text{Act}(t)$ is a strict non-empty subset of the activation space $\text{Act}(t)$. This stands in contrast to the trivial case. Here, (k, c) is either oriented towards t or the following holds. If (k, c) is monotone, we have $\text{Sol}(k, t) \cap \text{Act}(t) = \text{Act}(t)$ and if (k, c) is antitone, we have $\text{Sol}(k, t) \cap \text{Act}(t) = \emptyset$.

We employ Lemma 1 to derive a further necessary condition for non-trivial half spaces. It provides a lower bound for the absolute values of the vector k .

Lemma 2. *Let (k, c) be a t -inductive half space that is non-trivial for t . For any entry $k(i)$ of k , with $|k(i)|$ denoting its absolute value, we have:*

$$k(i) = 0 \vee |k(i)| \geq -k \cdot t^\Delta \quad (5)$$

The idea behind the lemma is the following. If the absolute value of an entry of k is too small then we can construct a vector $x \in \mathbb{N}^n$ such that $k \cdot x + k \cdot t^-$ lies between c and $c - k \cdot t^\Delta - 1$. This violates the condition stated in Theorem 1.

4 Generating Invariants

We combine the conditions from Section 3 to formulate a linear SMT-formula ϕ approximating the space of inductive half spaces. A solution to ϕ is a vector k that potentially forms an IHS. To keep the constraints in the formula linear, we cannot generate a corresponding constant c immediately. Instead, we replace c by bounds imposed by LSV(R) and LSV(C) and generate candidates for c in a second synthesis step with the algorithm CGA. The algorithm is given in Section 5.

Recall that in LSV(R), we are interested in finding an inductive half space (k, c) that separates an initial marking m_0 from a marking m_f . Phrased differently, we want $m_0 \in \text{Sol}(k, c)$ and $m_f \notin \text{Sol}(k, c)$. The former implies that $k \cdot m_0 \geq c$, the latter implies $k \cdot m_f < c$. The inequalities yield that $k \cdot m_0 > k \cdot m_f$ and impose two bounds on c , namely $c \in [k \cdot m_f + 1, k \cdot m_0]$. We apply the bounds

to the constraints obtained for trivial half spaces and derive the following condi-

$$\begin{aligned} \text{tions:} \quad & k \cdot m_0 > k \cdot m_f \quad (0) \quad k \leq 0 \wedge k \cdot t^- < k \cdot m_0 \quad (2) \\ & k \cdot t^\Delta \geq 0 \quad (1) \quad k \geq 0 \wedge k \cdot t^- > k \cdot m_f - k \cdot t^\Delta \quad (3) \end{aligned}$$

Each of the conditions (1), (2), and (3) models a type of trivial half spaces. For instance, (1) describes half spaces that are oriented towards t . Together with (0), we ensure t -inductivity for some c within the bounds. To describe non-trivial half spaces, we employ Theorem 3 and Lemma 2. We derive the following constraints:

$$k \geq 0 \vee k \leq 0 \quad (4) \quad \forall_i k(i) = 0 \vee |k(i)| \geq -k \cdot t^\Delta \quad (5)$$

We collect all the constraints in the SMT-formula ϕ_t in order to find the desired inductive half space. Note that the half space must be separating (0). Moreover, it is either trivial, so it satisfies one out of (1), (2), and (3), or it is non-trivial and satisfies (4) and (5). We construct the formula accordingly:

$$\phi_t := (0) \wedge ((1) \vee (2) \vee (3) \vee ((4) \wedge (5))).$$

As mentioned above, it is not possible to construct a linear constraint system of polynomial size that captures all t -inductive half spaces and yields k and c . However, ϕ_t is a tight approximation. In fact, its solutions are precisely those vectors k that can form a t -inductive half space which separates m_0 from m_f .

Lemma 3. *There exists a constant $c \in \mathbb{Z}$ such that (k, c) is a t -inductive half space with $k \cdot m_0 \geq c$ and $k \cdot m_f < c$ if and only if k is a solution to ϕ_t .*

Our goal is to synthesize an IHS that separates m_0 from m_f . Since IHS are t -inductive for all transitions t , we join all ϕ_t in a conjunction $\phi := \bigwedge_{t \in T} \phi_t$. The SMT-formula ϕ describes the desired linear approximation of the space of IHS. It is a main ingredient of our CEGAR loop outlined in Fig. 1. According to Lemma 3, solutions to ϕ are those vectors k that admit a constant c_t for each transition t such that (k, c_t) is t -inductive. The problem is that these c_t may be different for each transition. Hence, ϕ generates half space candidates and what is left to find is a single value c such that (k, c) is t -inductive for each t . We can compute all possible values for c with the algorithm CGA. A detailed explanation is given in Section 5. Once a common c is found, we have synthesized the desired IHS. Otherwise, the CEGAR loop starts the refinement.

If a solution k of ϕ does not have a suitable constant c to form an IHS, then neither does any multiple of k . This means we can exclude all multiples in future iterations of the CEGAR loop. Let $\text{mul}(k)$ be the formula satisfied by a $k' \in \mathbb{Z}^n$ if and only if there exists an $a \in \mathbb{N}$ such that $a \cdot k = k'$. Then, the refinement performs the update $\phi := \phi \wedge \neg \text{mul}(k)$. The following lemma states correctness.

Lemma 4. *Let $k' := a \cdot k$ with $a \in \mathbb{N}$. If (k', c) is an IHS, then so is $(k, \lceil \frac{c}{a} \rceil)$.*

The presented CEGAR approach generates inductive half spaces. In order to semi-decide LSV(R) , our approach needs to yield an IHS whenever we are given a yes-instance. This means we need to ensure that any candidate vector k is generated by the SMT-solver at some point so that we do not miss possible IHS. This is achieved by adding a constraint imposing a bound on the absolute values of the entries of k . If the formula becomes unsatisfiable, the bound is increased. It remains to show how our semi-decider for LSV(R) can be adapted to LSV(C) .

Coverability Recall that a solution k of ϕ satisfies Condition (0). It ensures the existence of a value c such that $k \cdot m_0 \geq c$ and $k \cdot m_f < c$, meaning $m \in \text{Sol}(k, c)$ and $m_f \notin \text{Sol}(k, c)$. While this is sufficient for disproving reachability, it is not for coverability. When we solve LSV(C) , we need to additionally guarantee that $\uparrow m_f \cap \text{Sol}(k, c)$ is empty. It turns out that this requirement can be captured by a simple modification of ϕ . We only need to ensure that k is negative.

Theorem 4. *Let (k, c) be a half space (not necessarily inductive) such that $m_f \notin \text{Sol}(k, c)$. Then we have $\uparrow m_f \cap \text{Sol}(k, c) = \emptyset$ if and only if $k \leq 0$.*

The intuition is as follows. If $k \leq 0$ does not hold, then we can start with $m := m_f$ and put tokens into a place i with $k_i > 0$ until $k \cdot m \geq c$. This means $k \leq 0$ is sufficient and necessary. Each solution k of ϕ satisfies $m_f \notin \text{Sol}(k, c)$ for some c . In order to disprove coverability, we apply Theorem 4 and add constraint $k \leq 0$ to ϕ . This ensures that any synthesized IHS separates m_0 from $\uparrow m_f$.

5 Checking Inductivity

We present the algorithms ICA and CGA. The former decides t -inductivity for a given half space (k, c) and transition t . The latter is an instrumentation of ICA capable of synthesizing all constants c such that (k, c) is t -inductive, if only the vector k is given. CGA constitutes the remaining bit of our CEGAR loop. Finally, we show that deciding t -inductivity is an coNP -complete problem. The proof once again employs a connection to discrete mathematics.

5.1 Algorithms

We start with the *inductivity checker* (ICA). Fix a half space (k, c) and a transition t . We need to decide whether (k, c) is t -inductive. If (k, c) is trivial with respect to t , then inductivity follows from Theorem 2. Hence, we assume that (k, c) is non-trivial. The idea of ICA is to algorithmically check the constraint formulated in Theorem 1 via dynamic programming. Roughly, the algorithm searches for a value $k \cdot m$, where $m \in \text{ACT}(t)$, that lies in the *target interval* $[c, c - k \cdot t^\Delta - 1]$. If such a value can be found, (k, c) is not t -inductive. Otherwise, it is t -inductive.

To state ICA, we adapt Theorem 1. Let $K := \{k(i) \mid i \in [1, n]\}$ contain all entries of the the given vector k . We consider sequences $k_1 \dots k_\ell \in K^*$. Note that k_i does not denote the i -th entry of k but the i -th element in the sequence.

Then, (k, c) is t -inductive if and only if there does not exist a sequence $k_1 \dots k_\ell$ with

$$c \leq k \cdot t^- + \sum_{i=1}^{\ell} k_i < c - k \cdot t^\Delta. \quad (6)$$

Algorithm 1: *Inductivity Checker* (ICA)

```

1 queue.add( $k \cdot t^-$ );
2 reached[ $k \cdot t^-$ ] := True;
3 repeat
4    $current := \text{queue.remove}()$ ;
5   if  $c \leq current < c - k \cdot t^\Delta$  then
6     return Not inductive;
7   for  $k \in K$  do
8     if  $(current + k < c - k \cdot t^\Delta \wedge k \geq 0)$ 
9        $\vee (current + k \geq c \wedge k \leq 0)$  then
10      if  $\neg \text{reached}[current + k]$  then
11        queue.add( $current + k$ );
12        reached[ $current + k$ ] := True;
13 until queue.isEmpty;
14 return Inductive

```

ICA is stated as Algorithm 1. It searches for a sequence in K^* satisfying (6). Recall that we assumed (k, c) to be non-trivial. Then, according to Theorem 3, k does not contain both, positive and negative entries. ICA starts at $k \cdot t^-$ and iteratively adds values of K until it either reaches the target interval $[c, c - k \cdot t^\Delta - 1]$ or finds that none such value is reachable. To this end, ICA employs dynamic programming. This avoids recomputing the same value and speeds up the running time. An example of a run of ICA is illustrated in Fig. 4. If the currently reached value lies below the target interval, at least one value of K has yet to be added. Once we overshoot the target interval, we can exclude the current value and go to the next one in the queue. When we hit the interval, we can report non-inductivity.

In the appendix we show that ICA is correct. Moreover, we prove that it runs in pseudopolynomial time. That is, polynomial in the values k, c , and t , or exponential in their bit size. Note that this does not contradict the coNP-hardness of checking t -inductivity which we prove below.

Constant Generation Given a vector k and a transition t , ICA can be instrumented to compute all values c such that (k, c) is t -inductive. We refer to the instrumentation as *constant generation algorithm* (CGA). CGA computes all necessary sums $k \cdot t^- + \sum_{i=1}^{\ell} k_i$ with $k_1 \dots k_\ell \in K^*$ and returns all c such that $[c, c - k \cdot t^\Delta - 1]$ does not contain any of the computed sums. Intuitively, we fit the interval between these sums. Note that each of the returned values c satisfies the

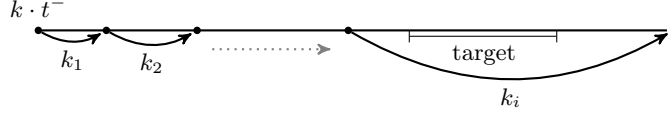


Fig. 4: Example run of Algorithm 1 (ICA) in the case $k \geq 0$. It starts at value $k \cdot t^-$. The algorithm adds values of K until it either overshoots the target interval or hits it.

characterization of t -inductivity as stated in (6). We show that CGA is correct and terminates.

For termination, we need the so-called *Frobenius number* [5]. Let $a \in \mathbb{N}^n$ be a vector such that $\gcd(a) = \gcd(a(1), \dots, a(n)) = 1$. Here, \gcd denotes the *greatest common divisor*. The *Frobenius number* of a is the largest integer that cannot be represented as a positive linear combination of $a(1), \dots, a(n)$. The number exists and is bounded by $a_{\max} \cdot a_{\min}$, where a_{\max} is the largest and a_{\min} the smallest entry of a [5]. Note that this means that each value $x \geq a_{\max} \cdot a_{\min}$ can be represented as a positive linear combination $x = a \cdot m$ with $m \in \mathbb{N}^n$.

This has implications for CGA. Assume we are given a vector $k \geq 0$ with $\gcd(k) = 1$. The possible values of c such that (k, c) is t -inductive cannot exceed $k \cdot t^- + k_{\max} \cdot k_{\min}$. Otherwise, the interval $[c, c - k \cdot t^\Delta - 1]$ will contain a linear combination of the form $k \cdot t^- + \sum_{i=1}^{\ell} k_i$ which breaks the inductivity requirement (6). The argument can be generalized for any $\gcd(k) \geq 1$:

Theorem 5. *Let (k, c) be a non-trivial t -inductive half space and let k_{\max}, k_{\min} denote the entries of k with maximal and minimal absolute value.*

1. *If $k \geq 0$, we have $c < k_{\max} \cdot k_{\min} + k \cdot t^-$.*
2. *If $k \leq 0$, we have $c \geq -k_{\max} \cdot k_{\min} + k \cdot t^-$.*

The theorem enforces termination and correctness of CGA. In fact, we only need to compute sums $k \cdot t^- + \sum_{i=1}^{\ell} k_i$ with $k_1 \dots k_i \in K^*$ up to the limit given in the theorem and still find all values c such that (k, c) is t -inductive. Since the limit is polynomial in the values of k and t , CGA runs in pseudopolynomial time.

We employ CGA within our CEGAR loop. Assume we have a solution k to our SMT-formula ϕ . It is left to decide whether there exists a $c \in \mathbb{N}$ such that (k, c) is an IHS. We apply CGA to k and each transition t . This yields a set C_t containing all c_t such that (k, c_t) is t -inductive and separates m_0 from m_f . Hence, the intersection $\bigcap_{t \in T} C_t$ contains all c such that (k, c) is an IHS that separates m_0 from m_f . Algorithmically, we only need to test the intersection for non-emptiness.

5.2 Complexity

We prove that deciding t -inductivity for a half space (k, c) and transition t is coNP-complete. Membership follows from a non-deterministic variant of ICA.

Further analyses show that the problem also lies in FPT and in coCSL, where CSL is the class of languages accepted by context-sensitive grammars. For unary input, it is in coNL and — if the dimension of k is fixed — in L. We provide details in the appendix. The interesting part is coNP-hardness for which we establish a reduction from the *unbounded subset sum problem* [21].

Theorem 6. *Checking t -inductivity of a half space (k, c) is coNP-complete.*

Before we elaborate on the reduction, we introduce the *unbounded subset sum problem* (USSP). An instance consists of a vector $w \in \mathbb{N}^n$ and an integer $d \in \mathbb{N}$. The task is to decide whether there exists a vector $x \in \mathbb{N}^n$ such that $w \cdot x = d$. The problem is NP-complete [21]. To prove Theorem 6, we reduce from USSP to the complement of checking t -inductivity. This yields the desired coNP-hardness.

Proof. Let (w, d) be an instance of USSP. We construct a half space (k, c) and a Petri net with a transition t such that (k, c) is not t -inductive if and only if there is an $x \in \mathbb{N}^n$ such that $w \cdot x = d$. We rely on the inductivity criterion from Theorem 1. The main difference between this criterion and USSP is that the latter requires reaching a precise value d , while the former requires reaching an interval. The idea is to define an appropriate half space (k, c) and a transition t such that in the corresponding interval only one value might be reachable.

We set $k = w$. Note that we can assume that d is a multiple of $\gcd(k)$. Otherwise, (w, d) is a no-instance of USSP since each linear combination $w \cdot x$ is a multiple of $\gcd(k)$. By using the Euclidean algorithm, we can compute an $a \in \mathbb{Z}^n$ such that $k \cdot a = \gcd(k)$ in polynomial time [2]. We construct a Petri net with n places and one transition t with $t^-(i) := a(i)$ if $a(i) > 0$, $t^+ := -a$ if $a(i) < 0$, and 0 otherwise. It holds $t^\Delta = -a$. Set $c = d + k \cdot t^-$. It is left to show that (k, c) is not t -inductive if and only if (w, d) is a yes-instance of USSP.

Assume that (k, c) is not t -inductive. Then there exists a vector $x \in \mathbb{N}^n$ such that $c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta$. By plugging in the above definitions, we obtain that $d \leq k \cdot x < d + \gcd(k)$. Since d , $d + \gcd(k)$, and $k \cdot x$ are all multiples of $\gcd(k)$, we obtain that $d = k \cdot x = w \cdot x$. Hence, (w, d) is a yes-instance of USSP.

For the other direction, let $w \cdot x = d$. We obtain that $d \leq k \cdot x < d + \gcd(k)$. As above, we can employ the definitions and derive that $c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta$. This shows non-inductivity of (k, c) and proves correctness of the reduction. \square

6 Experiments

We implemented the CEGAR loop in our Java prototype tool INEQUALIZER [16]. It employs Z3 [41] as a back-end SMT-solver. The tool makes use of *incremental solving* as well as *minimization*, a feature of Z3 that guides the CEGAR loop towards more likely candidates of IHS. Incremental solving reuses information learned from previous queries to Z3 and minimization prioritizes solutions with minimal values. Before INEQUALIZER starts the CEGAR loop, it uses an SMT-query to check whether there is a separating IHS (k, c) that is trivial for all transitions. We use minimization to get half spaces that are non-trivial with

Benchmark	$ P $	$ T $	INEQUALIZER	MIST				
				backward	ic4pn	tsi	eec	eec-cegar
BASICME	5	4	0.6	0.1	0.1	0.1	0.1	0.1
KANBAN	16	14	0.7	0.1	0.2	0.8	0.1	0.2
LAMPORT	11	9	T/O	0.1	0.1	0.1	0.1	0.1
MANUFACTURING	13	6	0.6	1.9	0.1	0.1	0.1	0.1
PETERSSON	14	12	T/O	0.2	0.1	0.1	0.1	0.1
READ-WRITE	13	9	0.5	0.1	1	0.1	0.9	0.5
MESH2X2	32	32	1.2	0.3	0.1	48.6	0.8	0.2
MESH3X2	52	54	2.1	2.2	0.2	T/O	T/O	2.2
MULTIPOOL	18	21	0.8	0.3	2	2.2	1	2.3

Table 1: INEQUALIZER vs. MIST.

respect to fewer transitions. The reason is that non-trivial half spaces are harder to find and typically only a few values for c ensure inductivity in this case. Before we show the applicability of INEQUALIZER on larger benchmarks, let us consider the Petri net in Fig. 2. When executing INEQUALIZER, we find that there are no trivial separating IHS. Using incremental solving, INEQUALIZER performs three iterations of the CEGAR loop and returns the non-trivial separating IHS with $k = (53, 52)$ and $c = 209$. When enabling minimization, we only require two iterations and obtain $k = (8, 5)$, $c = 22$. The difference in iterations is due to that we expect minimization to choose vectors k that are trivial for many transitions. This increases the chance of finding a suitable c . On the other hand, incremental solving improves the running time in executions with more iterations.

We evaluated INEQUALIZER for LSV(C) on a benchmark suite and compared it to various methods for coverability implemented in MIST [43,23,19,18,12]. Results are given in Table 1. The experiments were performed on a 1,7 GHz Intel Core i7 with 8GB memory. The running times are given in seconds. For entries marked as T/O, the timeout was reached. The running times of INEQUALIZER are similar to MIST although the former has a small overhead from generating the SMT-query. In each of the listed Petri nets, the unsafe marking is not coverable. Except for the mutual exclusion nets PETERSSON and LAMPORT, INEQUALIZER reliably finds separating IHS. Surprisingly, each found IHS is trivial. We suspect that the cases where INEQUALIZER timed out are actually negative instances of LSV(C).

The experiments show that many practical instances admit trivial IHS, which we synthesize using only one SMT-query. To test the generation of non-trivial IHS, we ran INEQUALIZER on a list of nets that do not admit trivial ones. The results are given in the appendix. They show that INEQUALIZER finds non-trivial IHS within few iterations of the CEGAR loop.

7 Conclusion and Outlook

We considered an invariant-based approach to disprove reachability and coverability in Petri nets. The idea was to synthesize an inductive half space that

over-approximates the reachable markings of the net and separates them from unsafe markings. For the synthesis, we established a structure theory of IHS and derived an SMT-formula which linearly approximates the space of IHS. We provided two algorithms, ICA and CGA. The former decides whether a half space is inductive, the latter generates suitable constants that guarantee inductivity. The SMT-formula and the algorithm CGA were then combined in a CEGAR loop which attempts to synthesize IHS. We implemented the loop into our tool INEQUALIZER. It combines SMT-queries with efficient heuristics and was capable of solving practical instances in our experiments.

We expect that further structural studies of IHS will improve the efficiency of the CEGAR loop. This may lead to a tighter approximation of the space of IHS or to an improved refinement step eliminating more than multiples. It is also an intriguing question whether the problems LSV(R) and LSV(C) are decidable. To tackle this, we are currently examining equivalence classes and normal forms of half spaces and their connection to well-quasi orderings.

Acknowledgements We thank Roland Meyer for his ideas and contributions that greatly influenced the work on inductive half spaces at hand. Moreover, we thank Chrisitan Eder for sharing with us his experience in commutative algebra, and Marvin Triebel for his ideas and the questions that he raised during his visit.

References

1. P. A. Abdulla, S. P. Iyer, and A. Nylén. Sat-solving the coverability problem for Petri nets. *Formal Methods in System Design*, 24(1):25–43, 2004.
2. E. Bach and J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms*. MIT Press, 1996.
3. D. Beyer, T. A. Henzinger, R. Majumdar, and A. Rybalchenko. Invariant synthesis for combined theories. In *VMCAI*, volume 4349 of *LNCS*, pages 378–394. Springer, 2007.
4. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *PLDI*, pages 196–207. ACM, 2003.
5. A. Brauer. On a problem of partitions. *American Journal of Mathematics*, 64(1):299–312, 1942.
6. E. Cardoza, R. Lipton, and A. R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups (preliminary report). In *STOC*, pages 50–54. ACM, 1976.
7. E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem. *Handbook of Model Checking*. Springer, 2018.
8. E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV*, volume 1855 of *LNCS*, pages 154–169. Springer, 2000.
9. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL*, pages 84–96. ACM, 1978.
10. W. Czerwinski, S. Lasota, R. Lazic, J. Leroux, and F. Mazowiecki. The reachability problem for Petri nets is not elementary. In *STOC*, pages 24–33. ACM, 2019.

11. W. Czerwiński and L. Orlikowski. Reachability in vector addition systems is ackermann-complete, 2021.
12. G. Delzanno, J. Raskin, and L. Van Begin. Towards the automated verification of multithreaded java programs. In *TACAS*, pages 173–187. Springer, 2002.
13. J. Esparza, R. Ledesma-Garza, R. Majumdar, P. Meyer, and F. Nksic. An SMT-based approach to coverability analysis. In *CAV*, pages 603–619. Springer, 2014.
14. J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In *CONCUR*, pages 2–20. Springer, 1999.
15. J. Esparza, S. Römer, and W. Vogler. An improvement of Mcmillan’s unfolding algorithm. In *TACAS*, volume 1055 of *LNCS*, pages 87–106. Springer, 1996.
16. F. Furbach. Inequalizer - a prototype tool for linear safety verification of Petri nets. <https://github.com/florianfurbach/Inequalizer>.
17. R. W. Floyd. Assigning meanings to programs. *Proceedings of a symposium on Applied Mathematics*, 19:19–32, 1967.
18. P. Ganty, C. Meuter, L. V. Begin, G. Kalyon, J. Raskin, and G. Delzanno. Symbolic data structure for sets of k-uples of integers. 2007.
19. P. Ganty, J. Raskin, and L. Begin. From many places to few: Automatic abstraction refinement for Petri nets. volume 88, pages 124–143, 06 2007.
20. P. Ganty, J.-F. Raskin, and L. Van Begin. From many places to few: automatic abstraction refinement for Petri nets. *Fundam. Inform.*, 88(3):275–305, 2008.
21. M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.
22. G. Geeraerts, J. Raskin, and L. Van Begin. On the efficient computation of the minimal coverability set for Petri nets. In *ATVA*, pages 98–113. Springer, 2007.
23. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Expand, enlarge and check: New algorithms for the coverability problem of WSTS. *Journal of Computer and System Sciences*, 72:180 – 203, 2006.
24. G.-M. Greuel and G. Pfister. *A Singular Introduction to Commutative Algebra*. Springer, 2002.
25. S. Gulwani, S. Srivastava, and R. Venkatesan. Program analysis as constraint solving. In *PLDI*, pages 281–292. ACM, 2008.
26. A. Gupta, R. Majumdar, and A. Rybalchenko. From tests to proofs. In *TACAS*, pages 262–276. Springer, 2009.
27. J. Hartmanis. Context-free languages and turing machine computations. In *Symposia in Applied Mathematics*, volume 19, pages 42–51, 1967.
28. C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.
29. A. Kaiser, D. Kroening, and T. Wahl. Efficient coverability analysis by proof minimization. In *CONCUR*, pages 500–515. Springer, 2012.
30. R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147 – 195, 1969.
31. J. Kloos, R. Majumdar, F. Nksic, and R. Piskac. Incremental, inductive coverability. In *CAV*, pages 158–173. Springer, 2013.
32. S. Rao Kosaraju. Decidability of reachability in vector addition systems. In *STOC*, pages 267–281. ACM, 1982.
33. J. Lambert. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.*, 99(1):79–104, 1992.
34. R. Langerak and E. Brinksma. A complete finite prefix for process algebra. In *CAV*, volume 1633 of *LNCS*, pages 184–195. Springer, 1999.
35. J. Leroux. The reachability problem for petri nets is not primitive recursive, 2021.

36. J. Leroux and S. Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *LICS*, pages 1–13. IEEE, 2019.
37. R. J. Lipton. *The reachability problem requires exponential space*. Research report (Yale University. Department of Computer Science). Department of Computer Science, Yale University, 1976.
38. E. Mayr. An algorithm for the general Petri net reachability problem. In *STOC*, pages 238–246. ACM, 1981.
39. E. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.
40. K. L. McMillan. A technique of state space search based on unfolding. *Form. Methods Syst. Des.*, 6(1):45–65, 1995.
41. L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
42. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
43. P. Ganty. *mist - a safety checker for Petri nets and extensions*. <https://github.com/pierreganty/mist>.
44. J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, 1981.
45. A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, pages 239–251. Springer, 2004.
46. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223 – 231, 1978.
47. P. Reynier and F. Servais. Minimal coverability set for Petri nets: Karp and Miller algorithm with pruning. In *PETRI NETS*, pages 69–88. Springer, 2011.
48. S. Sankaranarayanan, H. Sipma, and Z. Manna. *Petri Net Analysis Using Invariant Generation*, pages 682–701. Springer, 2003.
49. S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI*, volume 3385 of *LNCS*, pages 25–41. Springer, 2005.
50. M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.
51. M. Triebel and J. Sürmeli. Characterizing stable inequalities of Petri nets. In *PETRI NETS*, volume 9115 of *LNCS*, pages 266–286. Springer, 2015.
52. A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 01 1937.
53. A. Valmaris and H. Hansen. Old and new algorithms for minimal coverability sets. In *PETRI NETS*, pages 208–227. Springer, 2012.

Appendices

A Proof of Theorem 1

Theorem 1. *A half space (k, c) is t -inductive iff there is no vector $x \in \mathbb{N}^n$ with*

$$c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta.$$

Proof. We use the following property:

$$\exists m \in \mathbb{N}^n : m \geq t^- \Leftrightarrow \exists x \in \mathbb{N}^S : m = x + t^-. \quad (1)$$

According to the definition of t -inductivity, a half space is not t -inductive iff there is a marking m that satisfies the following condition:

$$\begin{aligned} & \exists m \in \mathbb{N}^n : k \cdot m \geq c \wedge m \geq t^- \wedge k \cdot m + k \cdot t^\Delta < c \\ \stackrel{(1)}{\Leftrightarrow} & \exists x \in \mathbb{N}^n : k \cdot x + k \cdot t^- \geq c \wedge k \cdot x + k \cdot t^- + k \cdot t^\Delta < c \\ \Leftrightarrow & \exists x \in \mathbb{N}^n : k \cdot x + k \cdot t^- \geq c \wedge k \cdot x + k \cdot t^- < c - k \cdot t^\Delta \\ \Leftrightarrow & \exists x \in \mathbb{N}^n : c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta \end{aligned}$$

B Proofs for Section 3

Lemma 5. *Let (k, c) be oriented towards t . If $m \in \text{Act}(t) \cap \text{Sol}(k, c)$, then $m + t^\Delta \in \text{Sol}(k, c)$.*

Proof. The lemma holds by $k \cdot (m + t^\Delta) = k \cdot m + k \cdot t^\Delta \geq k \cdot m \geq c$. The first equality is by distributivity of the scalar product, the following inequality is by the definition of orientation towards t , and the last inequality is by $m \in \text{Sol}(k, c)$. Finally, $(m + t^\Delta) \in \text{Sol}(k, c)$ follows from $k \cdot (m + t^\Delta) \geq c$ according to the definition of $\text{Sol}(k, c)$.

Lemma 6. *Let (k, c) be monotone for t . If $m \in \text{Act}(t) \cap \text{Sol}(k, c)$, then $m + t^\Delta \in \text{Sol}(k, c)$.*

Proof. Note that membership $m \in \text{Act}(t)$ implies $m = t^- + v$ with $v \in \mathbb{N}^n$. We have $k \cdot (m + t^\Delta) = k \cdot (t^- + t^\Delta + v) = k \cdot (t^- + t^\Delta) + k \cdot v \geq c$. The first equality rearranges the terms of $m + t^\Delta$, the second is distributivity, the third is by the fact that $k \cdot (t^- + t^\Delta) \geq c$ and $k \cdot v \geq 0$. The former holds by the definition of monotonicity for half spaces, the latter is by the fact that $k, c, v \geq 0$. It holds $m + t^\Delta \in \text{Sol}(k, c)$ since $k \cdot (m + t^\Delta) \geq c$.

Lemma 7. *If (k, c) is antitone for t , then $\text{Act}(t) \cap \text{Sol}(k, c) = \emptyset$.*

Proof. Note that membership $m \in \text{Act}(t)$ means $m = t^- + v$ for some $v \in \mathbb{N}^n$. Now $k \cdot m = k \cdot (t^- + v) = k \cdot t^- + k \cdot v < c$ follows. The inequality is by $k \cdot t^- < c$ and $k \cdot v \leq 0$. The former holds by the definition of antitone half spaces, the latter by $k \leq 0$ and $v \geq 0$.

Lemma 2. *Let (k, c) be a t -inductive half space that is non-trivial for t . For any entry $k(i)$ of k , with $|k(i)|$ denoting its absolute value, we have:*

$$k(i) = 0 \vee |k(i)| \geq -k \cdot t^\Delta \quad (5)$$

Proof. Assume towards contradiction that there is a $k(i)$ with $0 < |k(i)| \leq -k \cdot t^\Delta$. Since it is non-trivial, we can apply Lemma 1 and either (a) or (b) holds. For both cases, we define a value z such that $c \leq k \cdot t^- + z \cdot k(i) < c - k \cdot t^\Delta$ is satisfied.

Case (a) $k \cdot t^- < c - k \cdot t^\Delta$ and $k \geq 0$: There is a smallest $z \in \mathbb{N}$ such that $c \leq k \cdot t^- + z \cdot k(i)$. If $k \cdot t^- \geq c$, then $z = 0$ holds trivially. Note that $k \cdot t^- + z \cdot k(i) = k \cdot t^- < c - k \cdot t^\Delta$ holds by (a). If $k \cdot t^- < c$, then z exists since $k(i) > 0$. Assume towards contradiction that $c - k \cdot t^\Delta \leq k \cdot t^- + z \cdot k(i)$. It holds

$$k \cdot t^- + (z - 1) \cdot k(i) \geq c - k \cdot t^\Delta - k(i) \geq c.$$

The first inequality is by the assumption $c - k \cdot t^\Delta \leq k \cdot t^- + z \cdot k(i)$ and the second is by $|k(i)| \leq -k \cdot t^\Delta$. This means $z - 1$ also satisfies the condition which is a contradiction to z being smallest. The condition $c \leq k \cdot t^- + z \cdot k(i) < c - k \cdot t^\Delta$ is satisfied either way.

Case (b) $k \cdot t^- \geq c$ and $k \leq 0$: There is a smallest $z \in \mathbb{N}$ such that $k \cdot t^- + z \cdot k(i) < c - k \cdot t^\Delta$. Since z is smallest and $|k(i)| \leq -k \cdot t^\Delta$ holds it follows $c \leq k \cdot t^- + z \cdot k(i) < c - k \cdot t^\Delta$ as well. The detailed proof is omitted since it is analogue to (a).

We define the vector $x \in \mathbb{N}^n$ as $x = z \cdot e_i$. According to Theorem 1, this implies that (k, c) is not t -inductive which is a contradiction. \square

C Proofs for Section 4

In order to prove Lemma 3, we first require the following lemma which examines the non-trivial case:

Lemma 8. *Let k be an non-mixed vector with $k \cdot t^\Delta < 0$ and $|k(i)| > -k \cdot t^\Delta$ for all $k(i) \neq 0$. Then there is a $c \in \mathbb{Z}$ so that $k \cdot m \geq c$ is t -inductive.*

Proof. We examine both cases for the non-mixed vector k : $k \geq 0$ and $k \leq 0$.

$k \geq 0$: First, we assume $k \geq 0$ and set $c := k \cdot t^- + 1$. From $k \cdot t^\Delta < 0$ and $|k(i)| > -k \cdot t^\Delta$ follows $k(i) > 1$ for all $k(i) \neq 0$. We check if the condition in Theorem 1 is satisfied for any $x \in \mathbb{N}^n$:

$$c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta.$$

If x is the vector 0^n then it follows $c = k \cdot t^- + 1 \not\leq k \cdot x + k \cdot t^- = k \cdot t^-$.

Let x contain an entry $x(i) > 0$. If $k(i) = 0$ for all such entries then the case is analogue to $x = 0^n$. If there is an $i \leq n$ with $x(i) > 0$ and $k(i) > 0$ (and thus $k(i) \geq -k \cdot t^\Delta + 1$), then it holds

$$k \cdot x + k \cdot t^- \geq k(i) + k \cdot t^- \geq -k \cdot t^\Delta + 1 + k \cdot t^- = c - k \cdot t^\Delta.$$

$k \leq 0$: We define $c := k \cdot t^- + k \cdot t^\Delta$. If x is the vector 0^n , then it follows $k \cdot x + k \cdot t^- = k \cdot t^- \not\leq k \cdot t^\Delta = c - k \cdot t^\Delta$. Let x contain an entry $x(i) > 0$. If $k(i) = 0$ for all such entries then the case is analogue to $x = 0^n$. If there is an $i \leq n$ with $x(i) > 0$ and $k(i) < 0$ (and thus $k(i) < k \cdot t^\Delta$), then it holds

$$k \cdot x + k \cdot t^- \leq k(i) + k \cdot t^- < k \cdot t^\Delta + k \cdot t^- = c.$$

This means the condition is not satisfied in either case and the half space is t -inductive according to Theorem 1.

We recall the Theorem 1 from Section 4:

Lemma 3. *There exists a constant $c \in \mathbb{Z}$ such that (k, c) is a t -inductive half space with $k \cdot m_0 \geq c$ and $k \cdot m_f < c$ if and only if k is a solution to ϕ_t .*

Proof. Now, we examine the formula ϕ . Let (k, c) be separating, meaning $m_0 \in \text{Sol}(k, c)$ and $m_0 \notin \text{Sol}(k, c)$, and t -inductive. Since it is separating, condition (0) holds and c is in the interval $(k \cdot m_f, k \cdot m_0]$. If it is trivial, it follows that one of the conditions (1)-(3) holds. If it is non-trivial, conditions (4) and (5) hold. It follows that k is a satisfying assignment of ϕ_t .

Let k be a solution of ϕ_t . We show that there is a value c such that (k, c) is t -inductive:

- If k satisfies (1), it is t -inductive for any c and since (0) holds, we can choose a c such that it is separating.
- If k satisfies (2) then we set $c = k \cdot m_0$ and thus (k, c) is separating and antitone.
- Condition (3) is analogue, here we set $c = k \cdot m_f + 1$ and thus (k, c) is separating and monotone.
- If k satisfies Conditions (4) and (5), then the property holds according to the following Lemma.

Lemma 4. *Let $k' := a \cdot k$ with $a \in \mathbb{N}$. If (k', c) is an IHS, then so is $(k, \lceil \frac{c}{a} \rceil)$.*

Proof. We use contraposition. Given a marking m that violates t -inductivity of $(k, \lceil \frac{c}{a} \rceil)$. We show that it violates t -inductivity of (k', c) as well.

If it holds $k \cdot m \geq \lceil \frac{c}{a} \rceil$, then it follows $k \cdot m \geq \frac{c}{a}$ from $\lceil \frac{c}{a} \rceil \geq \frac{c}{a}$ and thus $k' \cdot m \geq c$. If it holds $k \cdot (m + t^\Delta) < \lceil \frac{c}{a} \rceil$ then it follows $k \cdot (m + t^\Delta) + 1 \leq \lceil \frac{c}{a} \rceil$. Finally, we conclude $k' \cdot (m + t^\Delta) < c$ using $\lceil \frac{c}{a} \rceil \leq \frac{c}{a} + 1$. \square

Theorem 7. *Let (k, c) be such that $m_f \notin \text{Sol}(k, c)$. It holds $m_f \uparrow \cap \text{Sol}(k, c) = \emptyset$ iff $k \leq 0$.*

Proof. " \Rightarrow :" Let $m_f \uparrow \cap \text{Sol}(k, c) = \emptyset$. We assume towards contradiction that $k(i) > 0$ holds for some $i \leq n$. Let m be such that $m(i) = m_f(i) + (c - k \cdot m_f)$ and $m(j) = m_f(j)$ for all $j \leq n$ with $j \neq i$. Then $k \cdot m = k \cdot m_f + k(i) \cdot (c - k \cdot m_f)$. Since $k(i) > 0$ and $(c - k \cdot m_f) > 0$, it follows $k \cdot m \geq k \cdot m_f + 1 \cdot (c - k \cdot m_f) = k \cdot m_f + c - k \cdot m_f = c \geq c$ and thus $m \in m_f \uparrow \cap \text{Sol}(k, c) \neq \emptyset$. This is a contradiction to $m_f \uparrow \cap \text{Sol}(k, c) = \emptyset$.

" \Leftarrow :" Let $k \leq 0$ and $m \in m_f \uparrow$ and thus $m \geq m_f$. It follows $k \cdot m \leq k \cdot m_f < c$ and thus $m \notin \text{Sol}(k, c)$. This means $m_f \uparrow \cap \text{Sol}(k, c) = \emptyset$.

D Proofs for Section 5

We recall Theorem 8 as well as Algorithm 1.

Theorem 8. *A half space (k, c) is t-inductive iff*

$$\nexists k_1 \dots k_l \in K^* : c \leq k \cdot t^- + \sum_{i=1}^l k_i < c - k \cdot t^\Delta$$

Proof. We use the following property: For any sum $\sum_{i=1}^l k_i$ over K with $x(j)$ the number of occurrences of value k_j in the sum, it holds

$$\sum_{i=1}^l k_i = \sum_{j=1}^n k(j) \cdot x(j) = k \cdot x. \quad (2)$$

Obviously, we can construct a sum with value $k \cdot x$ for any vector x . It follows:

$$\exists x \in \mathbb{N}^n : k \cdot x = z \Leftrightarrow \exists k_1 \dots k_l \in K^* : \sum_{i=1}^l k_i = z \quad (3)$$

A half space is not t-inductive iff there is a marking x that violates Theorem 1:

$$\begin{aligned} & \exists x \in \mathbb{N}^n : c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta \\ & \stackrel{(3)}{\Leftrightarrow} \exists k_1 \dots k_l \in K^* : c \leq \sum_{i=1}^l k_i + k \cdot t^- < c - k \cdot t^\Delta \end{aligned}$$

Lemma 9. *The algorithm is correct.*

Proof. Assume the algorithm returns "Not inductive". It holds $c \leq \text{current} < c - k \cdot t^\Delta$ and current was derived by adding elements of K to the starting value $k \cdot t^-$. It follows that there is a sequence $k_1 \dots k_l \in K^*$ and $c \leq k \cdot t^- + \sum_{i=1}^l k_i < c - k \cdot t^\Delta$. This violates the condition of Theorem 8 and thus the half space is not t-inductive.

Lemma 10. *The algorithm is complete.*

Proof. We show that the algorithm identifies any half space that is not t-inductive. We know that k is not mixed. We now examine the case $k \geq 0$. Let $k_1 \dots k_l$ be a sequence that satisfies the condition of Theorem 8. W.l.o.g., we can assume that $k_1 \dots k_l$ is the shortest sequence that reaches the target area. We examine its prefixes $k_1 \dots k_i$ with $i < l$. It follows that $k \cdot t^- + \sum_{j=1}^i k_j < c$ holds for all $i < l$ and $k \cdot t^- + \sum_{j=1}^l k_j < c - k \cdot t^\Delta$.

We now apply a induction over the sequence to show that the algorithm processes $k \cdot t^- + \sum_{j=1}^l k_j$ or returns "Not inductive" before that:

Induction basis: The algorithm processes $k \cdot t^-$ (Line 1 and 2).

Induction hypothesis: The algorithm processes $current = k \cdot t^- + \sum_{j=1}^i k_j$.

Induction step: We know $current + k_{i+1} = k \cdot t^- + \sum_{j=1}^{i+1} k_j < c - k \cdot t^\Delta$ holds and thus the condition in Line 8 is satisfied. It is added to the queue (Line 11) and it is either processed later or the algorithm returns "Not inductive" before that. The argument is analogue for $k \leq 0$. Here, the condition in Line 9 is satisfied.

It follows that unless "Not inductive" is returned earlier, $k \cdot t^- + \sum_{j=1}^l k_j$ is processed. It meets the condition in Line 5 and the algorithm returns "Not inductive".

Theorem 9. *The run-time of Algorithm 1 is polynomial in the input values.*

Proof. We show that the algorithm's runtime is polynomial in the input values. The processing time of one value is linear in $|K|$. Either the lowest processed value is the starting value $k \cdot t^-$ and the highest is some value at most $c - k \cdot t^\Delta$ (garanteed by the condition in Line 8) or the highest processed value is the starting value $k \cdot t^-$ and the lowest is at least c (garanteed by the condition in Line 9). It follows that the algorithm only processes values in the polynomial sized segment

$$[\min(k \cdot t^-, c), \max(k \cdot t^-, c - k \cdot t^\Delta)].$$

Since every processing step reaches a new unprocessed value in the segment, the number of processing steps are limited by the segment size

$$l_s := |\max(k \cdot t^-, c - k \cdot t^\Delta) - \min(k \cdot t^-, c)|.$$

We continue by analyzing the space-complexity of the problem. We study the complexity class **L** which denotes problems that can be solved deterministically using an amount of memory space that is logarithmic in the size of the input. The class **NL** describes problems that can be solved non-deterministically in logarithmic space. The problems that can be solved non-deterministically using space that is linear in the input size are in **CSL**. For any class of problems **C**, we denote the class of their complements as **co-C**.

Theorem 10. *A half space is not t-inductive if and only if there is a vector x that violates Theorem 1 and $x(j) \leq l_s$ for all $j \leq n$*

Proof. Since there are l_s many possible values, and the algorithm processes a value only once, the algorithm constructs a sum of length at most l_s iff it is not t-inductive. It follows from (Equation 2) that if (Theorem 1) is violated, than it is violated by some vector x with $|x| \leq l_s$.

It follows from Theorem 10 that deciding inductivity is in co-NP. Choosing some vector x with values at most l_s non-deterministically and checking if it violates Theorem 1 takes polynomial time.

We assume the dimension n of k (which is the number of places in the Petri net) is a fixed parameter and introduce a new algorithm that solves t-inductivity

in logarithmic space. It simply iterates all possible vectors x that satisfy $x(j) \leq l_s$ for all $j \leq n$ and checks whether they satisfy the inequality. The successor function *succ* handles the vector $x \leq l_s$ like a number with n digits to the basis l_s and works like a standard successor. It starts at the first value and if it is less than l_s it adds one and terminates, if the current value is l_s , it sets it to 0 and handles the next one.

Algorithm 2: Inductivity-LogSpace

```

1  $x = 0^n$ ;
2 repeat
3   if  $c \leq k \cdot x + k \cdot t^- < c - k \cdot t^\Delta$  then
4     return Not inductive
5    $x = \text{succ}_{l_s}(x)$ ;
6 until  $x = l_s^n$ ;
7 return Inductive

```

The value of l_s is linear in every input variable and thus it can be stored in logarithmic space. Any vector x with $x(j) \leq l_s$ for all $j \leq n$ can also be stored in logspace.

Theorem 11. *Deciding inductivity of a half space is in \mathbf{L} for unary encoded input and fixed dimension of k .*

A nondeterministic version of the inductivity algorithm has to store only the current value and the number of executed steps and it executes at most l_s steps.

Theorem 12. *Deciding inductivity of a half space is in $\mathbf{co-NL}$ for unary encoded input.*

For binary encoded input, it follows from Theorem 12:

Theorem 13. *Deciding inductivity of a half space is in $\mathbf{co-CSL}$ for binary encoded input.*

For an instance of the inductivity problem given by a half space and a transition we introduce the parametrized instance with the greatest total value k_{max} of k as the parameter.

Theorem 14. *The parametrized inductivity problem is fixed parameter tractable.*

Proof. For any vector $x \in \mathbb{N}^n$ with $x_i \geq k_1$ it holds

$$k \cdot x = k \cdot (x_1 + k_i, \dots, x_{i-1}, x_i - k_1, x_{i+1}, \dots, x_n)^T$$

We iterate this argument and it follows that if there is a vector that satisfies the condition of Theorem 1 then it is also satisfied by a vector x with $x_2, \dots, x_n \leq k_1$. We assume $k \geq 0$ and $k \cdot t^- < c$.

$$\begin{aligned}
k \cdot t^- + k \cdot x &= k \cdot t^- + k_1 \cdot x_1 + \sum_{i=2}^n k_i \cdot x_i \geq c \\
\Rightarrow k_1 \cdot x_1 &\geq c - k \cdot t^- - \sum_{i=2}^n k_i \cdot x_i \geq c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i \\
\Rightarrow x_1 &\geq \lceil \frac{c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i}{k_1} \rceil
\end{aligned}$$

Instead of imposing a lower bound on x_1 we introduce x' with $x_1 = x'_1 - \lceil \frac{c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i}{k_1} \rceil$ and $x'_i = x_i$ for $i > 1$:

$$k \cdot x = k \cdot x' + k_1 \cdot \lceil \frac{c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i}{k_1} \rceil$$

It follows that a half space $k \cdot m \geq c$ is t-inductive iff the following half space is t-inductive: $k \cdot m \geq c - k_1 \cdot \lceil \frac{c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i}{k_1} \rceil$. Note that the new half space is only bounded by k_{max} and not c :

$$c - k_1 \cdot \lceil \frac{c - k \cdot t^- - k_1 \cdot \sum_{i=2}^n k_i}{k_1} \rceil \leq k \cdot t^- + k_1 \cdot \sum_{i=2}^n k_i$$

The construction for $k \leq 0$ and $k \cdot t^- > c$ is analogue.

D.1 Proofs for Generating c

The main contribution of this subsection is the proof of Theorem 5. This requires the following two technical lemmas.

Lemma 11.

$$|\gcd k| \leq |k \cdot t^\Delta|$$

Proof. It holds $k \cdot t^\Delta = t^\Delta(1) \cdot k(1) + \dots + t^\Delta(n) \cdot k(n) = z \cdot \gcd k$ for some $z \in \mathbb{Z}$. It follows $|k \cdot t^\Delta| \geq |\gcd k|$.

Lemma 12. Let (k, c) be a non-trivial t-inductive half space and let $y \in \mathbb{N}$ denote the Frobenius number of $\frac{k(1)}{\gcd(k)}, \dots, \frac{k(n)}{\gcd(k)}$.

- a) If $k \geq 0$, it holds $k \cdot t^- + k \cdot t^\Delta < c \leq \gcd(k) \cdot y + k \cdot t^-$.
- b) If $k \leq 0$, it holds $\gcd(k) \cdot y + k \cdot t^- \leq c < k \cdot t^-$.

Proof. We prove the lower and upper bounds for both cases.

- a) The lower bound follows immediately from Lemma 1. For the upper bound, we assume $c > \gcd(k) \cdot y + k \cdot t^-$ and thus $\frac{c - k \cdot t^-}{\gcd(k)} > y$. Since y is the Frobenius number, there is a vector $b \in \mathbb{N}^n$ such that $\lfloor \frac{c - k \cdot t^-}{\gcd(k)} \rfloor = \frac{k^T}{\gcd(k)} \cdot b$. This means

$$c - k \cdot t^- \leq k \cdot b < c - k \cdot t^- + \gcd(k).$$

According to Lemma 1, it holds $k \cdot t^\Delta < 0$. We apply Lemma 11 and get $-k \cdot t^\Delta \geq \gcd(k)$. This means that $c \leq k \cdot b + kt^- < c - k \cdot t^\Delta$ holds and thus the vector b satisfies Theorem 1. This is a contradiction to inductivity.

- b) The upper bound follows immediately from Lemma 1. For the lower bound, we assume $c < \gcd(k) \cdot y + k \cdot t^-$ and thus $\frac{c-kt^-}{\gcd(k)} > y$. The remainder is analogue to a).

We recall Theorem 5:

Theorem 5. *Let (k, c) be a non-trivial t -inductive half space and let k_{\max}, k_{\min} denote the entries of k with maximal and minimal absolute value.*

1. *If $k \geq 0$, we have $c < k_{\max} \cdot k_{\min} + k \cdot t^-$.*
2. *If $k \leq 0$, we have $c \geq -k_{\max} \cdot k_{\min} + k \cdot t^-$.*

Proof. Let y denote the Frobenius number of $\frac{k(1)}{\gcd(k)}, \dots, \frac{k(n)}{\gcd(k)}$. By Lemma 2 and Lemma 11, we know that $|k(i)| > -k \cdot t^\Delta \geq |\gcd(k)|$. From this we obtain that $\frac{k(i)}{\gcd(k)} = \frac{|k(i)|}{|\gcd(k)|} \geq 2$. Now we apply the definition of the Frobenius number and get: $y \leq (\frac{k_{\max}}{\gcd(k)} - 1)(\frac{k_{\min}}{\gcd(k)} - 1)$.

Now assume that $k \geq 0$. We give an estimation for $\gcd(k) \cdot y$:

$$\begin{aligned} \gcd(k) \cdot y &\leq \gcd(k) \cdot \left(\frac{k_{\max}}{\gcd(k)} - 1\right) \left(\frac{k_{\min}}{\gcd(k)} - 1\right) \\ &\leq \gcd(k)^2 \cdot \left(\frac{k_{\max}}{\gcd(k)} - 1\right) \left(\frac{k_{\min}}{\gcd(k)} - 1\right) \\ &\leq (k_{\max} - \gcd(k))(k_{\min} - \gcd(k)) \\ &\leq k_{\max} \cdot k_{\min}. \end{aligned}$$

We now combine this with the bound proven in Lemma 12 and derive the criterion of Theorem 5

$$c < k_{\max} \cdot k_{\min} + k \cdot t^-.$$

If $k \leq 0$, we derive a similar bound. Note that it holds $\gcd(k) < 0$. We now derive a lower bound of $\gcd(k) \cdot y$:

$$\begin{aligned} \gcd(k) \cdot y &\geq \gcd(k) \cdot \left(\frac{k_{\max}}{\gcd(k)} - 1\right) \cdot \left(\frac{k_{\min}}{\gcd(k)} - 1\right) \\ &\geq -\gcd(k)^2 \cdot \left(\frac{k_{\max}}{\gcd(k)} - 1\right) \cdot \left(\frac{k_{\min}}{\gcd(k)} - 1\right) \\ &\geq -(k_{\max} - \gcd(k)) \cdot (k_{\min} - \gcd(k)) \\ &\geq -k_{\max} \cdot k_{\min}. \end{aligned}$$

Like above, we apply this to the bound on c from Lemma 12 and get

$$c \geq -k_{\max} \cdot k_{\min} + k \cdot t^-$$

E Non-Trivial Petri Nets

Since the benchmark suite did not require non-trivial separating IHS, it did not accurately present our CEGAR method. We would like a better understanding of which Petri nets require non-trivial separating IHS. For this purpose, we construct a simple Petri net that has a non-trivial separating IHS but not a trivial one. We begin by collecting sufficient conditions of a Petri net that ensure non-triviality for any separating IHS.

Lemma 13. *Let $a \in \mathbb{R}_+^m$ be such that $m_f = m_0 + \sum_{i=1}^m a(i) \cdot t_i^\Delta$. For any separating half space, there is a transition that is not oriented towards it.*

Proof. Since the half space is separating, it holds $k \cdot m_0 \geq c > k \cdot m_f$ and thus $0 > k \cdot (m_f - m_0) = \sum_{i=1}^m a(i) \cdot k \cdot t_i^\Delta$. One element in the sum has to be negative: $\exists_{i < m} : a(i) \cdot k \cdot t_i^\Delta < 0$. Since $a(i)$ can not be negative, it follows $\exists_{i < m} : k \cdot t_i^\Delta < 0$. So (k, c) is not oriented towards t_i .

It follows that, for any separating half space, one of the transitions t_i with an associated value a_i greater than zero is not oriented towards it. In order to ensure that the separating half space is not trivial, we require two additional properties: it can neither be antitone, nor monotone.

For any t_i with $a_i > 0$ we require $t_i^- \leq m_0$, i.e. the transition is activated in the initial marking. This means $\text{Act}(t) \cap \text{Sol}(k, c) \neq \emptyset$ and thus it is not antitone.

For any t_i with $a_i > 0$ we require $t_i^- + t_i^\Delta \leq m_f$, which means there is a marking from which t_i can be fired in order to reach m_f . Assume there is separating half space (k, c) that is monotone for t_i . Then it holds $k \geq 0$ and thus $k \cdot m_f \geq k \cdot (t_i^- + t_i^\Delta)$. Since $k \cdot m_f < c$, it follows $k \cdot (t_i^- + t_i^\Delta) < c$. This is a contradiction to monotonicity for t_i .

In summary, if the marking equation has a continuous solution such that the used transitions can all be fired from the initial marking and they can all be fired to reach m_f , then there are no trivial separating half spaces.

This sufficient condition for non-triviality is useful, because it is not much stronger than the following necessary condition for unreachability. If no solution of the marking equation exists where at least one used transition can be fired in the beginning and one in the end, then m_f is unreachable. This condition is very easy to check. This comparison suggests that for a Petri net where it is not immediately obvious that m_f is unreachable, a non-trivial half space is likely to be required.

Example We now construct a minimal non-trivial example for larger dimensions. We introduce a Petri net N_n of size $n \geq 3$ that has a non-trivial separating half space but no trivial separating half spaces (see Fig. 5). Furthermore, if one transition is removed, a trivial separating invariant exists.

We set $m_0 = 1^n, m_f = 2^n$, meaning we start with one token in each place and ask whether we can avoid getting having tokens in each place. Then, we choose some $j \leq n$ and we define n transition t_1, \dots, t_n such that each transition

t_i removes one token from each place and then puts n tokens in place p_i . The exception is t_j which puts $n + 1$ tokens into p_j .

Formally, this means $t_i^- = 1^n$ for all $i \leq n$ and $t_i^\Delta(i) = n - 1, t_i^\Delta(k) = -1$ for $k \leq n, k \neq i$. For t_j , it holds $t_j^\Delta(i) = -1$ for $i \leq n, i \neq j$ and $t_j^\Delta(j) = n$.

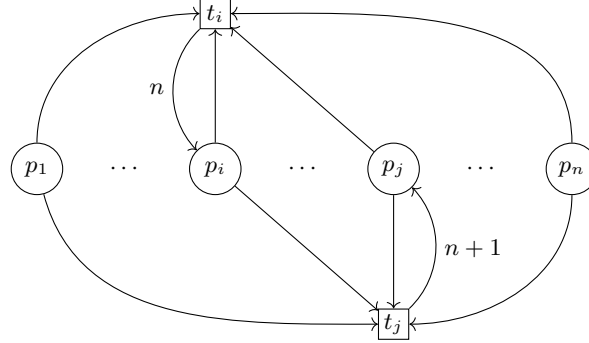


Fig. 5: A non-trivial Petri net

Lemma 14. *The Petri net N_n has a separating non-trivial IHS but no separating trivial IHS.*

Proof. Let $c = -n \cdot (n + 1)$, $k(j) = -n$, and $k(i) = -(n + 1)$ for $i \neq j, i \leq n$. The half space is separating, since it holds $k \cdot m_0 = -(n + 1) \cdot (n - 1) - n = -n \cdot (n + 1) + 1 > c$ and $k \cdot m_f = -(n + 1) \cdot (n - 1) \cdot 2 - 2n = -2(n + 1) \cdot n + 2 < c$.

We show that the half space is a IHS using Theorem 8. Since $k \cdot t_j^\Delta = (n + 1) \cdot (n - 1) - n^2 = -1$ it holds $k \cdot t_j^- = k \cdot m_0 \geq c - k \cdot t_j^\Delta$. If we add any k_i then we get $k \cdot t_j^- + k_i = -n \cdot (n + 1) + 1 - n < c$ and if we add k_j or additional values of k we get an even smaller value.

Let (k, c) be any separating half space. Since $k \cdot m_0 > k \cdot m_f$, it holds $-k_1 \dots -k_n > 0$. Let $k_l = \min(k_1, \dots, k_n)$ be the negative entry of k with the largest absolute value. If $l = j$, then $k \cdot t_l^\Delta = -k_1 \dots -k_n + n \cdot k_l + k_l \leq -n \cdot k_l + n \cdot k_l + k_l = k_l < 0$.

If $l \neq j$, then $k \cdot t_l = -k_1 \dots -k_n + n \cdot k_l = k_l$. If all entries of k are not equal, then it holds $-k_1 + \dots -k_n > n \cdot k_l$ and thus $k \cdot t_l < 0$. If all entries of k are equal then they are also negative and it holds

$$k \cdot t_j = -k_1 \dots -k_n + n \cdot k_j + k_j = -n \cdot k_j + n \cdot k_j + k_j = k_j < 0.$$

It follows that any separating half space is oriented towards at least one transition.

Obviously all transitions are enabled at m_0 and the half space is not antitone. According to $-k_1 \dots -k_n > 0$, it holds $k \not\geq 0$ and thus it is not monotone either.

$ P $	Iterations	Time
3	2	0.4
4	113	6.9
5	2	0.4
6	2	0.4
7	6	0.6
8	3	0.6
9	378	205.2
10	2	0.5

Table 2: INEQUALIZER on non-trivial Petri nets.

We evaluate the performance of INEQUALIZER for reachability on the non-trivial Petri nets of sizes three to ten in Table 2. We give the number of iterations of the CEGIS loop performed by the tool. We do not include the run-time results of MIST for these Petri nets in the table since they were all well below 0.1 second. We use incremental solving and find that our tool usually computes the IHS quickly using few iterations. There are only two diverging results where the SMT-solver returns a number of unusable vectors k .