Acyclic Petri and Workflow Nets with Resets

Dmitry Chistikov
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Wojciech Czerwiński
University of Warsaw, Poland

Piotr Hofman
University of Warsaw, Poland

Filip Mazowiecki
University of Warsaw, Poland

Henry Sinclair-Banks
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Abstract

In this paper we propose two new subclasses of Petri nets with resets, for which the reachability and coverability problems become tractable. Namely, we add an acyclicity condition that only applies to the consumptions and productions, not the resets. The first class is acyclic Petri nets with resets, and we show that coverability is PSPACE-complete for them. This contrasts the known Ackermann-hardness for coverability in (not necessarily acyclic) Petri nets with resets. We prove that the reachability problem remains undecidable for acyclic Petri nets with resets. The second class concerns workflow nets, a practically motivated and natural subclass of Petri nets. Here, we show that both coverability and reachability in acyclic workflow nets with resets are PSPACE-complete. Without the acyclicity condition, reachability and coverability in workflow nets with resets are known to be equally hard as for Petri nets with resets, that being Ackermann-hard and undecidable, respectively.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Petri nets, Workflow Nets, Resets, Acyclic, Reachability, Coverability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.16

Related Version *Full Version*: `https://arxiv.org/abs/2310.01992`

Funding *Dmitry Chistikov*: Supported in part by the Engineering and Physical Sciences Research Council [EP/X03027X/1].
Wojciech Czerwiński: Supported by the ERC grant INFSYS, agreement no. 950398.
Piotr Hofman: Supported by the ERC grant INFSYS, agreement no. 950398.
Filip Mazowiecki: Supported by the ERC grant INFSYS, agreement no. 950398.
Henry Sinclair-Banks: Supported by EPSRC Standard Research Studentship (DTP), grant number EP/T5179X/1.

Acknowledgements We would like to thank Alain Finkel for pointing out an error in the introduction. We would also like to thank our anonymous reviewers for their detailed comments.

1 Introduction

Petri nets [22] are among the most fundamental formalisms for modelling processes. They are defined by a finite set of places and a finite set of transitions. A configuration of a Petri net, known as a marking, is a vector of dimension equal to the number of places, with entries equal to the number of tokens in particular places. Transitions change markings by consuming and producing tokens in places. For an example, see Figure 1.

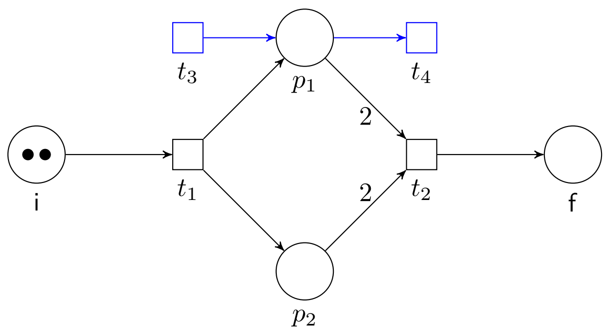

Figure 1 An example Petri net with four places i, p_1, p_2, f and four transitions t_1, t_2, t_3, t_4. Arcs pointing to transitions consume tokens from the respective places, and arcs pointing away from transitions produce tokens in the respective places. Arcs without labels denote single token consumption or production. Other labels, such as "2" in this example, are explicit. Initially, the marking can be represented by the vector $(2, 0, 0, 0)$ where there are two tokens in i, and no tokens in the other three places. At this marking, the transition t_2 cannot be fired as it needs to consume 2 tokens from each of p_1 and p_2. One can see that by firing a sequence of transitions (t_1, t_1, t_2) we reach the marking $(0, 0, 0, 1)$. The transitions t_3 and t_4 are highlighted (in blue) because t_3 does not consume any tokens and t_4 does not produce any tokens.

The central decision problem for Petri nets is the *reachability problem*. Given a Petri net, an initial marking, and a target marking, reachability asks whether there is a run between the two markings. Reachability in Petri nets is a decision problem with non-primitive recursive complexity [17, 18], recently it has been shown to be Ackermann-complete [7, 8]. The *coverability problem*, a relaxation of the reachability problem, asks whether there is a run that reaches a marking at least as great as the target marking. Coverability is provably simpler than reachability and it is known to be EXPSPACE-complete [20, 23, 16]. In the example (Figure 1), from the initial marking $(2, 0, 0, 0)$, one can reach $(0, 0, 0, 1)$, but one cannot reach $(0, 0, 1, 0)$. However, $(0, 0, 1, 0)$ can be covered since $(1, 1, 1, 0)$ can be reached.

In this paper, we consider Petri nets that are equipped with *resets* but are restricted to be *acyclic*. Resets are an extra feature of transitions that allow transitions to empty a subset of places. In modelling processes, resets offer the ability to express cancellation, which is important in many applications [29, Table 1]. Unfortunately, in general without any acyclicity restriction, for Petri nets with resets, reachability is undecidable [1, 9] and coverability is Ackermann-complete [24, 11]. Therefore, in order to observe the decidability of the reachability problem one needs to focus on a subclass of Petri nets with resets. A natural restriction is *acyclicity* that applies to the graph representation of the Petri net. For example, observe that the Petri net in Figure 1 is acyclic since the arcs do not induce any cycles between the places and transitions. Both reachability and coverability in acyclic Petri nets are NP-complete [19]. The NP upper bound is straightforward: it suffices to guess how many times each transition is fired in the run. It is always possible to transform this guess into an actual run by sorting the transitions in a topological order induced by the acyclic structure. As far as we know, acyclic Petri nets with resets have not been studied previously; they are a natural candidate for an expressive yet tractable class of Petri nets. We remark that the NP upper bound argument for reachability does not translate to the model with resets; changing the order of the resets does not preserve the reached marking.

We study Petri nets and their popular subclass *workflow nets* [28]. Workflow nets are Petri nets that have two special places, an input place i and an output place f. The places and transitions are also restricted so that no tokens can be produced in i, no tokens can

	Coverability	Reachability
Acyclic workflow nets with resets	PSPACE-complete (Section 4.1)	PSPACE-complete (Section 3.1)
Acyclic Petri nets with resets	PSPACE-complete (Section 3.2)	Undecidable (Section 4.2)

Figure 2 A summary of our results. Section 4.1 contains the PSPACE lower bound and Section 3.1 and Section 3.2 contain the PSPACE upper bounds.

be consumed from f, and all places and transitions lie on paths from i to f. The Petri net in Figure 1 without the (blue) highlighted transitions, t_3 and t_4, is a workflow net. Many practical instances of Petri nets are workflow nets [10], which forbid unnatural behaviour. Workflow nets are well studied [30], also with resets [29]. The complexities of the reachability and coverability problems for workflow nets are the same as for Petri nets. Indeed, the special places i and f produce and consume the initial and target markings, respectively. By introducing additional "artificial" places, it is not challenging to ensure that all places and transitions are on some path from i to f. However, the last construction does not preserve acyclicity. It turns out that acyclic Petri nets are more involved than acyclic workflow nets. For example, while the set of markings reachable from the initial marking is always finite for acyclic workflow nets [25], this is not true for acyclic Petri nets. In Figure 1, place p_1 can contain arbitrarily many tokens by firing t_3. In contrast, the workflow net obtained by removing transitions t_3 and t_4 will never contain more than 2 tokens in any place.

Our results. We determine the complexity of reachability and coverability in both acyclic Petri nets with resets and acyclic workflow nets with resets. We prove that coverability in acyclic Petri nets with resets is PSPACE-complete. Further, we show that both reachability and coverability in acyclic workflow nets with resets are also PSPACE-complete. On the other hand, we prove that, rather surprisingly, reachability in acyclic Petri nets with resets is undecidable. A summary of our results is in Figure 2.

For reachability in acyclic workflow nets with resets, we argue that a place cannot contain more than an exponential number of tokens with respect to the size of the reachability instance. The proof is comparable to the proof of the NP upper bound for acyclic Petri nets: one can reorder the firing sequence of transitions according to a topological order induced by acyclicity.

▶ **Theorem 3.1.** *Reachability in acyclic workflow nets with resets is in* PSPACE.

For coverability in acyclic Petri nets with resets, we show that there are two cases for the number of tokens that a place may contain. A place may either take at most an exponential number M of tokens, or it can take an arbitrarily large number of tokens, represented by ω. By abstracting the space of markings to a subset of $\{0, 1, \ldots, M-1, M, \omega\}^n$, we can search for a coverability run in polynomial space.

▶ **Theorem 3.2.** *Coverability in acyclic Petri nets with resets is in* PSPACE.

We complement these upper bounds with matching lower bounds. We show that coverability in acyclic workflow nets with resets requires polynomial space via a polynomial time reduction from QSAT. In the reduction, we construct an acyclic workflow net with resets that simulates assignments to the quantified variables (using places whose non-emptiness corresponds to the satisfaction of a literal) and checks that the formula evaluates to true for each assignment (using places whose non-emptiness corresponds to the satisfaction of a clause).

▶ **Theorem 4.2.** *Coverability in acyclic workflow nets with resets is* PSPACE*-hard.*

These three results allow us to conclude that coverability in acyclic Petri nets with resets and both coverability and reachability in acyclic workflow nets with resets are PSPACE-complete problems. We contrast this with the undecidability of reachability in acyclic Petri net with resets. Our proof is a reduction from reachability in general Petri nets with resets, which is known to be undecidable [1]. The core of our proof is the ability to simulate transitions whose arcs are not acyclic.

▶ **Theorem 4.13.** *Reachability in acyclic Petri nets with resets is undecidable.*

Related work. For workflow nets, a central decision problem is the *soundness* problem. An instance of soundness usually fixes the initial and target markings to only have one token in i and f, respectively. The soundness problem asks whether every marking reachable from the initial marking can then go on to reach the target marking. For workflow nets, it is known that soundness reduces to reachability [28], and an optimal algorithm for soundness (which does not rely on reachability) was only recently presented [3]. Variants of reachability and coverability have also been used as relaxations to implement soundness [26, 4]. Thus, we expect this work to provide a good background to study soundness on acyclic workflow nets with resets in the future.

In order to obtain decidability for the reachability problem on Petri nets with resets we both restrict the class to workflow nets and enforce acyclicity. However, instead of relaxing the class of Petri nets, one could allow the places to contain a negative number of tokens. Reachability in this relaxed model is called *integer reachability* and is known to be in NP for Petri nets[1], even with resets [6].

There are many extensions of Petri nets other than adding resets. We would like to highlight one extension in particular: Petri nets with transfers. Similar to resets, transfers move all the tokens from one place to another (instead of just removing them) [1]. Transfers allow the modelling of some properties of C programs [15]. For Petri nets with transfers, reachability in undecidable [1], but, coverability is decidable [9]. More generally, Petri nets with transfers and Petri nets with resets are examples of affine Petri nets [27, 12]. The previously mentioned integer reachability problem has been studied for this broad class of Petri nets [2, 5]. Consequently, integer reachability in Petri nets with transfers is in PSPACE [2]. As far as we know, reachability and coverability have not been considered for acyclic Petri nets with transfers or acyclic affine Petri nets, which we leave as possible future work.

2 Preliminaries

Let $\mathbb{Z}$ be the set of *integers* and $\mathbb{N}$ the set of *natural numbers* (nonnegative integers). Let ω stand for the first infinite cardinal, i.e. $\omega = |\mathbb{N}|$. Symbols $\mathbb{Z}_\omega$ and $\mathbb{N}_\omega$ denote the set of natural numbers and the set of integer numbers, each extended with ω, respectively. As usual, $|S|$ denotes the number of elements of a set S. We denote intervals by $[x, y] = \{z \in \mathbb{Z} \mid x \leq z \leq y\}$.

We use boldface to denote vectors, and we specify a vector by listing its coordinates, which are indexed using square brackets, in a tuple, so $\mathbf{v} = (\mathbf{v}[1], \ldots, \mathbf{v}[k])$. For two vectors $\mathbf{v}$ and $\mathbf{w}$ of equal dimension, we write $\mathbf{v} \geq \mathbf{w}$ if for every coordinate s we have $\mathbf{v}[s] \geq \mathbf{w}[s]$. If $\mathbf{v} \geq \mathbf{w}$ and $\mathbf{v} \neq \mathbf{w}$, then $\mathbf{v} > \mathbf{w}$; this partial order is called the pointwise order of vectors.

[1] Integer reachability is NP-complete for vector addition systems with states [13].

A vector $\mathbf{v}$ is non-negative if $\mathbf{v} \geq (0, 0, \ldots, 0)$. The norm $\|\cdot\|$ of a k-dimensional vector $\mathbf{v}$ is the sum of absolute values of its coordinates that are not equal to ω: $\|\mathbf{v}\| = \sum_{\mathbf{v}[i] \in \mathbb{N}} |\mathbf{v}[i]|$. We overload notation by saying that the norm $\|\cdot\|$ of a collection of vectors V is the sum of the norms of vectors in V, $\|V\| = \sum_{\mathbf{v} \in V} \|\mathbf{v}\|$.

Petri nets. A *Petri net* is a tuple (P, T, F) consisting of a finite set of *places* P, a finite set of *transitions* T (disjoint from P), and a function defining the *arcs* $F\colon (P \times T) \cup (T \times P) \to \mathbb{N}$. There is an arc from x to y for $(x, y) \in (P \times T) \cup (T \times P)$ if and only if $F(x, y) > 0$. In diagrams, this arc is *labelled* with the value of $F(x, y)$. One can view Petri nets as labelled graphs where $P \cup T$ is the set of nodes, and arcs are edges, labelled according to F. For example, in Figure 1 for transition t_1 we have $F(\mathsf{i}, t_1) = F(t_1, p_1) = F(t_1, p_2) = 1$ and all other values involving t_1 are 0. We can define a *path* in a Petri net as a sequence of places and transitions connected by arcs. A Petri net is *acyclic* if the graph of places and transitions with arcs is acyclic. The *norm* of a Petri net $\mathcal{N} = (P, T, F)$ is $\|\mathcal{N}\| = |P| \cdot |T| + \sum_{p \in P, t \in T} (F(t, p) + F(p, t))$.

▶ **Definition 2.1.** *A* Petri net with resets *is a tuple (P, T, F, R), where (P, T, F) is a Petri net and $R\colon T \to 2^P$ is a function defining* reset edges*. There is reset edge between a transition $t \in T$ and a place $p \in P$ if and only if $p \in R(t)$. A Petri net with resets (P, T, F, R) is an* acyclic Petri net with resets *if (P, T, F) is acyclic according to the definition above.*

Importantly, reset edges are *not* subject to the acyclicity restriction. We discuss this in more detail below, after the formal definition of the semantics.

The norm of a Petri net with resets is $\|(P, T, F, R)\| = \|(P, T, F)\| + \sum_{t \in T} |R(t)|$.

For a Petri net with resets (P, T, F, R), the *pre-vector* of a transition t is ${}^\bullet t\colon P \to \mathbb{N}$, where ${}^\bullet t[p] = F(p, t)$, and its *post-vector* is $t^\bullet\colon P \to \mathbb{N}$, where $t^\bullet[q] = F(t, q)$. We use similar notation for the *reset-operator* $t^\circ \subseteq P$, namely $t^\circ = R(t)$.

Let us define the semantics of Petri nets (with resets). The collection of *markings* of a Petri net with resets (P, T, F, R) is the set of all vectors in $\mathbb{N}^P$. Places are said to contain *tokens*, a finite resource that can be *consumed*, *produced*, and *reset* by transitions. For a given marking $\mathbf{m}$, a place p contains tokens if $\mathbf{m}[p] > 0$, otherwise it is *empty*. A transition t can be *fired* at a marking $\mathbf{m}$ if and only if $\mathbf{m} \geq {}^\bullet t$. The firing proceeds through the following phases (see Figure 3 for an example):

- first, tokens are *consumed*, which results in $\mathbf{m}' = \mathbf{m} - {}^\bullet t$;
- then, places are *reset*, which results in $\mathbf{m}''$ where $\mathbf{m}''[p] = 0$ for all $p \in t^\circ$ and $\mathbf{m}''[p] = \mathbf{m}'[p]$ for all $p \notin t^\circ$;
- finally, tokens are *produced*, which results in the new marking $\mathbf{n} = \mathbf{m}'' + t^\bullet$.

We write $\mathbf{m} \xrightarrow{t} \mathbf{n}$.

▶ Note. In the semantics of Petri nets with resets, whether or not a place under reset contains tokens does not affect whether the transition can be fired. This makes the effect of resets distinct from the usual consumption of tokens by a transition. Resets do not produce any tokens either. Thus, resets are considered "undirected", and we refer to reset *edges* (rather than arcs). For the sake of clarity, in all drawings of Petri nets with resets, the reset edges are undirected and will be coloured red to distinguish them further.

A *firing sequence* $\sigma = (t_1, t_2, \ldots, t_n)$ is a sequence of transitions. It forms a *run* from a marking $\mathbf{m}_0$ to a marking $\mathbf{m}_n$ if $\mathbf{m}_0 \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2 \xrightarrow{t_3} \cdots \xrightarrow{t_n} \mathbf{m}_n$ for some intermediate markings $\mathbf{m}_1, \ldots, \mathbf{m}_{n-1}$. The run is denoted $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}_n$. We also write $\mathbf{m} \xrightarrow{*} \mathbf{n}$ if there exists a run from $\mathbf{m}$ to $\mathbf{n}$; in this case we say that $\mathbf{n}$ is *reachable* from $\mathbf{m}$. Further, we say that a run $\mathbf{m} \xrightarrow{*} \mathbf{n}'$ *covers* $\mathbf{n}$ if $\mathbf{n}' \geq \mathbf{n}$. If such a σ exists. we say that $\mathbf{n}$ can be *covered* from $\mathbf{m}$.

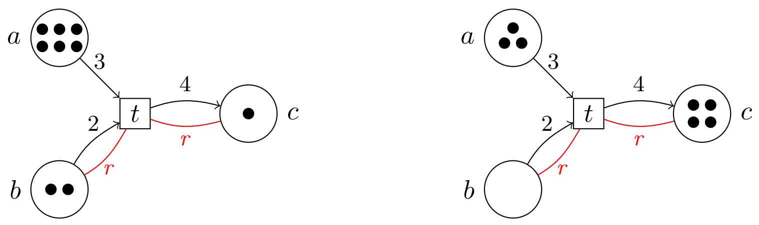

Figure 3 Two markings of an acyclic Petri net with resets with three places a, b, and c. Left: upon firing t from marking $(6, 2, 1)$ shown, 3 tokens are consumed from a and 2 tokens are consumed from b. Then, b and c are reset to 0 tokens, from 0 tokens and 1 token, respectively. Finally, 4 tokens are produced in c; this is the only number of tokens c can contain after t is fired. Right: marking $(3, 0, 4)$ is reached as the result of firing t.

▶ Note. Every Petri net can be seen as a Petri net with resets whose reset function is null, $R(t) = \emptyset$ for all t. So all definitions for Petri nets with resets naturally extend to Petri nets.

Workflow nets. A *workflow net* is a triple $(\mathcal{P}, \mathsf{i}, \mathsf{f})$ where $\mathcal{P}$ is a Petri net (P, T, F), $\mathsf{i} \in P$ is the *initial place*, $\mathsf{f} \in P$ is the *final place*, and all places and transitions lie on paths from i to f. A *workflow net with resets* is a triple $(\mathcal{R}, \mathsf{i}, \mathsf{f})$ where $\mathcal{R} = (P, T, F, R)$ is a Petri net with resets, and $((P, T, F), \mathsf{i}, \mathsf{f})$ is a workflow net. We say that a workflow net (with resets) $(\mathcal{N}, \mathsf{i}, \mathsf{f})$ is *acyclic* if the Petri net (with resets) $\mathcal{N}$ is acyclic. In Figure 1 the Petri net without transitions t_3 and t_4 is also a workflow net.

Decision problems. The following problems can be posed with any combination of added resets, acyclicity, and the workflow restriction.

Reachability in Petri nets

INPUT: A Petri net $\mathcal{N}$, an initial marking $\mathbf{m}$, and a target marking $\mathbf{n}$.
QUESTION: Does there exist a firing sequence σ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{n}$?

Coverability in Petri nets

INPUT: A Petri net $\mathcal{N}$, an initial marking $\mathbf{m}$, and a target marking $\mathbf{n}$.
QUESTION: Does there exist a firing sequence σ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{n}'$, where $\mathbf{n}' \geq \mathbf{n}$?

To give *instances* of these problems, we use tuples $(\mathcal{N}, \mathbf{m}, \mathbf{n})$. The *norm* of an instance $\|(\mathcal{N}, \mathbf{m}, \mathbf{n})\| = \|\mathcal{N}\| + \|\mathbf{m}\| + \|\mathbf{n}\|$. Depending on whether the arc weights are written in unary or binary, the *bit size* of the input is polynomial in the norm or logarithmic in the norm, respectively. Unary encoding suffices for our PSPACE lower bound (Theorem 4.2); see Lemma 4.1. Both of our PSPACE upper bounds (Theorem 3.1 and Theorem 3.2) hold even when the arc weights are binary-encoded. The undecidability result (Theorem 4.13) is independent of the encoding.

3 Upper Bounds

3.1 Reachability in Acyclic Workflow Nets with Resets

▶ **Theorem 3.1.** *Reachability in acyclic workflow nets with resets is in* PSPACE.

Proof. We rely on the simple property that reachable markings in acyclic workflow nets with resets are exponentially bounded. Let $\mathcal{R} = (P, T, F, R)$ be a given acyclic workflow net with resets and fix an initial marking $\mathbf{m}$. Consider the workflow net $\mathcal{W} = (P, T, F)$ that is just

$\mathcal{R}$ with the resets removed. Suppose from a marking $\mathbf{p}$ in $\mathcal{R}$, firing a transition t leads to marking $\mathbf{q}$. Clearly with the resets removed, firing t from $\mathbf{p}$ in $\mathcal{W}$ leads to a marking $\mathbf{q}'$ and $\mathbf{q}' \geq \mathbf{q}$. Notice also that the removal of resets does not alter whether or not a transition can be fired, if a transition can be fired from $\mathbf{p}$ then it can be fired from any $\mathbf{p}' \geq \mathbf{p}$. It follows that if $\mathbf{m} \xrightarrow{\pi} \mathbf{n}$ in $\mathcal{R}$, then $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ in $\mathcal{W}$ for some $\mathbf{n}' \geq \mathbf{n}$. With this in mind, it suffices to argue that any reachable marking in $\mathcal{W}$ can be stored in polynomial space, relative to the norms of $\mathbf{m}$ and $\mathcal{W}$.

Let $m = \|\mathcal{R}\| + \|\mathbf{m}\|$. We prove that if $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ in $\mathcal{W}$, then $\|\mathbf{n}'\| \leq m^{n+1}$, where n is the number of distinct transitions occurring in the firing sequence π. Since $\mathcal{W}$ is acyclic, there is a topological order on (the sources of) the transitions, and π can be permuted to respect this order (cf. [14]). Every transition in a workflow net must consume at least one token, so it follows that the i-th distinct transition can be fired at most m^i many times, resulting in a marking of norm at most m^{i+1}. Therefore, the norm of the largest possible marking is m^{n+1} and since $n \leq |T|$, all markings observed in the (permuted) run can be written down using polynomially many bits. Hence, reachability in acyclic workflow nets with resets can be decided using polynomial space. ◀

3.2 Coverability in Acyclic Petri Nets with Resets

▶ **Theorem 3.2.** *Coverability in acyclic Petri nets with resets is in* PSPACE.

We fix our attention on an instance $(\mathcal{P}, \mathbf{m}, \mathbf{n})$ of coverability in acyclic Petri nets with resets. Our approach can be summarised in two parts. First, we construct another infinite-state system $\mathcal{N}$ by modifying $\mathcal{P}$, that is much like a Petri net. The difference is that the places of $\mathcal{N}$ may contain an "infinite" number of tokens, denoted ω. Importantly, we will argue that $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{P}$ if and only if $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{N}$. Second, we show that the set of markings reachable from $\mathbf{m}$ in $\mathcal{N}$ has cardinality exponential in $\|\mathcal{N}\|$ and $\|\mathbf{m}\|$. Together, this allows us to decide, in polynomial space, this instance of coverability in acyclic Petri nets with resets.

We say that a transition t is *generating from a marking* $\mathbf{r}$ if it only consumes tokens from places which contain ω tokens, more precisely, for each place p such that ${}^\bullet t[p] > 0$, we have $\mathbf{r}[p] = \omega$. In other words, a generating transition can only decrease the number of tokens in some place by resetting it; notice that consuming a finite number of tokens from a place that contains ω tokens leaves ω tokens is that place. Suppose that $\mathbf{p} \xrightarrow{t} \mathbf{q} \xrightarrow{t} \mathbf{r}$, where t is a generating transition in $\mathbf{p}$, then $\mathbf{r} \geq \mathbf{q}$. Indeed, if some place is reset by t then by immediately firing t again, the number of tokens in such a place does not decrease below zero. By definition, the number of tokens in places that t only consumes from is ω, both before and after firing a generating transition t. Finally, the number of tokens in all other places can only increase after firing t again. By firing t an arbitrary number of times, the places where t only produces tokens to will then contain ω many tokens.

Formally, $\mathcal{N}$ is the same object as $\mathcal{P}$: it consists of the same sets of places, transitions, and resets, but its semantics differs. A marking $\mathbf{m}$ of $\mathcal{N}$ is allowed to have ω tokens in its places, so $\mathbf{m} \in \mathbb{N}_\omega^n$, where n is the number of places. Recall that ω denotes the first infinite cardinal, so $\omega + z = \omega$ for all $z \in \mathbb{Z}$. To define the semantics of $\mathcal{N}$, we need to specify the behaviour of its transitions. Fix a marking $\mathbf{m}$. As is the case in $\mathcal{P}$, a transition t can be fired in $\mathcal{N}$ if, for every place p, $\mathbf{m}[p] \geq {}^\bullet t[p]$. The marking reached depends on whether t is generating from $\mathbf{m}$. If t is not generating from $\mathbf{m}$, then its behaviour is defined as it was in $\mathcal{P}$; first subtract ${}^\bullet t$, then perform the resets, and lastly add $t^\bullet$. Otherwise, if t is generating from $\mathbf{m}$, then $\mathbf{m} \xrightarrow{t} \mathbf{n}$ is defined so that

$$\mathbf{n}[p] = \begin{cases} \omega & \text{if } p \notin t^\circ, \text{ and either } t^\bullet[p] \geq 1 \text{ or } \mathbf{m}[p] = \omega; \\ t^\bullet[p] & \text{if } p \in t^\circ; \\ \mathbf{m}[p] & \text{otherwise.} \end{cases}$$

Intuitively, the transition is applied arbitrarily many times producing ω tokens to some places, whenever it is possible. Claim 3.3 allows us to instead decide the coverability instance in $\mathcal{N}$ with the abstracted space of configurations.

▷ Claim 3.3. Let $\mathbf{m}, \mathbf{n} \in \mathbb{N}^n$. Then $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{P}$ if and only if $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{N}$.

Recall the norm of a marking $\|\mathbf{v}\| = \sum_{\mathbf{v}[p]\in\mathbb{N}} |\mathbf{v}[p]|$. Claim 3.4, shows that because $\mathcal{N}$ is acyclic, only markings with an exponential norm can be reached. Note, critically, that places containing ω tokens do not contribute to the norm.

▷ Claim 3.4. Let k be the greatest number of tokens produced by a transition in $\mathcal{P}$ and let $C = \|\mathbf{m}\|$. If $\mathbf{m} \xrightarrow{*} \mathbf{v}$ in $\mathcal{N}$, then $\|\mathbf{v}\| \leq C \cdot k^n$.

Proof of Theorem 3.2. By Claim 3.3, it suffices to show that coverability in the modified acyclic Petri net with resets $\mathcal{N}$ can be decided in polynomial space. We can do this by non-deterministically exploring the markings that are reachable from $\mathbf{m}$ in $\mathcal{N}$. Given Claim 3.4, if $\mathbf{v}$ is reachable from $\mathbf{m}$ in $\mathcal{N}$, then $\|\mathbf{v}\| \leq C \cdot k^n$. These reachable markings can be written down using polynomially many bits since n is the number of places, k is the greatest number of tokens produced by a place, and C is the norm of $\mathbf{m}$. Thus, coverability in acyclic Petri nets with resets is in PSPACE. ◀

4 Lower Bounds

4.1 Coverability in Acyclic Workflow Nets with Resets

With the coverability objective, binary-encoded transitions can be weakly simulated by unary-encoded transitions. We do this for convenience, since the later reductions can be more succinctly presented with binary-encoded transitions.

▶ **Lemma 4.1.** *Given be an instance of coverability in acyclic workflow nets with resets with binary-encoded transitions $I = (\mathcal{B}, \mathbf{m}, \mathbf{n})$, one can construct, in polynomial time, an instance of coverability in acyclic workflow nets with resets $I' = (\mathcal{U}, \mathbf{x}, \mathbf{y})$ with unary-encoded transitions such that I is positive if and only if I' is positive.*

In this section we prove the following theorem. The proofs of all auxiliary claims and lemmata can be found in the full version of the paper.

▶ **Theorem 4.2.** *Coverability in acyclic workflow nets with resets is* PSPACE*-hard.*

Proof Approach

We will reduce from the QSAT problem.

QSAT

INPUT: A QBF φ in CNF over variables $y_1, x_1, \ldots, y_k, x_k$.
QUESTION: Does $\forall y_1 \, \exists x_1 \, \ldots \, \forall y_k \, \exists x_k : \varphi(y_1, x_1, \ldots, y_k, x_k)$ evaluate to true?

Given a Quantified Boolean Formula (QBF) φ in Conjunctive Normal Form (CNF), we will construct an acyclic workflow net with resets $\mathcal{W}$ that mimics the exhaustive approach to verifying φ. There will be a collection of places that represent an assignment to the variables $y_1, x_1, \ldots, y_k, x_k$. There will be transitions that consume tokens from these places and produce tokens into a component of $\mathcal{W}$ that is used to test whether the current assignment is satisfying. If the current assignment is satisfying, then one token can be produced to some final place which counts the number of satisfying assigments observed. The places representing an assignment are controlled by a series of gadgets that exhaustively iterate through each possible assignment to the universal variables and allow for nondeterministic assignment to the existential variables. A marking in which the final place contains 2^k tokens can only be reached if and only if every considered assignment has been checked to be satisfying. A detailed description of the coverability instance follows. The proof of correctness consists of two parts.

First, we will verify that the QBF evaluates to true given that coverability holds. We achieve this via an inductive argument that tracks the simulated assignments to variables over parts of the run witnessing coverability.

In the second part, we would like to recover a firing sequence for coverability if the QBF evaluates to true. We achieve this by using (partial) assignments to variables in the QBF to inform which transitions need be fired to make progress towards the final marking.

Construction of the Acyclic Workflow Net with Resets

For this section, we focus our attention on a QBF

$$\forall y_1 \, \exists x_1 \, \forall y_2 \, \exists x_2 \, \ldots \, \forall y_k \, \exists x_k : \varphi(y_1, x_1, y_2, x_2, \ldots, y_k, x_k).$$

We remark that we can add "dummy" clauses $(\overline{y}_i \vee y_i)$ and $(\overline{x}_i \vee x_i)$ for each $i \in [1, k]$ to φ without changing any valuation.

For the proof of Theorem 4.2, we construct an acyclic workflow net with resets $\mathcal{W} = (P, T, F, R)$ from the QBF; we first list the places and transitions including resets of $\mathcal{W}$. See Figure 4 for an example.

The places. There is a place for each literal: for every $i \in [1, k]$, there is b_i for y_i, $\overline{b}_i$ for $\overline{y}_i$, a_i for x_i, and $\overline{a}_i$ for $\overline{x}_i$. Let L denote the set of the literal places. The non-emptiness of the place $\overline{b}_i$, for example, will represent assigning false to the variable y_i.

There is a place for each clause: for every $j \in [1, m]$, there is c_j for the j-th clause. Furthermore, for every $i \in [1, k]$, there is d_{y_i} for the dummy clause $(y_i \vee \overline{y}_i)$ and there is d_{x_i} for the dummy clause $(x_i \vee \overline{x}_i)$. All *clause places* $c_1, \ldots, c_m, d_{x_1}, d_{y_1}, \ldots, d_{x_k}, d_{y_k}$ are distinct; the set comprising them is denoted C. The non-emptiness of a clause place $c \in C$ will represent whether the corresponding clause has been satisfied.

For each $i \in [1, k]$, there is a *holding place* h_i and a *waiting place* w_i for each universally quantified variable, as well as a *decision place* v_i for each existentially quantified variable. If the holding place h_i contains a token, one should think that the universally quantified variable y_i and all subsequent variables $x_i, y_{i+1}, x_{i+1}, \ldots, y_k, x_k$ have not yet been assigned. The waiting place w_i will contain a token if the universally quantified variable y_i is currently assigned false. The decision place v_i contains a token after the truth assignment of the prior universally quantified variable y_i has completed, but the existentially quantified variable x_i has not yet received a value. The literal places, holding places, waiting places, decision places, and dummy clause places are grouped into *gadgets*. There are k *universal gadgets* $U_i = \{h_i, w_i, b_i, \overline{b}_i, d_{y_i}\}$ and k *existential gadgets* $E_i = \{v_i, a_i, \overline{a}_i, d_{x_i}\}$.

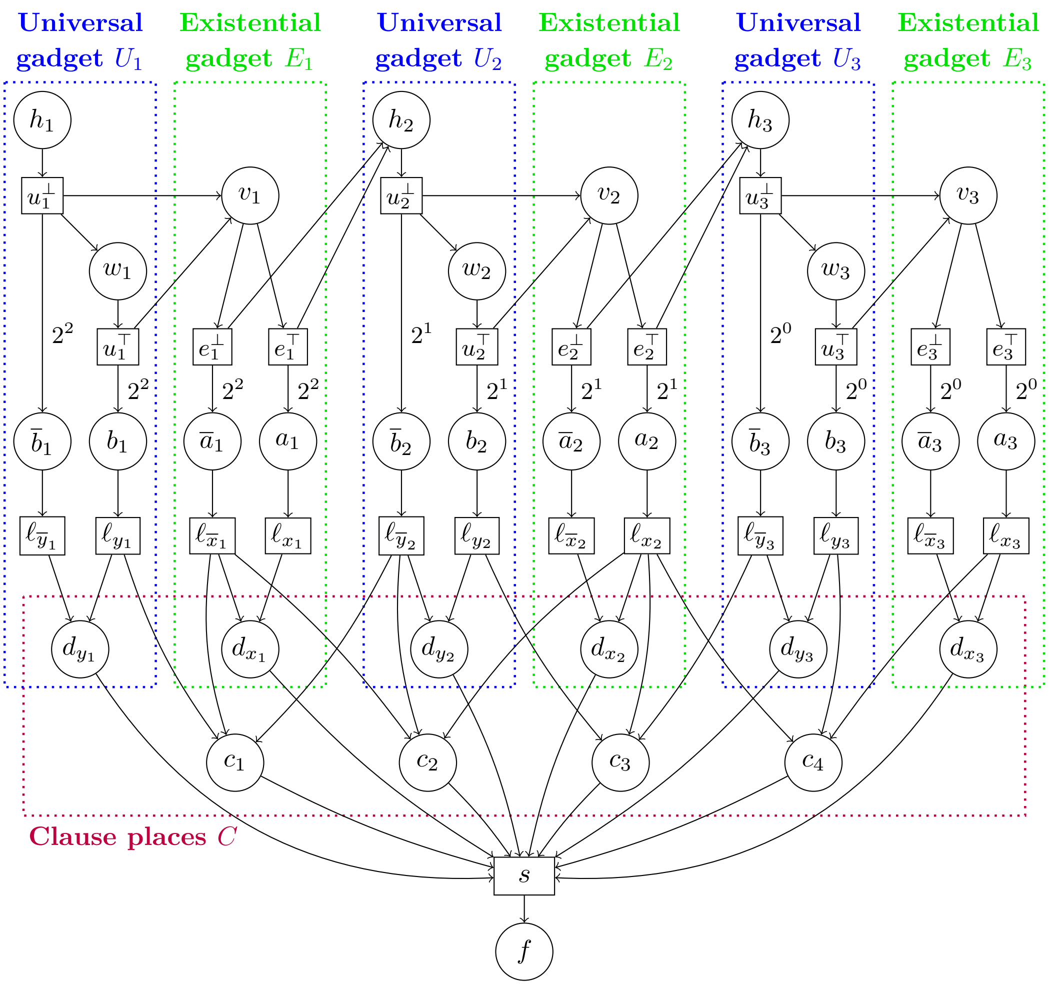

Figure 4 The acyclic workflow net with resets $\mathcal{W}$, drawn without resets for sake of clarity, for the QBF $\forall y_1 \exists x_1 \forall y_2 \exists x_2 \forall y_3 \exists x_3 : \varphi(y_1, x_1, y_2, x_2, y_3, x_3)$ where $\varphi(y_1, x_1, y_2, x_2, y_3, x_3) = (y_1 \vee \overline{x}_1 \vee \overline{y}_2) \wedge (\overline{x}_1 \vee \overline{y}_2 \vee x_2) \wedge (y_2 \vee x_2 \vee \overline{y}_3) \wedge (x_2 \vee y_3 \vee x_3) \wedge (y_1 \vee \overline{y}_1) \wedge (x_1 \vee \overline{x}_1) \wedge (y_2 \vee \overline{y}_2) \wedge (x_2 \vee \overline{x}_2) \wedge (y_3 \vee \overline{y}_3) \wedge (x_3 \vee \overline{x}_3)$. All universal and existential control transitions reset all later occurring places in the universal and existential gadgets and in all clause places. The loading transitions reset all later occurring dummy clause places. The satisfaction transition resets all clause places.

Finally, there is a place f which counts the number of assignments that have been verified to satisfy the QBF.

The initial place i of the workflow is h_1 and the final place f of the workflow is f.

The transitions. Here, binary-encoded transitions are used, see Lemma 4.1. The resets will be specified later.

Inside the universal gadget U_i, there are two *universal control transitions* $u_i^\perp$ and $u_i^\top$. Firing $u_i^\perp$ corresponds to setting y_i to false, and firing $u_i^\top$ corresponds to setting y_i to true. The transition $u_i^\perp$ consumes one token from h_i, produces one token to w_i, produces one token to v_i, and produces 2^{k-i} tokens to $\overline{b}_i$; the transition $u_i^\top$ consumes one token from w_i, produces one token to v_i, and produces 2^{k-i} tokens to b_i.

Inside the existential gadget E_i, there are two *existential control transitions* $e_i^\perp$ and $e_i^\top$. Firing $e_i^\perp$ corresponds to setting x_i to false, and firing $e_i^\top$ corresponds to setting x_i to true. The transition $e_i^\perp$ consumes one token from v_i, produces 2^{k-i} tokens to $\overline{a}_i$, and produces one token to h_{i+1}; similarly, the transition $e_i^\top$ consumes one token from v_i, produces 2^{k-i} tokens to a_i, and produces one token to h_{i+1}.

Informally, the i-th universal or existential controlling transitions produce 2^{k-i} tokens to places $\overline{b}_i$, b_i, $\overline{a}_i$, and a_i so that their values are "remembered" whilst the inner quantified variables have their assignments exhausted.

Connecting the universal and existential gadgets to the clause places are a series of *loading transitions.* There is a loading transition for each literal; for each $i \in [1, k]$, there are transitions $\ell_{\overline{y}_i}$, ℓ_{y_i}, $\ell_{\overline{x}_i}$, and ℓ_{x_i}. The loading transition ℓ_{y_i}, for example, consumes a token from the place b_i and produces a token to each clause place corresponding to a clause containing the literal y_i, including the dummy clause place d_{y_i}.

There is a *satisfaction transition* s that consumes a token from each of the clause places and produces a token into a final place f. Intuitively, s can only be fired when all of the clauses have been satisfied (and f is used to count the number of satisfying assignments).

Ordering places and transitions. The following linear ordering *earlier than* (denoted $\prec$) on $P \cup T$ shows that $\mathcal{W}$ is acyclic:

$$h_1, u_1^\perp, w_1, u_1^\top, \overline{b}_1, b_1, v_1, e_1^\perp, e_1^\top, \overline{a}_1, a_1, \ldots, h_k, u_k^\perp, w_k, u_k^\top, \overline{b}_k, b_k, v_k, e_k^\perp, e_k^\top, \overline{a}_k, a_k,$$
$$\ell_{\overline{y}_1}, \ell_{y_1}, \ell_{\overline{x}_1}, \ell_{x_1}, \ldots, \ell_{\overline{y}_k}, \ell_{y_k}, \ell_{\overline{x}_k}, \ell_{x_k}, d_{y_1}, d_{x_1}, \ldots, d_{y_k}, d_{x_k}, c_1, \ldots, c_m, s, f.$$

The resets. The universal and existential control transitions reset all *later* occurring places in the universal gadgets and existential gadgets and *all* dummy clause places. This also includes the places corresponding to the literals; for example, $u_i^\perp$ and $u_i^\top$ reset both $\overline{b}_i$ and b_i, so it is always true that either $\overline{b}_i$ and b_i is empty.

This effectively forces the universal and existential control transitions to be fired in sequence: $u_1^\perp$ or $u_1^\top$, then $e_1^\perp$ or $e_1^\top$, then $u_2^\perp$ or $u_2^\top$, etc., until $e_k^\perp$ or $e_k^\top$ is fired.

The loading transitions reset all later occurring dummy clause places. For example, ℓ_{y_i} resets $d_{x_i}, d_{y_{i+1}}, d_{x_{i+1}}, \ldots, d_{y_k}, d_{x_k}$. Similarly, this forces the loading transitions to also be fired in sequence: $\ell_{\overline{y}_1}$ or ℓ_{y_1}, then $\ell_{\overline{x}_1}$ or ℓ_{x_1}, then $\ell_{\overline{y}_2}$ or ℓ_{y_2}, until $\ell_{\overline{x}_k}$ or ℓ_{x_k} is fired. This is due to the fact that all dummy places must be non-empty to fire the satisfaction transition.

Finally, the satisfaction transition resets all clause places. It could be the case that a clause contains two true literals under an assignment, so the clause place contains two tokens. It is necessary to clear such a place. Note that the final place f cannot be reset.

Coverability instance $(\mathcal{W}, \mathbf{m}, \mathbf{n})$. We have just defined the acyclic workflow net with resets $\mathcal{W}$. The initial marking $\mathbf{m}$ only has one token in the initial place; $\mathbf{m}[h_1] = 1$ and, for all $p \in P \setminus \{h_1\}, \mathbf{m}[p] = 0$. The target marking $\mathbf{n}$ only has 2^k tokens in the final place; $\mathbf{n}[f] = 2^k$ and, for all $p \in P \setminus \{f\}, \mathbf{n}[p] = 0$.

Part One: Coverability implies QBF is true

We would like to prove an inductive statement of the following, informally described, kind. Consider any run from the initial marking that covers the target marking. Let σ be an infix of this run from $\mathbf{p}$ to $\mathbf{q}$, and let i be a number in $[0, k]$ such that 2^i divides $\mathbf{p}[f]$ and that $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$. This means that σ fires the satisfaction transition, s, 2^i many times. Then the following (partial) QBF is true:

$$\forall y_{k-i+1}\ \exists x_{k-i+1} \ldots \forall y_k\ \exists x_k : \varphi(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \ldots, y_k, x_k).$$

Here $(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}) \in \{0,1\}^{2(k-i)}$ are determined by $\mathbf{p}$.

To realise this plan, we need several ingredients. For the base case of the induction, $i = 0$: σ only fires s once. We will determine $(\beta_1, \alpha_1, \beta_2, \alpha_2, \ldots, \beta_k, \alpha_k) \in \{0,1\}^{2k}$ based on $\mathbf{p}$, in particular on which of the places $\overline{b}_i$ and b_i, as well as $\overline{a}_i$ and a_i, are non-empty in $\mathbf{p}$. Note that it might not be sufficient to consider only the marking $\mathbf{p}$ since this could be, for instance, the initial marking $\mathbf{m}$, which has all places empty, bar h_1. So the "existential decisions" that determine $\alpha_1, \alpha_2, \ldots, \alpha_k$ need to be found from a prefix of σ.

For the inductive step, $i > 0$: the infix σ fires the satisfaction transition 2^i times. We will split σ in two: σ_0 and σ_1. We will use the inductive hypothesis on both subruns. For this to work, we will show that the partial assignments

$$(\beta_1, \alpha_1, \ldots, \beta_{k-i+1}, \alpha_{k-i+1}) \in \{0,1\}^{2(k-i+1)} \qquad \text{and}$$
$$(\beta'_1, \alpha'_1, \ldots, \beta'_{k-i+1}, \alpha'_{k-i+1}) \in \{0,1\}^{2(k-i+1)},$$

which are determined based on each half of the run, satisfy the constraints

$$\beta_1 = \beta'_1,\ \alpha_1 = \alpha'_1,\ \ldots,\ \beta_{k-i} = \beta'_{k-i},\ \alpha_{k-i} = \alpha'_{k-i}, \quad \beta_{k-i+1} = 0, \quad \text{and} \quad \beta'_{k-i+1} = 1.$$

Informally speaking, these partial assignments are complementary with respect to the i-th innermost universally quantified variable. Note that the index variable i is reused in a variety of contexts throughout the following claims.

Properties of markings.

▷ Claim 4.3. If $\mathbf{v}$ is reachable from $\mathbf{m}$, then for every $i \in [1,k]$, $\mathbf{v}[\overline{b}_i] = 0$ or $\mathbf{v}[b_i] = 0$, and $\mathbf{v}[\overline{a}_i] = 0$ or $\mathbf{v}[a_i] = 0$.

▷ Claim 4.4. If $\mathbf{p} \xrightarrow{t} \mathbf{q}$, then $\mathbf{q}[f] - \mathbf{p}[f] \in \{0,1\}$.

Let us define, for each $i \in [1,k]$, two functions $g_i, g'_i : \mathbb{N}^P \to \mathbb{N}$ that map a marking to a natural number. We will use these functions to define a collection of *good* markings.

$$g_i(\mathbf{v}) \coloneqq \mathbf{v}[f] + \mathbf{v}[\overline{b}_i] + \mathbf{v}[b_i] + \mathbf{v}[d_{y_i}] + \sum_{j=1}^{i} 2^{k-j} \cdot (2\mathbf{v}[h_j] + \mathbf{v}[w_j] + \mathbf{v}[v_j]) - 2^{k-i} \cdot \mathbf{v}[v_i]$$
$$g'_i(\mathbf{v}) \coloneqq \mathbf{v}[f] + \mathbf{v}[\overline{a}_i] + \mathbf{v}[a_i] + \mathbf{v}[d_{x_i}] + \sum_{j=1}^{i} 2^{k-j} \cdot (2\mathbf{v}[h_j] + \mathbf{v}[w_j] + \mathbf{v}[v_j])$$

▶ **Definition 4.5** (Good marking). *A marking $\mathbf{v}$ is* good *if for each $i \in [1,k]$, $g_i(\mathbf{v}) = 2^k$ and $g'_i(\mathbf{v}) = 2^k$. A marking is* bad *if it is not good.*

Roughly speaking, a marking is good if no tokens in the universal gadgets U_i and no tokens in the existential gadgets E_i have been lost due to a reset. We discuss good markings in more detail in the full version of the paper.

▷ Claim 4.6. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$, then $g_i(\mathbf{p}) \geq g_i(\mathbf{q})$ and $g'_i(\mathbf{p}) \geq g'_i(\mathbf{q})$ for each $i \in [1,k]$.

▷ Claim 4.7. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$, where $\mathbf{p}$ is reachable from $\mathbf{m}$. If $\mathbf{q}$ is good, then $\mathbf{p}$ is good.

Given Claim 4.7 and since the target marking $\mathbf{n}$ is good, only good markings can be observed on a covering run from the initial marking $\mathbf{m}$. From this, we know that if a bad marking is ever reached, the target marking cannot be covered.

▷ Claim 4.8. If $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ where $\mathbf{n}' \geq \mathbf{n}$, then $\mathbf{n}' = \mathbf{n}$.

The following claim shows that resetting any non-empty place in any of the universal or existential gadgets results in a bad marking. Recall $\prec$, the previously defined *earlier than* ordering of places and transitions.

▷ Claim 4.9. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$ where $\mathbf{p}$ is reachable and $t \in \{u_i^\perp, u_i^\top, e_i^\perp, e_i^\top : i \in [1,k]\}$. If there exists $p \in U_1 \cup E_1 \cup \cdots \cup U_k \cup E_k$ such that $t \prec p$ and $\mathbf{p}[p] \geq 1$, then $\mathbf{q}$ is bad.

Extracting Assignments from Markings

We will now explain the relationship between markings and partial assignments. For a good marking $\mathbf{v}$, let $\mathrm{val}(\mathbf{v})$ be the vector $(\beta_1, \alpha_1, \beta_2, \alpha_2, \ldots, \beta_k, \alpha_k) \in \{0,1,?\}^{2k}$ such that

$$\beta_i := \begin{cases} 0 & \mathbf{v}[\overline{b}_i] \geq 1 \\ 1 & \mathbf{v}[b_i] \geq 1 \\ ? & \text{otherwise,} \end{cases} \quad \text{and} \quad \alpha_i := \begin{cases} 0 & \mathbf{v}[\overline{a}_i] \geq 1 \\ 1 & \mathbf{v}[a_i] \geq 1 \\ ? & \text{otherwise.} \end{cases}$$

The intention is that, for every $i \in [1,k]$, β_i and α_i correspond to the values of the Boolean variables y_i and x_i, respectively. Note that Claim 4.3 ensures that β_i and α_i are well-defined, since, for example, $\overline{b}_i$ and b_i cannot both be non-empty in a reachable marking. Notice that not all good markings correspond to fully defined variable assignments, but only those in which all h_i and v_i are empty. We will see that $\mathbf{p}[h_i] = \mathbf{p}[v_i] = 0$ implies that either $\overline{b}_i$ or b_i and either $\overline{a}_i$ or a_i are non-empty, except for right at the end, for example when the target marking $\mathbf{n}$ is reached. To take an example, if h_i contains a token, then neither $\overline{b}_i$ nor b_i will contain a token. Informally, this can be interpreted as thinking that the Boolean variable y_i has not yet been assigned its value; only after firing $u_i^\perp$ does it first get assigned false (before later being assigned true when $u_i^\top$ is eventually fired).

Recall that $C \subseteq P$ is the collection of clause places. We say that a marking $\mathbf{v}$ is *clause-free* if $\mathbf{v}[c] = 0$ for all $c \in C$.

▶ **Lemma 4.10.** *Fix $i \in [0,k]$ and suppose $\mathbf{p} \xrightarrow{\sigma} \mathbf{q}$ and the following properties hold:*

(1) *$\mathbf{p}$ is a clause-free marking that is reachable from $\mathbf{m}$,*
(2) *$\mathbf{n}$ is coverable from $\mathbf{q}$,*
(3) *2^i divides $\mathbf{p}[f]$ and $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$,*
(4) *the last transition of σ is s,*
(5) *for all $j \in [1, k-i]$, $\mathbf{p}[\overline{b}_i] + \mathbf{p}[b_i] \geq 2^i$ and $\mathbf{p}[\overline{a}_i] + \mathbf{p}[a_i] \geq 2^i$,*
(6) *if $i > 0$, then $\mathbf{p}[h_{k-i+1}] = 1$, and*
(7) *if $i > 0$, then, for all $p \in U_{k-i+1} \cup E_{k-i+1} \cup \cdots \cup U_k \cup E_k$ except h_{k-i+1}, $\mathbf{p}[p] = 0$.*

Let $\mathrm{val}(\mathbf{p}) = (\beta_1, \alpha_1, \ldots, \beta_k, \alpha_k)$. Then the following QBF evaluates to true:

$$\forall y_{k-i+1}\, \exists x_{k-i+1} \, \ldots \, \forall y_k \, \exists x_k : \varphi(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \ldots, y_k, x_k).$$

Moreover, σ does not fire transitions $u_1^\perp, u_1^\top, e_1^\perp, e_1^\top, \ldots, u_{k-i}^\perp, u_{k-i}^\top, e_{k-i}^\perp, e_{k-i}^\top$.

Part Two: QBF is true implies Coverability

Here we would like to recover a firing sequence for coverability if the QBF evaluates to true. Depending on the current assignment of the universally quantified variables, $y_1, \dots y_i$, and the already selected assignments of the existentially quantified variables $x_1, \dots, x_{i-1}$, one can use the truth of the QBF to determine whether x_i is assigned true or false, this informs which of the next existentially quantified transitions to fire.

▶ **Lemma 4.11.** *Fix $i \in [0, k]$ and suppose that for some $\beta_1, \alpha_1, \dots, \beta_{k-i}, \alpha_{k-i} \in \{0, 1\}$, the following QBF evaluates to true.*

$$\forall y_{k-i+1} \, \exists x_{k-i+1} \, \dots \, \forall y_k \, \exists x_k : \varphi(\beta_1, \alpha_1, \dots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \dots, y_k, x_k)$$

Let $\mathbf{p}$ be a marking such that, if $i > 0$ then $\mathbf{p}[h_{k-i+1}] = 1$, and for every $j \in [1, k-i]$,
(1) $\mathbf{p}[\bar{b}_j] \geq 2^i$ *if* $\beta_j = 0$, *otherwise* $\mathbf{p}[b_j] \geq 2^i$ *if* $\beta_j = 1$,
(2) $\mathbf{p}[\bar{a}_j] \geq 2^i$ *if* $\alpha_j = 0$, *otherwise* $\mathbf{p}[a_j] \geq 2^i$ *if* $\alpha_j = 1$.
Then there exists a firing sequence σ such that $\mathbf{p} \xrightarrow{\sigma} \mathbf{q}$ where $\mathbf{q}$ is a marking such that $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$ and for every $j \in [1, k-i]$,
(a) $\mathbf{q}[\bar{b}_j] + \mathbf{q}[b_j] = \mathbf{q}[\bar{b}_j] + \mathbf{q}[b_j] - 2^i$,
(b) $\mathbf{q}[\bar{a}_j] + \mathbf{q}[a_j] = \mathbf{q}[\bar{a}_j] + \mathbf{q}[a_j] - 2^i$, *and*
(c) $\mathbf{q}[h_j] = \mathbf{p}[h_j]$, $\mathbf{q}[w_j] = \mathbf{p}[w_j]$, *and* $\mathbf{q}[v_j] = \mathbf{p}[v_j]$.

Completing the proof

Proof of Theorem 4.2. The reduction from QSAT is already outlined above. Given an instance of QSAT that consists of a QBF φ over $y_1, x_1, \dots, y_k, x_k$, there exists an instance of coverability in acyclic workflow nets with resets $(\mathcal{W}, \mathbf{m}, \mathbf{n})$ such that $\forall y_1 \, \exists x_1 \, \dots \, \forall y_k \, \exists x_k$: $\varphi(y_1, x_1, \dots, y_k, x_k)$ evaluates to true if and only if $\mathbf{m} \xrightarrow{*} \mathbf{n}'$ in $\mathcal{W}$ where $\mathbf{n}' \geq \mathbf{n}$. The backwards implication is given by Lemma 4.10 with $i = k$, $\mathbf{p} = \mathbf{m}$, and $\mathbf{q} = \mathbf{n}'$. The forwards implication is given by Lemma 4.11 with $i = k$. ◀

▶ **Corollary 4.12.** *Reachability in acyclic workflow nets with resets and coverability in both acyclic Petri nets with resets and acyclic workflow net with resets are all* PSPACE*-complete.*

4.2 Reachability in Acyclic Petri Nets with Resets

In this section, we will prove that reachability in acyclic Petri nets with resets is undecidable. We reduce from reachability in Petri nets with zero tests, a problem that is well-known to be undecidable [1], following from the undecidability of reachability in counter machines [21]. A *Petri net with zero tests* is a tuple (P, T, F, Z), where (P, T, F) is a Petri net and $Z : T \to 2^P$ is a function defining the zero-test edges. A transition $t \in T$ zero-tests a place $p \in P$ if $p \in Z(t)$. Then t can be fired only if p is empty. As is the case for resets, an *acyclic Petri net with zero tests* does not subject zero-test edges to the acyclicity restriction.

▶ **Theorem 4.13.** *Reachability in acyclic Petri nets with resets is undecidable.*

The reduction is split into two parts. Lemma 4.14 shows how acyclic Petri nets with zero tests can simulate (not necessarily acyclic) Petri nets with zero tests. This requires using zero tests and transitions that do not consume tokens and transitions that do not produce tokens. Then, in Lemma 4.15, we show how acyclic Petri nets with resets can simulate acyclic Petri nets with zero tests. This requires some additional places and relies on the reachability objective to ensure that zero tests are simulated faithfully. The proof is very similar to the proof that reachability in Petri nets with resets is undecidable. We follow through with the construction to make it clear that acyclicity is preserved.

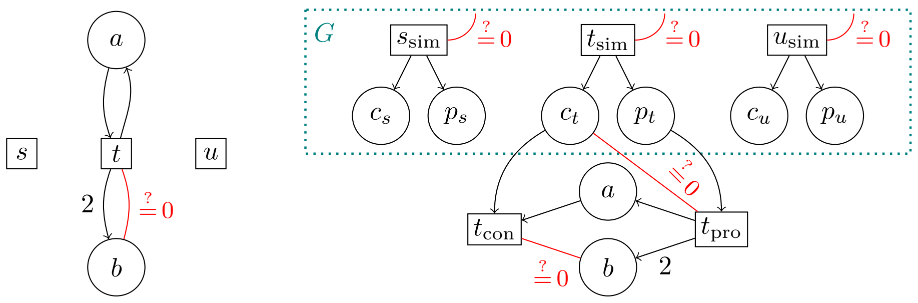

Figure 5 Suppose there is a Petri net with zero tests with transitions s, t, and u. Left: part of the Petri net with zero tests concerning the transition t. The consumptions and productions of s and u are not shown for simplicity. This Petri net is *not* acyclic since t both consumes 1 token from and produces 1 token to a. Right: part of the equivalent acyclic Petri net with zero tests. Places in G are shown, as well as transitions s_{sim}, t_{sim}, and u_{sim} for choosing the next transition to be fired. Since the consumptions and productions of s and u are not shown for simplicity, we also omit the corresponding s_{con}, s_{pro}, u_{con}, and u_{pro}. Importantly, zero-test edges between t_{con} and b, and between t_{pro} and c_t are not subject to the acyclicity restriction. Zero-test edges incident to s_{sim}, t_{sim}, and u_{sim} indicate that all places in G are zero-tested.

▶ **Lemma 4.14.** *The reachability problem in Petri nets with zero tests is reducible in logarithmic space to the reachability problem in acyclic Petri nets with zero tests.*

Proof. Let $\mathcal{P} = (P, T, F, Z)$ be a Petri net with zero tests. We will construct an acyclic Petri net with zero tests $\mathcal{Z} = (P', T', F', Z')$. For every transition $t \in T$, we will add two additional places c_t and p_t to the set of places. Formally, we define $G = \{c_t, p_t \mid t \in T\}$ and $P' = P \cup G$. For every transition $t \in T$, we create three transitions t_{sim}, t_{con}, and t_{pro}, so $T' = \{t_{\text{sim}}, t_{\text{con}}, t_{\text{pro}} \mid t \in T\}$. The intention is that firing $t \in T$ will be simulated by firing t_{sim}, t_{con}, and t_{pro} successively. Figure 5 illustrates the construction. To define the transitions in detail, fix $t \in T$.

- The transition t_{sim} simulates choosing t to be the next transition. Formally, $t_{\text{sim}}^{\bullet}[c_t] = t_{\text{sim}}^{\bullet}[p_t] = 1$, and $Z'(t_{\text{sim}}) = G$. Note that to fire t_{sim} all places in G must be empty, and upon firing t_{sim}, a token is placed in c_t and p_t. Thus no other transition s_{sim}, t_{sim}, or u_{sim} can be fired until the tokens in c_t and p_t are consumed.
- The transition t_{con} performs the token consumption and zero tests of t. Formally, ${}^{\bullet}t_{\text{con}}[c_t] = 1$, $Z'(t_{\text{con}}) = Z(t)$, and ${}^{\bullet}t_{\text{con}}[p] = {}^{\bullet}t[p]$ for every $p \in P$. The consumption of the token from c_t indicates that the consumptions and zero tests of t have been actioned.
- The transition t_{pro} performs the token productions of t. Formally, ${}^{\bullet}t_{\text{pro}}[p_t] = 1$, $Z'(t_{\text{con}}) = \{c_t\}$, and $t_{\text{pro}}^{\bullet}[p] = t^{\bullet}[p]$, for each $p \in P$. The consumption of the token from p_t indicates that the productions of t have been actioned. The zero test on c_t forces a firing order that mimics the semantics of firing t.

Indeed, after firing t_{sim} the only transition that can be fired is t_{con} since all other transitions require c_t to be empty or require a place in $G \setminus \{c_t\}$ to be non-empty. Then, after firing t_{con} the only transition that can be fired is t_{pro} since all other transitions either require p_t to be empty, or require a place in $G \setminus \{p_t\}$ to be non-empty.

Given a marking $\mathbf{v}$ over P, define $\mathbf{v}'$ over P' such that $\mathbf{v}'[p] = \mathbf{v}[p]$ for all $p \in P$ and $\mathbf{v}'[q] = 0$ for all $q \in G$. It follows that $\mathbf{m} \xrightarrow{*} \mathbf{n}$ in $\mathcal{P}$ if and only if $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{Z}$. Indeed, runs in $\mathcal{P}$ have equivalent runs in $\mathcal{Z}$, where each firing of a transition t is replaced with the firing of transitions t_{sim}, then t_{con}, then t_{pro}. Conversely, as previously detailed, runs in $\mathcal{Z}$ must fire t_{sim}, t_{con}, and t_{pro} successively for some transition $t \in T$.

It remains to observe that $\mathcal{Z}$ is acyclic. Consider the following ordering: places in G occur before production transitions (such as t_{pro}), which occur before places in P, which occur before consumption transitions (such as t_{con}). ◀

▶ **Lemma 4.15.** *The reachability problem for acyclic Petri nets with zero tests is reducible in logarithmic space to the reachability problem for acyclic Petri nets with resets.*

Proof of Lemma 4.15. Via leveraging the reachability objective, the idea is to add a copy of each place that will make sure zero tests are simulated faithfully.

Let $\mathcal{Z} = (P, T, F, Z)$ be an acyclic Petri net with zero tests. We will construct an acyclic Petri net with resets $\mathcal{R} = (P', T', F', R)$. For each place $p \in P$ we will add a copy place c_p, so $P' = \{p, c_p : p \in P\}$. For each transition $t \in T$, there will be a corresponding transition $t' \in T'$ with the following behaviour. Firstly, t' will mimic the token consumption and token production between the original places and their copies, so for every place p, ${}^\bullet t'[p] = {}^\bullet t'[c_p] = {}^\bullet t[p]$ and $t'^\bullet[p] = t'^\bullet[c_p] = t^\bullet[p]$. Secondly, suppose a place $p \in P$ is zero-tested by $t \in T$, i.e. $p \in Z(t)$. Then $t' \in T'$ will reset $p \in P'$ but not the copy $c_p \in P'$. Note that none of the copy places are ever reset.

Given the initial marking $\mathbf{m}$ and target marking $\mathbf{n}$ over P, we define $\mathbf{m}'$ and $\mathbf{n}'$ over P' so that $\mathbf{m}[p] = \mathbf{m}'[p] = \mathbf{m}'[c_p]$ and $\mathbf{n}[p] = \mathbf{n}'[p] = \mathbf{n}'[c_p]$. In other words, the markings over P' allocate the same number of tokens to the copy places as their original counterparts. Suppose $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{R}$. Then the invariant $\mathbf{m}'[c_p] - \mathbf{m}'[p] \leq \mathbf{n}'[c_p] - \mathbf{n}'[p]$ holds for all $p \in P$. This inequality is strict only if at some point during the run, a transition is fired that resets a non-empty place. Therefore, $\mathbf{m} \xrightarrow{*} \mathbf{n}$ in $\mathcal{Z}$ if and only if $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{R}$. Indeed, a zero test on p succeeds in $\mathcal{Z}$ if and only if its corresponding reset has no effect, this occurs when p and c_p are empty. To conclude, it is clear that this is a logarithmic-space reduction and that the acyclicity of the consumption and production arcs between places and transitions is preserved in $\mathcal{R}$. ◀

▶ Remark. Lemma 4.14 does not hold with the workflow properties but Lemma 4.15 does; neither holds for the coverability objective.

Proof of Theorem 4.13. Combine Lemma 4.14, Lemma 4.15, and the fact that reachability in Petri nets with zero tests is undecidable. ◀

References

1 Toshiro Araki and Tadao Kasami. Some decision problems related to the reachability problem for Petri nets. *Theor. Comput. Sci.*, 3(1):85–104, 1976. doi:10.1016/0304-3975(76)90067-0.

2 Michael Blondin, Christoph Haase, Filip Mazowiecki, and Mikhail A. Raskin. Affine extensions of integer vector addition systems with states. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:1)2021.

3 Michael Blondin, Filip Mazowiecki, and Philip Offtermatt. The complexity of soundness in workflow nets. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 20:1–20:13. ACM, 2022. doi:10.1145/3531130.3533341.

4 Michael Blondin, Filip Mazowiecki, and Philip Offtermatt. Verifying generalised and structural soundness of workflow nets via relaxations. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification – 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 468–489. Springer, 2022. doi:10.1007/978-3-031-13188-2_23.

5 Michael Blondin and Mikhail A. Raskin. The complexity of reachability in affine vector addition systems with states. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:3)2021.

6 Dmitry Chistikov, Christoph Haase, and Simon Halfon. Context-free commutative grammars with integer counters and resets. *Theor. Comput. Sci.*, 735:147–161, 2018. Special issue for RP 2014. doi:10.1016/j.tcs.2016.06.017.

7 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021. doi:10.1145/3422822.

8 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

9 Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 1998. doi:10.1007/BFb0055044.

10 Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings*, volume 5701 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2009. doi:10.1007/978-3-642-03848-8_19.

11 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 269–278. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.39.

12 Alain Finkel, Pierre McKenzie, and Claudine Picaronny. A well-structured framework for analysing Petri net extensions. *Inf. Comput.*, 195(1-2):1–29, 2004. doi:10.1016/j.ic.2004.01.005.

13 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR 2009 – Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009. doi:10.1007/978-3-642-04081-8_25.

14 Kunihiko Hiraishi and Atsunobu Ichikawa. A class of Petri nets that a necessary and sufficient condition for reachability is obtainable. *Transactions of the Society of Instrument and Control Engineers*, 24(6):635–640, 1988. doi:10.9746/sicetr1965.24.635.

15 Alexander Kaiser, Daniel Kroening, and Thomas Wahl. A widening approach to multithreaded program verification. *ACM Trans. Program. Lang. Syst.*, 36(4):14:1–14:29, 2014. doi:10.1145/2629608.

16 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in VASS Revisited: Improving Rackoff's Bound to Obtain Conditional Optimality. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 131:1–131:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.131.

17 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

18 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.

19 Duan Li, Xiaoling Sun, Jianjun Gao, Shenshen Gu, and Xiaojin Zheng. Reachability determination in acyclic Petri nets by cell enumeration approach. *Autom.*, 47(9):2094–2098, 2011. doi:10.1016/j.automatica.2011.06.017.

20 R. J. Lipton. *The reachability problem requires exponential space.* Research Report. Department of Computer Science, Yale University, 1976. URL: http://www.cs.yale.edu/publications/techreports/tr63.pdf.

21 Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

22 C. A. Petri. Introduction to general net theory. In Wilfried Brauer, editor, *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, Germany, October 8-19, 1979*, volume 84 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 1979. doi:10.1007/3-540-10001-6_21.

23 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

24 Philippe Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer, 2010. doi:10.1007/978-3-642-15155-2_54.

25 Ferucio Laurentiu Tiplea, Corina Bocaneala, and Raluca Chirosca. On the complexity of deciding soundness of acyclic workflow nets. *IEEE Trans. Syst. Man Cybern. Syst.*, 45(9):1292–1298, 2015. doi:10.1109/TSMC.2015.2394735.

26 Ferucio Laurentiu Tiplea and Dan C. Marinescu. Structural soundness of workflow nets is decidable. *Inf. Process. Lett.*, 96(2):54–58, 2005. doi:10.1016/j.ipl.2005.06.002.

27 Rüdiger Valk. Self-modifying nets, a natural extension of Petri nets. In Giorgio Ausiello and Corrado Böhm, editors, *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July 17-21, 1978, Proceedings*, volume 62 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 1978. doi:10.1007/3-540-08860-1_35.

28 Wil M. P. van der Aalst. The application of Petri nets to workflow management. *J. Circuits Syst. Comput.*, 8(1):21–66, 1998. doi:10.1142/S0218126698000043.

29 Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets with reset arcs. *Trans. Petri Nets Other Model. Concurr.*, 3:50–70, 2009. doi:10.1007/978-3-642-04856-2_3.

30 Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects Comput.*, 23(3):333–363, 2011. doi:10.1007/s00165-010-0161-4.