The Downward-Closure of Petri Net Languages*

Peter Habermehl[1], Roland Meyer[1], and Harro Wimmel[2]

[1] LIAFA, Paris Diderot University & CNRS
e-mail: {peter.habermehl,roland.meyer}@liafa.jussieu.fr

[2] Department of Computing Science, University of Rostock
e-mail: harro.wimmel@uni-rostock.de

Abstract. We show that the downward-closure of a Petri net language is effectively computable. This is mainly done by using the notions defined for showing decidability of the reachability problem of Petri nets. In particular, we rely on Lambert's construction of marked graph transition sequences — special instances of coverability graphs that allow us to extract constructively the simple regular expression corresponding to the downward-closure. We also consider the remaining language types for Petri nets common in the literature. For all of them, we provide algorithms that compute the simple regular expressions of their downward-closure. As application, we outline an algorithm to automatically analyse the stability of a system against attacks from a malicious environment.

1 Introduction

Petri nets or the very similar vector addition systems are a popular fundamental model for concurrent systems. Deep results have been obtained in Petri net theory, among them and perhaps most important decidability of the reachability problem [6, 10, 8], whose precise complexity is still open.

Petri nets have also been studied in formal language theory, and several notions of Petri net languages have been introduced. The standard notion to which we simply refer as *Petri net language* accepts sequences of transition labels in a run from an initial to a final marking. Other notions are the *prefix language* considering all markings to be final, the *covering language* where sequences leading to markings that dominate a given final marking are accepted, and *terminal languages* where all sequences leading to a deadlock are computed.

We study the *downward-closure* of all these languages wrt. the subword ordering [4]. It is well known that given a language L over some finite alphabet its downward-closure is regular; it can always be written as the complement of an upward-closed set, which in turn is characterised by a finite set of minimal elements. Even more, downward-closed languages correspond to *simple regular expressions* [1]. However, such an expression is not always effectively computable. This depends on L. For example, the reachability set of lossy channel systems is downward-closed but not effectively computable [11], even though membership

* The first authors were supported by the French ANR projects Averiss and Veridyc. The article appeared in Proc. of ICALP. LNCS. Springer-Verlag, 2010.

in the set is decidable. On the contrary, for pushdown-automata the problem has been solved positively by Courcelle [2].

We show as our main result that the downward-closure of Petri net languages is *effectively computable*. This is done by a careful inspection of the *proof of decidability of the reachability problem* due to Lambert [8]. From his so-called perfect marked graph transition sequences (MGTS) we directly extract the simple regular expression corresponding to the downward-closure of the language. Key to this is an iteration argument that employs Lambert's pumping lemma for Petri nets and the particular structure of MGTS in a non-trivial way.

We also establish computability of the downward-closure for the remaining language types. For terminal languages we rely on the previous result, whereas for covering and prefix languages we directly construct the expressions from the coverability tree of the Petri net.

To be able to compute the downward-closure of a language is important for several reasons. For example, it is precisely what an environment observes from a language in an asynchronous interaction. A component which periodically observes the actions (or alternatively states) of another process will see exactly the downward-closure of the language of actions the partner issues. Another application of the downward-closure of a language is the use as a regular overapproximation of the system behaviour, allowing for safe inclusion checks between a Petri net language and all types of languages for which inclusion of a regular language (or even only simple regular expressions) is decidable.

We apply our results to automatically analyse the stability of a system against attacks. Consider a malicious environment that tries to force the system into an undesirable state. Then the downward-closure of the environment's language provides information about the intrusions the system can tolerate.

The paper is organised as follows. In Section 2, we provide preliminary definitions concerning Petri nets, languages, and downward-closed sets. In Section 3, we state our main result. The downward-closure of Petri net languages is effectively computable. In Section 4, we investigate the other language types. In Section 5 we illustrate an application of our result before concluding in Section 6.

2 Petri nets and their languages

Petri nets generalise finite automata by distributed states and explicite synchronisation of transitions. A *Petri net* is a triple (P, T, F) with finite and disjoint sets of *places* P and *transitions* T. The *flow function* $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ determines the mutual influence of places and transitions.

States of Petri nets, typically called *markings*, are functions $M \in \mathbb{N}^P$ that assign a natural number to each place. We say that a place p *has* k *tokens under* M if $M(p) = k$. A marking M *enables* a transition t, denoted by $M[t\rangle$, if the places carry at least the number of tokens required by F, i.e., $M(p) \geq F(p,t)$ for all $p \in P$. A transition t that is enabled in M may be *fired* and yields marking M' with $M'(p) = M(p) - F(p,t) + F(t,p)$ for all $p \in P$. The firing relation is extended inductively to transition sequences $\sigma \in T^*$.

Let symbol ω represent an unbounded number and abbreviate $\mathbb{N} \cup \{\omega\}$ by $\mathbb{N}_\omega$. The usual order $\leq$ on natural numbers extends to $\mathbb{N}_\omega$ by defining $n \leq \omega$ for all $n \in \mathbb{N}$. Similar to markings, ω-markings are functions in $\mathbb{N}_\omega^P$. The ordering $\preceq_\omega \subseteq \mathbb{N}_\omega^P \times \mathbb{N}_\omega^P$ defines the precision of ω-markings. We have $M \preceq_\omega M'$ if $M(p) = M'(p)$ or $M'(p) = \omega$.

To adapt the firing rule to ω-markings, we define $\omega - n := \omega =: \omega + n$ for any $n \in \mathbb{N}$. The relation defined above can now be applied to ω-markings, and firing a transition will never increase or decrease the number of tokens for a place p with $M(p) = \omega$. An ω-marking M' is *reachable* from an ω-marking M in a net N if there is a firing sequence leading from M to M'. We denote the set of ω-markings reachable from M by $\mathcal{R}(M)$.

Definition 1. *The* reachability problem **RP** *is the set*

$$\mathbf{RP} := \{(N, M, M') \mid N = (P, T, F), M, M' \in \mathbb{N}_\omega^P, \textit{ and } M' \in \mathcal{R}(M)\}.$$

The reachability problem **RP** is known to be decidable. This was first proved by Mayr [9, 10] with an alternative proof by Kosaraju [6]. In the '90s, Lambert [8] presented another proof, which can also be found in [13].

To deal with reachability, reachability graphs and coverability graphs were introduced in [5]. Consider $N = (P, T, F)$ with an ω-marking $M_0 \in \mathbb{N}_\omega^P$. The *reachability graph* R of (N, M_0) is the edge-labelled graph $R = (\mathcal{R}(M_0), E, T)$, where a t-labelled edge $e = (M_1, t, M_2)$ is in E whenever $M_1[t\rangle M_2$.

A *coverability graph* $C = (V, E, T)$ of (N, M_0) is defined inductively. First, M_0 is in V. Then, if $M_1 \in V$ and $M_1[t\rangle M_2$, check for every M on a path from M_0 to M_1 if $M \leq M_2$. If the latter holds, change $M_2(s)$ to ω whenever $M_2(s) > M(s)$. Add, if not yet contained, the modified M_2 to V and (M_1, t, M_2) to E. The procedure is repeated, until no more nodes and edges can be added.

Reachability graphs are usually infinite, whereas coverability graphs are always finite. But due to the inexact ω-markings, coverability graphs do not allow for deciding reachability. However, the concept is still valuable in dealing with reachability, as it allows for a partial solution to the problem. A marking M is not reachable if there is no M' with $M' \geq M$ in the coverability graph. For a complete solution of the reachability problem, coverability graphs need to be extended as discussed in Section 3.

Our main contributions are procedures to compute representations of Petri net languages. Different language types have been proposed in the literature that we shall briefly recall in the following definition [12].

Definition 2. *Consider a Petri net $N = (P, T, F)$ with initial and final markings $M_0, M_f \in \mathbb{N}^P$, Σ a finite alphabet, and $h \in (\Sigma \cup \{\epsilon\})^T$ a labelling that is extended homomorphically to T^*. The* language *of N accepts firing sequences to the final marking:*

$$\mathcal{L}_h(N, M_0, M_f) := \{h(\sigma) \mid M_0[\sigma\rangle M_f \textit{ for some } \sigma \in T^*\}.$$

We write $\mathcal{L}(N, M_0, M_f)$ if h is the identity. The prefix language *of the net accepts all transition sequences:*

$$\mathcal{P}_h(N, M_0) := \{h(\sigma) \mid M_0[\sigma\rangle M \textit{ for some } \sigma \in T^* \textit{ and } M \in \mathbb{N}^P\}.$$

The terminal language *of the Petri net accepts runs to deadlock markings, i.e., markings where no transition is enabled:*

$$\mathcal{T}_h(N, M_0) := \{h(\sigma) \mid M_0[\sigma\rangle M \textit{ with } \sigma \in T^*, M \in \mathbb{N}^P, \textit{ and } M \textit{ is a deadlock}\}.$$

The covering language *requires domination of the final marking:*

$$\mathcal{C}_h(N, M_0, M_f) := \{h(\sigma) \mid M_0[\sigma\rangle M \geq M_f \textit{ for some } \sigma \in T^* \textit{ and } M \in \mathbb{N}^P\}.$$

Note that the prefix language $\mathcal{P}_h(N, M_0)$ is the covering language of the marking that assigns zero to all places, $\mathcal{P}_h(N, M_0) = \mathcal{C}_h(N, M_0, \mathbf{0})$.

We are interested in the downward-closure of the above languages wrt. the subword ordering $\preceq \subseteq \Sigma^* \times \Sigma^*$. The relation $a_1 \ldots a_m \preceq b_1 \ldots b_n$ requires word $a_1 \ldots a_m$ to be embedded in $b_1 \ldots b_n$, i.e., there are indices $i_1, \ldots, i_m \in \{1, \ldots, n\}$ with $i_1 < \ldots < i_m$ so that $a_j = b_{i_j}$ for all $j \in \{1, \ldots, m\}$. Given a language L, its downward-closure is $L\downarrow := \{w \mid w \preceq v \text{ for some } v \in L\}$. A downward-closed language is a language L such that $L\downarrow = L$. Every downward-closed language is regular since it is the complement of an upward-closed set which can be represented by a finite number of minimal elements with respect to $\preceq$. This follows from the fact that the subword relation is a well-quasi-ordering on words [4]. More precisely, every downward-closed set can be written as a *simple regular expression* over Σ (see [1]): We call an atomic expression any regular expression e of the form $(a + \epsilon)$ where $a \in \Sigma$, or of the form $(a_1 + \cdots + a_m)^*$ where $a_1, \ldots, a_m \in \Sigma$. A product p is either the empty word ϵ or a finite sequence $e_1 e_2 \ldots e_n$ of atomic expressions. A simple regular expression is then either $\emptyset$ or a finite union $p_1 + \cdots + p_k$ of products.

3 Downward-closure of Petri net languages

Fix a Petri net $N = (P, T, F)$ with initial and final markings $M_0, M_f \in \mathbb{N}^P$ and labelling $h \in (\Sigma \cup \{\epsilon\})^T$. We establish the following main result.

Theorem 1. *$\mathcal{L}_h(N, M_0, M_f)\downarrow$ is computable as (♣) below.*

Recall that any downward-closed language is representable by a simple regular expression [1]. We show that in case of Petri net languages these expressions can be computed effectively. In fact, they turn out to be rather natural; they correspond to the transition sets in the precovering graphs of the net. To see this, we shall need some insight into the decidability proof for reachability in Petri nets. We follow here essentially the presentation given in [14] for solving the infinity problem of intermediate states in Petri nets.

3.1 A look at the decidability of RP

We present here some main ideas behind the proof of decidability of **RP** according to Lambert [8, 13]. The proof is based on marked graph transition sequences

(MGTS), which are sequences of special instances of coverability graphs C_i alternating with transitions t_j of the form $G = C_1.t_1.C_2 \ldots t_{n-1}.C_n$. These special instances of coverability graphs are called *precovering graphs* in [8] and have additional structure from which we shall only use strong connectedness. Each precovering graph C_i is augmented by two additional ω-markings, the *input* $m_{i,in}$ and the *output* $m_{i,out}$. The *initial marking* M_i of C_i is less concrete than input and output, $m_{i,in} \preceq_\omega M_i$ and $m_{i,out} \preceq_\omega M_i$. The transitions $t_1, \ldots, t_{n-1}$ in an MGTS connect the output $m_{i,out}$ of one precovering graph to the input $m_{i+1,in}$ of the next, see Figure 1.

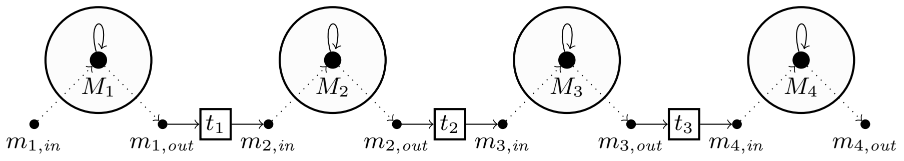

Fig. 1. A marked graph transition sequence $C_1.t_1.C_2 \ldots t_3.C_4$. Dots represent markings and circles represent strongly connected precovering graphs with in general more than one node. The initial marking is depicted in the center. Solid lines inside these circles are transition sequences that must be firable in the Petri net. Dotted lines show the entry to and exit from precovering graphs, which do not change the actual marking in the Petri net. Both $m_{i,in} \preceq_\omega M_i$ and $m_{i,out} \preceq_\omega M_i$ hold for every i.

A *solution* of an MGTS is by definition a transition sequence leading through the MGTS. In Figure 1 it begins with marking $m_{1,in}$, leads in cycles through the first precovering graph until marking $m_{1,out}$ is reached, then t_1 can fire to reach $m_{2,in}$, from which the second coverability graph is entered and so on, until the MGTS ends. Whenever the marking of some node has a finite value for some place, this value *must be reached exactly* by the transition sequence. If the value is ω, there are no such conditions. The *set of solutions* of an MGTS G is denoted by $\mathcal{L}(G)$ [8, page 90].

An instance $RP = (N, M_0, M_f)$ of the reachability problem can be formulated as the problem of finding a solution for the special MGTS G_{RP} depicted in Figure 2. The node $\boldsymbol{\omega}$ (with all entries ω) is the only node of the coverability graph, i.e., we allow for arbitrary ω-markings and firings of transitions between $m_{1,in} = M_0$ and $m_{1,out} = M_f$, but the sequence must begin exactly at the (concrete) initial marking of the net and end at its final marking. The solutions to this MGTS are precisely the runs in the net N leading from M_0 to M_f:

$$\mathcal{L}(N, M_0, M_f) = \mathcal{L}(G_{RP}).$$

Hence, to decide **RP** it is sufficient to solve arbitrary MGTS. Lambert defines for each MGTS a characteristic equation that is fulfilled by all its solutions. In other words, the equation is a necessary condition for solutions of the MGTS. More

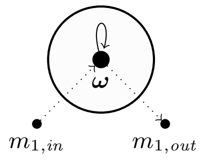

Fig. 2. MGTS representation of an instance $(N, m_{1,in}, m_{1,out})$ of the reachability problem. The MGTS consists of one precovering graph with a single node $\boldsymbol{\omega}$ which represents the ω-marking where all places have an unbounded number of tokens and from which every transition can fire. A solution for this MGTS is a transition sequence from $m_{1,in}$ to $m_{1,out}$.

precisely, the author derives a system of linear equations $Ax = b$ where A and b range over integers. It encodes the firing behaviour of the precovering graphs and intermediary transitions and can become quite large. There is one variable for every marking entry $m_{i,in}$ and $m_{i,out}$ (including zero and ω entries) as well as one variable for each edge in every precovering graph. Since markings must not become negative, solutions sought must be semi-positive. This (possibly empty) set of semi-positive solutions can always be computed [7].

If the characteristic equation was sufficient for the existence of solutions of an MGTS, **RP** would have been solved immediately. While not valid in general, Lambert provides precise conditions for when this implication holds. Generally speaking, a solution to the characteristic equation yields a solution to the MGTS if the variables for the edges and the variables for all ω-entries of the markings are unbounded in the solution space. An MGTS with such a sufficient characteristic equation is called *perfect* and denoted by $\mathbb{G}$. Unboundedness of the variables can be checked effectively [7].

Since not all MGTS are perfect, Lambert presents a decomposition procedure [8]. It computes from one MGTS G a new set of MGTS that are to a greater degree perfect and have the same solutions as G. This means each transition sequence leading through the original MGTS and solving it will also lead through at least one of the derived MGTS and solve it, and vice versa. The degree of perfectness is discrete and cannot be increased indefinitely. Therefore the decomposition procedure terminates and returns a finite set Γ_G of perfect MGTS. With the assumption that $m_{1,in}$ and $m_{n,out}$ are ω-free, the corresponding decomposition theorem is simplified to the following form.

Theorem 2 (Decomposition [8, 13]). *An MGTS G can be decomposed into a finite set Γ_G of perfect MGTS with the same solutions, $\mathcal{L}(G) = \bigcup_{\mathbb{G}\in\Gamma_G} \mathcal{L}(\mathbb{G})$.*

When we apply the decomposition procedure to the MGTS G_{RP} for the instance $RP = (N, m_{1,in}, m_{1,out})$ of the reachability problem (Figure 2), we obtain a set $\Gamma_{G_{RP}}$ of perfect MGTS. For each of these perfect MGTS $\mathbb{G}$ we can decide whether it has solutions, i.e., whether $\mathcal{L}(\mathbb{G}) \neq \emptyset$. If at least one has a solution, we obtain a positive answer to the reachability problem, otherwise a negative answer. This means, the following algorithm decides **RP**:

```
**input** $RP = (N, m_{1,in}, m_{1,out})$
create $G_{RP}$ according to Figure 2
decompose $G_{RP}$ into $\Gamma_{G_{RP}}$ with $\mathbb{G}$ perfect for all $\mathbb{G} \in \Gamma_{G_{RP}}$
if $\exists \mathbb{G} \in \Gamma_{G_{RP}}$ with $\mathcal{L}(\mathbb{G}) \neq \emptyset$ answer yes else answer no.
```

The reachability problem is not only decidable. If it has a solution, it is also possible to calculate a solving transition sequence. Consider a perfect MGTS $\mathbb{G} = C_1.t_1.C_2 \ldots t_{n-1}.C_n$ and let each precovering graph have the initial marking M_i (cf. Figure 1). We search for covering sequences u_i that indefinitely increase the token count on ω-places of M_i. More precisely, u_i is a transition sequence from M_i to M_i with the following properties.

- The sequence u_i is enabled under marking $m_{i,in}$.
- If $M_i(s) = \omega > m_{i,in}(s)$, then u_i will add tokens to place s.
- If $M_i(s) = m_{i,in}(s) \in \mathbb{N}$, then u_i will not change the token count on place s.

In case $M_i(s) = \omega = m_{i,in}(s)$, no requirements are imposed. Sequence u_i is accompanied by a second transition sequence v_i with similar properties, except that the reverse of v_i must be able to fire backwards from $m_{i,out}$. This decreases the token count on ω-places and lets v_i reach the output node $m_{i,out}$ from M_i. Having such pairs of covering sequences $((u_i, v_i))_{1 \leq i \leq n}$ available for all precovering graphs, the following theorem yields a solution to the perfect MGTS.

Theorem 3 (Lambert's Iteration Lemma [8, 13]). *Consider some perfect MGTS $\mathbb{G}$ with at least one solution and let $((u_i, v_i))_{1 \leq i \leq n}$ be covering sequences satisfying the above requirements. We can compute $k_0 \in \mathbb{N}$ and transition sequences β_i, w_i from M_i to M_i such that* for every $k \geq k_0$ *the sequence*

$$(u_1)^k \beta_1 (w_1)^k (v_1)^k t_1 (u_2)^k \beta_2 (w_2)^k (v_2)^k t_2 \ldots t_{n-1} (u_n)^k \beta_n (w_n)^k (v_n)^k$$

is a solution of $\mathbb{G}$.

Lambert proved that such covering sequences u_i, v_i always exist and that at least one can be computed [8]. Firing u_i repeatedly, at least k_0 times, pumps up the marking to the level necessary to execute $\beta_i(w_i)^k$. Afterwards v_i pumps it down to reach $m_{i,out}$. Transition t_i then proceeds to the next precovering graph.

3.2 Computing the downward-closure

According to the decomposition theorem, we can represent the Petri net language $\mathcal{L}(N, M_0, M_f)$ by the decomposition of the corresponding MGTS G_{RP}. We shall restrict our attention to perfect MGTS $\mathbb{G}$ that have a solution, i.e., $\mathcal{L}(\mathbb{G}) \neq \emptyset$. They form the subset $\Gamma^{\checkmark}_{G_{RP}}$ of $\Gamma_{G_{RP}}$. As the labelled language just applies a homomorphism, we derive

$$\mathcal{L}_h(N, M_0, M_f) = h(\mathcal{L}(N, M_0, M_f)) = h(\bigcup_{\mathbb{G} \in \Gamma^{\checkmark}_{G_{RP}}} \mathcal{L}(\mathbb{G})) = \bigcup_{\mathbb{G} \in \Gamma^{\checkmark}_{G_{RP}}} h(\mathcal{L}(\mathbb{G})).$$

Since downward-closure $-\downarrow$ and the application of h commute, and since downward-closure distributes over $\cup$, we obtain

$$\mathcal{L}_h(N, M_0, M_f)\downarrow = \bigcup_{\mathbb{G}\in\Gamma^{\checkmark}_{G_{RP}}} h(\mathcal{L}(\mathbb{G})\downarrow).$$

The main result in this section is a characterisation of $\mathcal{L}(\mathbb{G})\downarrow$ as a simple regular expression $\phi_{\mathbb{G}}$. By the previous argumentation this solves the representation problem of $\mathcal{L}_h(N, M_0, M_f)\downarrow$ and hence proves Theorem 1. We compute $\phi_{\mathbb{G}}$ for the language of every perfect MGTS $\mathbb{G} \in \Gamma^{\checkmark}_{G_{RP}}$. Then we apply the homomorphism to these expressions, $h(\phi_{\mathbb{G}})$, and end up in a finite disjunction

$$\mathcal{L}_h(N, M_0, M_f)\downarrow = \mathcal{L}(\sum_{\mathbb{G}\in\Gamma^{\checkmark}_{G_{RP}}} h(\phi_{\mathbb{G}})). \tag{$\clubsuit$}$$

We spend the remainder of the section on the representation of $\mathcal{L}(\mathbb{G})\downarrow$. Surprisingly, the simple regular expression turns out to be just the sequence of transition sets in the precovering graph,

$$\phi_{\mathbb{G}} := T_1^*.(t_1 + \epsilon).T_2^* \ldots (t_{n-1} + \epsilon).T_n^*,$$

where $\mathbb{G} = C_1.t_1.C_2 \ldots t_{n-1}.C_n$ and C_i contains the transitions T_i.

Proposition 1. *$\mathcal{L}(\mathbb{G})\downarrow = \mathcal{L}(\phi_{\mathbb{G}})$.*

The inclusion from left to right is trivial. The proof of the reverse direction relies on the following key observation about Lambert's iteration lemma. The sequences u_i can always be chosen in such a way that they contain all transitions of the precovering graph C_i. By iteration we obtain all sequences u_i^k. Since u_i contains all transitions in T_i, we derive

$$T_i^* \subseteq (\bigcup_{k\in\mathbb{N}} u_i^k)\downarrow = \bigcup_{k\in\mathbb{N}} u_i^k\downarrow .$$

Hence, all that remains to be shown is that u_i can be constructed so as to contain all edges of C_i and consequently all transitions in T_i. Lets start with a covering sequence u_i' that satisfies the requirements stated above and that can be found with Lambert's procedure [8]. Since C_i is strongly connected, there is a finite path z_i from M_i to M_i that contains all edges of C_i. The corresponding transition sequence may have a negative effect on the ω-places, say at most $m \in \mathbb{N}$ tokens are removed. Concrete token counts are, by construction of precovering graphs, reproduced exactly. Since u_i' is a covering sequence, we can repeat it $m+1$ times. By the second requirement, this adds at least $m+1$ tokens to every ω-place. If we now append z_i, we may decrease the token count by m but still guarantee a positive effect of $m + 1 - m = 1$ on the ω-places. This means

$$u_i := {u_i'}^{m+1}.z_i$$

is a covering sequence that we may use instead of u_i' and that contains all transitions. This concludes the proof of Proposition 1.

4 Downward-closure of other language types

We consider the downward-closure of terminal and covering languages. For terminal languages that accept via deadlocks we provide a reduction to the previous computability result. For covering languages, we avoid solving reachability and give a direct construction of the downward-closure from the coverability tree.

4.1 Terminal languages

Deadlocks in a Petri net $N = (P', T, F)$ are characterised by a finite set $\mathcal{P}$ of partially specified markings where the token count on some places is arbitrary, $M_P \in \mathbb{N}^P$ with $P \subseteq P'$. Each such partial marking corresponds to a case where no transition can fire. Hence, the terminal language is a finite union of partial languages that accept by a partial marking, $\mathcal{T}_h(N, M_0) = \bigcup_{M_P \in \mathcal{P}} \mathcal{L}_h(N, M_0, M_P)$. We now formalise the notion of a partial language and then prove computability of its downward-closure. With the previous argumentation, this yields a representation for the downward-closure of the terminal language.

A partial marking $M_P \in \mathbb{N}^P$ with $P \subseteq P'$ denotes a potentially infinite set of markings M that coincide with M_P in the places in P, $M|_P = M_P$. The *partial language* is therefore defined to be $\mathcal{L}_h(N, M_0, M_P) := \bigcup_{M|_P = M_P} \mathcal{L}_h(N, M_0, M)$. We apply a construction due to Hack [3] to compute this union.

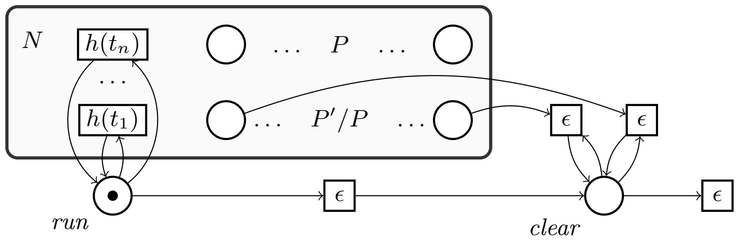

Fig. 3. Hack's construction to reduce partial Petri net languages to ordinary languages.

We extend the given net $N = (P', T, F)$ to $N_e = (P' \cup P_e, T \cup T_e, F_e)$ as illustrated in Figure 3. The idea is to guess a final state by removing the *run* token and then empty the places outside $P \subseteq P'$. As a result, the runs in N from M_0 to a marking M with $M|_P = M_P$ are precisely the runs in N_e from M_0^r to the marking M_f, up to the token removal phase in the end. Marking M_0^r is M_0 with an additional token on the *run* place. Marking M_f coincides with M_P and has no tokens on the remaining places. Projecting away the new transitions t with $h_e(t) = \epsilon$ characterises the partial language by an ordinary language.

Lemma 1. $\mathcal{L}_h(N, M_0, M_P) = \mathcal{L}_{h \cup h_e}(N_e, M_0^r, M_f)$.

Combined with Theorem 1, a simple regular expression ϕ_{M_p} is computable that satisfies $\mathcal{L}_h(N, M_0, M_P)\downarrow = \mathcal{L}(\phi_{M_p})$. As a consequence, the downward-closure of the terminal language is the (finite) disjunction of these expressions.

Theorem 4. $\mathcal{T}_h(N, M_0)\downarrow = \mathcal{L}(\Sigma_{M_P \in \mathcal{P}}\ \phi_{M_p})$.

Note that the trick we employ for the partially specified final markings also works for partial input markings. Hence, we can compute the language and the terminal language also for nets with partially specified input markings.

4.2 Covering languages

We extend the coverability tree to a finite automaton where the downward-closure coincides with the downward-closure of the covering-language. Hence, the desired regular expression is computable. The idea is to add edges to the coverability tree that represent the domination of markings by their successors and thus, by monotonicity of Petri nets, indicate cyclic behaviour. The final states reflect domination of the final marking. In the remainder, fix $N = (P, T, F)$ with initial and final markings M_0 and M_f and labelling $h \in (\Sigma \cup \{\epsilon\})^T$.

The coverability tree $CT = (V, E, \lambda)$ is similar to the coverability graph discussed in Section 2 but keeps the tree structure of the computation. Therefore, the vertices are labelled by extended markings, $\lambda(v) \in (\mathbb{N} \cup \{\omega\})^P$, and the edges $e \in E \subseteq V \times V$ by transitions, $\lambda(e) \in T$. A path is truncated as soon as it repeats an already visited marking.

We extend CT to a finite automaton $FA = (V, v_0, V_f, E \cup E', \lambda \cup \lambda')$ by adding backedges. The root of CT is the initial state v_0. States that cover M_f are final, $V_f := \{v \in V \mid \lambda(v) = M \geq M_f\}$. If the marking of v dominates the marking of an E-predecessor v', $\lambda(v) = M \geq M' = \lambda(v')$, we add a backedge $e' = (v, v')$ to E' and label it by $\lambda'(e') = \epsilon$. The downward-closed language of this automaton is the downward-closed covering language without labelling.

Lemma 2. $\mathcal{L}(FA)\downarrow = \mathcal{C}(N, M_0, M_f)\downarrow$.

To compute $\mathcal{L}(FA)\downarrow$ we represent the automaton as tree of its strongly connected components $SCC(FA)$. The root is the component C_0 that contains v_0. We need two additional functions to compute the regular expression. For two components $C, C' \in SCC(FA)$, let $\gamma_{C,C'} = (t + \varepsilon)$ if there is a t-labelled edge from C to C', and let $\gamma_{C,C'} = \emptyset$ otherwise. Let $\tau_C = \varepsilon$ if C contains a final state and $\tau_C = \emptyset$ otherwise. Concatenation with $\gamma_{C,C'} = \emptyset$ or $\tau_C = \emptyset$ suppresses further regular expressions if there is no edge or final state, respectively. Let T_C denote the transitions occurring in component C as edge labels. We recursively define regular expressions ϕ_C for the downward-closed languages of components:

$$\phi_C := T_C^*.\big(\tau_C + \sum_{C' \in SCC(FA)} \gamma_{C,C'}.\phi_{C'}\big).$$

Due to the tree structure, all regular expressions are well-defined. The following lemma is easy to prove.

Lemma 3. $\mathcal{L}(FA)\downarrow = \mathcal{L}(\phi_{C_0})$.

As the application of h commutes with the downward-closure, a combination of Lemma 2 and 3 yields the desired representation.

Theorem 5. $\mathcal{C}_h(N, M_0, M_f)\downarrow = \mathcal{L}(h(\phi_{C_0}))$.

Note that $h(\phi_{C_0})$ can be transformed into a simple regular expression by distributivity of concatenation over $+$ and removing possible occurrences of $\emptyset$.

5 Applications to stability analysis

Consider a system modelled as a Petri net N_s. It typically interacts with some potentially malicious environment. This means, $N_s = (P_s, T_s, F_s)$ is embedded[1] in a larger net $N = (P, T, F)$ where the environment changes the token count or restricts the firing behaviour in the subnet N_s. Figure 3 illustrates the situation. The environment is Hack's gadget that may stop the Petri net and empty some places. The results obtained in this paper allow us to approximate the attacks system N_s can tolerate without reaching undesirable states.

Consider an initial marking M_0^s of N_s and a bad marking M_b^s that should be avoided. For the full system N we either use $M_0^s, M_b^s \in \mathbb{N}^{P_s}$ as partially specified markings or assume full initial and final markings, $M_0, M_b \in \mathbb{N}^P$ with $M_0|_{P_s} = M_0^s$ and $M_b|_{P_s} = M_b^s$. The stability of N_s is estimated as follows.

Proposition 2. *An upward-closed language is computable that underapproximates the environmental behaviour N_s tolerates without reaching M_b^s from M_0^s.*

We consider the case of full markings M_0 and M_b of N. For partially specified markings, Hack's construction in Section 4.1 reduces the problem to this one. Let the full system N be labelled by h. Relabelling all transitions of N_s to ϵ yields a new homomorphism h' where only environmental transitions are visible. By definition, the downward-closure always contains the language itself, $\mathcal{L}_{h'}(N, M_0, M_b)\downarrow \supseteq \mathcal{L}_{h'}(N, M_0, M_b)$. This is, however, equivalent to

$$\overline{\mathcal{L}_{h'}(N, M_0, M_b)\downarrow} \subseteq \overline{\mathcal{L}_{h'}(N, M_0, M_b)}.$$

By Theorem 1, the simple regular expression for $\mathcal{L}_{h'}(N, M_0, M_b)\downarrow$ is computable. As regular languages are closed under complementation, the expression for $\overline{\mathcal{L}_{h'}(N, M_0, M_b)\downarrow}$ is computable as well. The language is upward-closed and underapproximates the attacks the system can tolerate.

Likewise, if we consider instead of M_b a desirable good marking M_g, then language $\mathcal{L}_{h'}(N, M_0, M_g)\downarrow$ overapproximates the environmental influences required to reach it. The complement of the language provides behaviour that definitely leads away from the good marking. Note that for covering instead of reachability similar arguments apply that rely on Theorem 5.

[1] Formally, $N = (P, T, F)$ is *embedded* in $N' = (P', T', F')$ if $P \subseteq P'$, $T \subseteq T'$, and $F'|_{(S\times T)\cup(T\times S)} = F$. If homomorphism h labels N and h' labels N' then $h'|_T = h$.

6 Conclusion

We have shown that the downward-closures of all types of Petri net languages are effectively computable. As an application of the results, we outlined an algorithm to estimate the stability of a system towards attacks from a malicious environment. In the future, we plan to study further applications. Especially in concurrent system analysis, our results should yield fully automated algorithms for the verification of asynchronous compositions of Petri nets with other models like pushdown-automata. A different application domain is compositional verification of Petri nets. For an observer of a system, it is sufficient to check whether it reaches a critical state in the composition with the downward-closure of the system's language. However, cyclic proof rules are challenging.

References

1. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Form. Methods Syst. Des.*, 25(1):39–65, 2004.
2. B. Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, 44:178–186, 1991.
3. M. Hack. Decidability questions for Petri nets. Technical report, Cambridge, MA, USA, 1976.
4. G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.* (3), 2(7):326–336, 1952.
5. R. M. Karp and R. E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.
6. S. R. Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *STOC*, pages 267–281. ACM, 1982.
7. J. L. Lambert. Finding a partial solution to a linear system of equations in positive integers. *Comput. Math. Applic.*, 15(3):209–212, 1988.
8. J. L. Lambert. A structure to decide reachability in Petri nets. *Theor. Comp. Sci.*, 99(1):79–104, 1992.
9. E. W. Mayr. An algorithm for the general Petri net reachability problem. In *STOC*, pages 238–246. ACM, 1981.
10. E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comp.*, 13(3):441–460, 1984.
11. R. Mayr. Undecidable problems in unreliable computations. *Theor. Comp. Sci.*, 297(1-3):337–354, 2003.
12. J. L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.
13. L. Priese and H. Wimmel. *Petri-Netze.* Springer, 2003.
14. H. Wimmel. Infinity of intermediate states is decidable for Petri nets. In *ICATPN*, volume 3099 of *LNCS*, pages 426–434. Springer, 2004.