Efficient Restrictions of Immediate Observation Petri Nets*

Michael Raskin and Chana Weil-Kennedy
{raskin,chana.weilkennedy}@in.tum.de

Technical University of Munich, Germany

Abstract. In a previous paper we introduced immediate observation Petri nets [9], a subclass of Petri nets with application domains in distributed protocols (population protocols) and theoretical chemistry (chemical reaction networks). IO nets enjoy many useful properties [9,12], but like the general case of conservative Petri nets they have a PSPACE-complete reachability problem. In this paper we explore two restrictions of the reachability problem for IO nets which lower the complexity of the problem drastically. The complexity is NP-complete for the first restriction with applications in distributed protocols, and it is polynomial for the second restriction with applications in chemical settings.

Keywords: Petri nets, reachability, computational complexity

1 Introduction

In this paper we refine our results about the complexity of verifying immediate observation Petri nets [9] in the case of two restrictions of such nets. Petri nets and their subclasses are widely used and studied in the context of software and system verification (e.g. [7]), but also others such as game theory (e.g. [11]), chemical reaction networks (e.g. [3]) etc. Unfortunately many important problems there have high complexity, and reachability is at least TOWER-hard in the general case [6]. This motivates the study of subclasses of Petri nets.

Immediate observation Petri nets (IO nets) are a reformulation of immediate observation population protocols, which have been introduced by Angluin et al. in [2]. Initially, they were studied from the point of view of computing predicates in a distributed system, where their expressive power is lower than general population protocols (conservative Petri nets) but still considerable. Many verification problems for IO nets are PSPACE-complete; among them set-parametrized problems for sets defined by boolean combinations of bounds on token counts. This is a significant improvement compared to the general or conservative case of Petri nets, where EXPSPACE-hard [4] and even harder verification problems are the

* This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS)

norm. IO nets provide a natural description of some distributed systems, but also can be used to describe enzymatic chemical networks [1].

Of course, a subclass of reachability problems with a better computational complexity raises some natural, even if informal, questions. What allows better complexity and can it be generalized to some wider subclass? What keeps the complexity from being even lower and are there useful subclasses without these obstacles? Are there applications where a typical problem can be solved more efficiently? We believe that branching immediate observation nets, a generalization of IO nets and basic parallel processes with reachability problem in PSPACE[12], answer the first question. The present paper is devoted the last two questions.

We consider two restrictions, the first one a syntactic restriction defining a subclass of IO nets, and the second a condition on the initial and final markings considered in the reachability problem for IO nets. Such restrictions are plausible in applications to some distributed systems (delayed observation population protocols,[2]) and to some chemical systems (enzymatic chemical reaction networks, [1]). We show the first restriction entails an NP-complete reachability problem, and for the second restriction we provide a polynomial algorithm deciding reachability or giving a witness that the restriction does not hold.

The rest of the paper is organized as follows. In section 2, we recall some general definitions regarding Petri nets, as well as the classic maximum flow minimum cut problem. Section 3 defines immediate observation Petri nets. Then we show the effects for reachability complexity of two restrictions on IO nets: keeping transitions enabled once enabled in Section 4, and requiring all token counts and their combinations to be large or zero in Section 5. Finally, we summarize our results in the conclusion and outline some further directions.

2 Preliminaries

Multisets. A *multiset* on a finite set E is a mapping $C\colon E \to \mathbb{N}$, i.e. for any $e \in E$, $C(e)$ denotes the number of occurrences of element e in C. Let $\wr e_1, \ldots, e_n \int$ denote the multiset C such that $C(e) = |\{j \mid e_j = e\}|$. Operations on $\mathbb{N}$ like addition or comparison are extended to multisets by defining them component wise on each element of E. We call $|C| \overset{\text{def}}{=} \sum_{e \in E} C(e)$ the *size* of C.

Place/transition Petri nets with weighted arcs. A *Petri net* N is a triple (P, T, F) consisting of a finite set of *places* P, a finite set of *transitions* T and a *flow function* $F\colon (P \times T) \cup (T \times P) \to \mathbb{N}$. A *marking* M is a multiset on P, and we say that a marking M puts $M(p)$ *tokens* in place p of P. The *size* of M, denoted by $|M|$, is the total number of tokens in M. The *preset* ${}^\bullet t$ and *postset* $t^\bullet$ of a transition t are the multisets on P given by ${}^\bullet t(p) = F(p, t)$ and $t^\bullet(p) = F(t, p)$. A transition t is *enabled* at a marking M if ${}^\bullet t \leq M$, i.e. ${}^\bullet t$ is component-wise smaller or equal to M. If t is enabled then it can be *fired*, leading to a new marking $M' = M - {}^\bullet t + t^\bullet$. We let $M \xrightarrow{t} M'$ denote this. Given $\sigma = t_1 \ldots t_n$ we write $M \xrightarrow{\sigma} M_n$ when $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots \xrightarrow{t_n} M_n$, and call σ

a *firing sequence*. We write $M' \xrightarrow{*} M''$ if $M' \xrightarrow{\sigma} M''$ for some $\sigma \in T^*$, and say that M'' is *reachable* from M'.

Flows and cuts. A *flow graph* is a triple $G = (V, A, c)$ where V is a finite set of vertices, $A \subseteq V^2$ is a finite set of arcs, and $c : A \to \mathbb{N}$ is a nonnegative *capacity* function on arcs A flow graph contains two special vertices i and o, called the *inlet* and *outlet*, such that the i has no incoming arc and o has no outgoing arc. A *flow* of a flow graph is a function $f : A \to \mathbb{N}$ such that $f(a) \le c(a)$ for each arc $a \in A$, and for each vertex $v \in V \setminus \{i, o\}$, the sum of the flow over v's incoming arcs is equal to the sum of the flow over v's outgoing arcs. The *value* of a flow is the sum $\sum_{(i,p)\in A} f(p)$ of the flow over all arcs from the inlet, or equivalently the sum $\sum_{(p,o)\in A} f(p)$ of the flow over all arcs to the outlet. A *cut* in a flow graph $G = (V, A, c)$ is a pair of disjoint subsets $V_I \sqcup V_O = V$ such that the inlet is in V_I and the outlet is in V_O. The *capacity* of a cut (V_I, V_O) is the sum of the capacities of all the arcs going from vertices in V_I to vertices in V_O.

We recall two classic theorems.

Theorem 1 (Max-flow min-cut theorem [10]). *In a flow graph, the maximum value of a flow is equal to the minimum capacity of a cut.*

Theorem 2 (Dinic algorithm [8]). *Given a flow graph, a flow with the maximum value and a cut with the minimum capacity can be found in polynomial time.*

3 Immediate observation Petri nets

We recall the definition of immediate observation nets (IO nets) from [9].

Definition 1. *A transition t of a Petri net is an* immediate observation transition *(IO transition) if there are places p_s, p_d, p_o, not necessarily distinct, such that ${}^\bullet t = \lbag p_s, p_o \rbag$ and $t^\bullet = \lbag p_d, p_o \rbag$. We call p_s, p_d, p_o the* source*,* destination*, and* observed *places of t, respectively. A Petri net is an* immediate observation net *(IO net) if all its transitions are IO transitions.*

IO nets are *conservative*, i.e. there is no creation or destruction of tokens.s

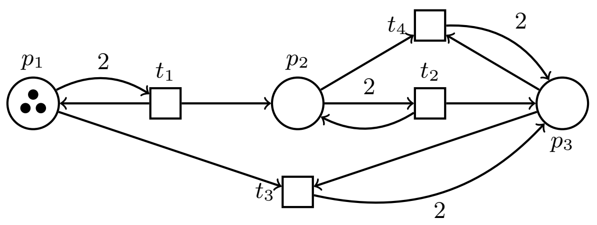

Fig. 1: An IO net.

Example 1. Figure 1 shows an IO net taken from the literature on population protocols [2]. Intuitively, it models a protocol allowing a crowd of undistinguishable agents that can only interact in pairs to decide whether they are at least 3. Initially all agents are in state p_1, modelled by tokens in place p_1. If two agents in state p_1 interact, one of them moves to state p_2 (transition t_1). If two agents in state p_2 interact, one of them moves to p_3 (transition t_2). Finally, an agent in state p_3 can "attract" all other agents to state p_3 (transitions t_3 and t_4). Given a marking M_0 with tokens only in p_1, if $M_0(p_1) \geq 3$ and the pairs of tokens that interact next are chosen uniformly at random, then eventually all tokens reach p_3.

In [9], we showed that given an IO net N and two markings M, M', deciding whether M' is reachable from M is a PSPACE-complete problem. The proof of PSPACE-hardness for the reachability problem in IO nets uses a reduction from the halting problem of linear-space Turing machines. The reduction is done by simulating the runs of the Turing machine: places describe the state of the head and of the tape cells, and transitions model the movement of the head and the change in the symbols on the tape cells. In the construction a specific "success" place becomes marked if and only if the machine reaches the halting state without exceeding the permitted space.

We observe that the nets provided by this reduction have two common properties. First, the transitions get enabled and disabled a very large number of times. Second, the markings put at most one token per place. We show how avoiding at least one of these conditions leads to much easier verification.

4 First restriction: transition enabling

The PSPACE-hardness proof for IO reachability relies on the observation requirements of some transitions switching between satisfied and unsatisfied many times. In some distributed systems observations correspond to irrevocable declarations of the agents, for example in some multi-phase commit protocols. Correspondingly, we consider IO nets where token moves once enabled remain enabled. Sometimes this is not the case for the system on the whole, but it is useful to have indefinite enabling when considering reachability questions. This is the case for example in the delayed observation population protocols introduced by Angluin et al. in [2]. In this model, agents can send an unlimited amount of messages containing their current state. These messages can then be received at any later time by any other agent in the system, who can change their state based on this information. By sending a large amount of messages as soon as a first agent reaches a certain state in a run of the system, certain transitions can become enabled and stay enabled throughout the run. We formalize such a property in the following definition.

Definition 2. *An IO net is* non-forgetting *if for each transitions $p \xrightarrow{r} q$ and $r \xrightarrow{s} r'$ there is also a transition $p \xrightarrow{r'} q$.*

In other words, once it becomes possible to move a token from p to q, it stays possible. The reachability problem for such IO nets becomes much simpler.

Theorem 3. *The reachability problem for non-forgetting IO nets is in* NP*.*

Proof. First, we show that the reachability problem in an IO net with a fixed set of enabled transitions is equivalent to the maximum flow problem on graphs.

Let N be an IO net, let M, M' be two markings of N. We define G as the flow graph with vertices identified with the places P of N, as well as two additional vertices i and o, the inlet and outlet of the flow graph. For each transition $t = p_s \xrightarrow{p_o} p_d \in N$, there is an arc from p_s to p_d in G with infinite capacity. Each vertice p identified with a place of N has one incoming arc from the inlet i with capacity $M(p)$, and one outgoing arc to the outlet o with capacity $M'(p)$. Figure 2 illustrates such a flow graph for a non-forgetting IO net.

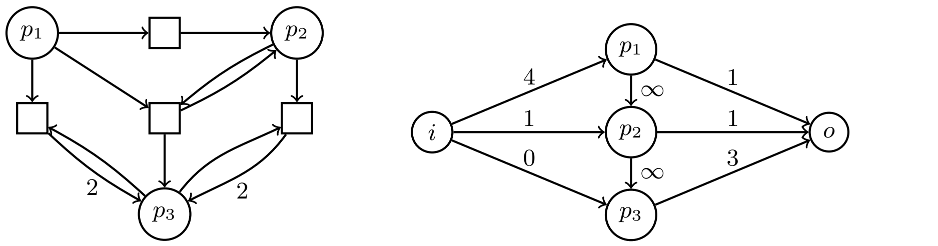

Fig. 2: A non-forgetting IO net and corresponding flow graph for $M = (4, 1, 0)$ and $M' = (1, 1, 3)$.

A firing sequence σ from M to M' in N corresponds naturally to an integer flow f on G, where for all vertices p and q corresponding to places of the IO net, $f(i, p) = M(p), f(p, o) = M'(p)$ and $f(p, q)$ is equal to the number of transitions from p to q in σ. This flow has value $|M| = |M'|$.

Conversely, an integer flow of value $|M| = |M'|$ corresponds to a firing sequence in N, provided N has a fixed set of enabled transitions. Let us consider such a flow f. There exists a multiset θ of transitions of N containing exactly $f(p, q)$ transitions with source place p and destination place q for every pair of places $p, q \in P$. To prove existence of a firing sequence for each such multiset, we consider the following (simple but inefficient) procedure. We repeatedly fire an arbitrary remaining transition such that its source place has more tokens in the current marking than in the final one. As IO nets are conservative, we will be able to pick such a transition unless we have reached the final marking. The transition is enabled because all transitions of N are enabled by assumption.

We proceed to prove that reachability problem for non-forgetting IO nets is in NP. The certificate for reachability corresponding to a firing sequence is defined to consist of the markings where some transition is enabled for the first time. As the number of transitions is less than the length of the input, such a certificate has polynomial length. We use the above reduction to maximum flow to verify the existence of firing sequences between the provided markings where the set of enabled token moves does not change. This can be checked in polynomial time, for example using the Dinic algorithm.

In fact the reachability problem is NP-complete.

Theorem 4. *Reachability problem for non-forgetting IO nets is* NP*-hard.*

Proof (Sketch). NP-hardness of reachability is proved by a reduction from the NP-complete SAT problem. Consider a SAT instance represented as a circuit of binary "NAND" ($\neg(x \wedge y)$) operations. One can construct a net such that its runs correspond to the input nodes of the circuit nondeterministically picking arbitrary input values, and the operation nodes of the circuit evaluating the function given the chosen values of the inputs. The technical details are provided in the appendix.

5 Second restriction: token counts

Another property of the PSPACE-hardness reduction for IO nets is the low number of tokens in each place. Specifically, no reachable marking puts more than one token in any place. Some systems exhibit a very different behaviour. For instance in most cases of chemical reaction networks, we expect the number of individual molecules to be much larger than the number of species of molecules. Additionally, we do not expect any chance "near-misses" between the configuration of the molecules before and after a reaction sequence. In other words, if the total amount of molecules of some group of species before the reaction sequence is approximately equal to the amount of molecules of some other group of species afterwards, there must be a precise equality. Informally, we can consider an example from [5] cited in [1]. Five species of molecules are considered in a milliliter-scale cell with nanomolar (picomole per milliliter) concentrations of molecules. As a picomole contains more than 10^{11} molecules, equalities that hold up to 10^3 molecules have a relative error of 10^{-8}. Such equalities can reasonably be expected to follow from some conservation laws and be precise.

This behaviour can be formalized by the following condition.

Definition 3. *A pair of markings M and M' of an IO net with the set of places P is a* near-miss *pair, if for there exists sets of places X and Y such that $0 < |M(X) - M'(Y)| \leq |P|^3$. A pair which is not a near-miss is called a* no-near-miss *pair.*

In terms of our informal example, even for a hundred molecular species a near-miss corresponds to an absolute error of at most 10^6 molecules, which is low compared to the molecule numbers in many applications.

Observe that each place of markings M and M' such that M, M' are a no-near-miss pair can be either unmarked or contain at least $|P|^3$ tokens. This can be seen by examining sets $X = \{p\}$ and $Y = \emptyset$, or $X = \emptyset$ and $Y = \{p\}$.

Applications avoiding near-miss markings enjoy easier reachability problem.

Theorem 5. *The reachability problem for no-near-miss pairs of markings is in P. Moreover, there is a polynomial-time algorithm such that for every pair of markings M, M' it either resolves reachability, giving a witness firing sequence if it exists, or reports a near-miss in M and M'.*

Remark 1. Requiring only that the initial and final markings of a firing sequence have many tokens in the non-empty places does not give us a better complexity than the general PSPACE-complete case.

The basic idea of the algorithm is to maintain an increasing set of restrictions. Once we cannot prove any new restrictions, we can either construct a firing sequence from M to M' satisfying the obtained restrictions and no other ones, show that the set of restrictions is unsatisfiable, or find a near-miss.

5.1 Restriction inference

Given an IO net N and two markings M and M', the algorithm expands an initially empty set $\mathcal{R}$ of restrictions of the form "no token can go from place p to place q via place r". We say that a pair of places (p, q) is *forbidden* if for all $r \in P$ the restriction "no token can go from p to q via r" is in $\mathcal{R}$. *Forbidding* a pair (p, q) means adding the restriction "no token can go from p to q via r" for all $r \in P$. A pair of places (p, q) that is not forbidden is *allowed*.

The algorithm alternatingly applies two operations which infer new restrictions, a reachability-based inference step and a cut-based inference step.

Reachability-based inference For each allowed pair (p, q), we keep two growing sets of places: the *initially-reachable set* $R_i(p, q)$ of places reachable from p, and the *finally-reachable set* $R_f(p, q)$ of places backwards-reachable from q.

A *reachability-based inference step* is performed as follows. We initialize $R_i(p, q)$ as $\{p\}$ and $R_f(p, q)$ as $\{q\}$. For every allowed pair (p, q), add place r' to $R_i(p, q)$ if:

- there is a transition $r \xrightarrow{s} r'$ for some places r, s,
- r is in $R_i(p, q)$ and r' is not,
- s is in $R_i(p', q')$ for some places p', q', and
- the restriction "no token can go from p to q via r'" is not in $\mathcal{R}$.

Symmetrically, for every allowed pair (p, q), add place r' to $R_f(p, q)$ if:

- there is a transition $r' \xrightarrow{s} r$ for some places r, s,
- r is in $R_f(p, q)$ and r' is not,
- s is $R_f(p', q')$ for some places p', q', and
- the restriction "no token can go from p to q via r'" is not in $\mathcal{R}$.

Once no initially-reachable or finally-reachable set can be extended, we define the reachable set $R(p, q)$ for each pair (p, q) as the intersection of $R_i(p, q)$ and $R_f(p, q)$ if they exist, and the empty set otherwise. If $R(p, q)$ does not contain both p and q, we forbid the pair, i.e. we add the restriction "no token can go from p to q via r" to $\mathcal{R}$ for all $r \in P$. Otherwise, for every place r not in $R(p, q)$, we add the restriction "no token can go from p to q via r" to $\mathcal{R}$.

Cut-based inference To describe the second kind of inference step, we define a correspondence between the reachability problem in an IO net with a restriction set and the maximum flow problem for a special graph.

Let N be an IO net of place set P, let M, M' be two markings of N, and let $\mathcal{R}$ be a set or restrictions of the form "no token can go from p to q via r". We define a flow graph $G = (V, A, c)$ with $2|P| + 2$ vertices. There are two vertices for each place $p \in P$, an "initial" copy v_p^i and a "final" copy v_p^f, as well as a distinguished inlet vertex i and a distinguished outlet vertex o. For each place $p \in P$, there is an arc $a = (i, v_p^i)$ with capacity $c(a) = M(p)$, and an arc $a = (i, v_p^f)$ with capacity $c(a) = M'(p)$. For each pair of places $(p, q) \in P^2$ such that (p, q) is not forbidden in $\mathcal{R}$, there is an arc $a = (v_p^i, v_q^f)$ from the initial p-labeled vertex to the final q-labeled vertex with infinite capacity. Note that the maximum flow value in graph G thus constructed is at most $|M| = |M'|$.

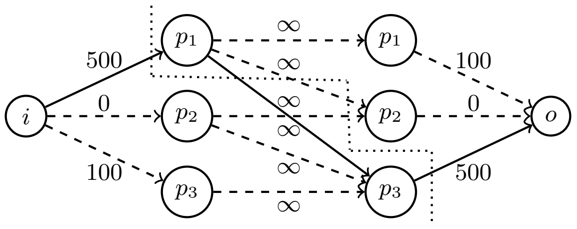

Fig. 3: Flow graph for the IO net of Figure 1 with a cut.

Given such a flow graph $G = (V, A, c)$, we define two operations on the capacity c relative to a place pair $(p, q) \in P^2$ and a integer $k > 0$:

- *Increasing c by k along* (p, q) consists in increasing $c(i, v_p^i)$ and $c(v_q^f, o)$ by k.
- *Decreasing c by k along* (p, q) consists in decreasing $c(i, v_p^i)$ and $c(v_q^f, o)$ by k. This operation is not possible if $c(i, v_p^i)$ or $c(v_q^f, o)$ are smaller than k.

A *cut-based inference step* is performed as follows. We construct the flow graph $G = (V, A, c)$. If the maximum flow value on G is smaller than $|M|$, we terminate the algorithm and report that M' is unreachable from M. Otherwise, we forbid each allowed pair (p, q) such that decreasing c by 1 along (p, q) is impossible or reduces the maximum flow value to $|M| - 2$.

The idea is to root out any pair (p, q) for which the path from i to o via (i, v_p^i) and (v_q^f, o) crosses the minimum cut twice.

Example 2. Figure 3 illustrates the flow graph G constructed for the IO net of Figure 1, the markings $M = (500, 0, 100)$ and $M' = (100, 0, 500)$, and the forbidden pairs $\{(p_2, p_1), (p_3, p_1), (p_3, p_2)\}$. The path $i \to v_{p_1}^i \to v_{p_3}^f \to o$ contains two arcs crossing the cut.

5.2 Firing sequence construction

When no new restrictions can be produced by applying any of the two kinds of inference steps, we say the set of restrictions $\mathcal{R}$ is *stable*. Given a stable set of restrictions with b allowed pairs (p, q), a *solution flow* is a result of the following procedure: Construct the flow graph G. Decrease the capacity by $|P|$ along each allowed pair; if this step fails because some arc has insufficient capacity, terminate the algorithm and report that M, M' is a near-miss pair. Otherwise, compute a maximal flow. If it has value less than $|M| - b \times |P|$, terminate the algorithm and report that M, M' is a near-miss pair. Otherwise, increase its capacity by $|P|$ along each (allowed) pair.

Observe that a solution flow does not always exist, and when it exists it might not be unique. The algorithm builds a firing sequence from the solution flow. For this we recall some definitions from [9].

Trajectories and histories Since the transitions of IO nets do not create or destroy tokens, we can give tokens identities. Given a firing sequence, each token of the initial marking follows a *trajectory* through the places of the net until it reaches the final marking of the sequence. The trajectories of the tokens between given source and target markings constitute a *history*.

A *trajectory* of IO net N is a sequence $\tau = p_1 \dots p_k$ of places. We let $\tau(i)$ denote the i-th place of τ. The *i-th step* of τ is the pair $\tau(i)\tau(i+1)$. A *history* H of length h is a multiset of trajectories of length h. Given an index $1 \leq i \leq h$, *the i-th marking of* H, denoted M_H^i, is defined as follows: for every place p, $M_H^i(p)$ is the number of trajectories $\tau \in H$ such that $\tau(i) = p$. The markings M_H^1 and M_H^h are the *initial* and *final* markings of H, and we write $M_H^1 \xrightarrow{H} M_H^h$. A history H of length $h \geq 1$ is *realizable* if there exist transitions $t_1, \dots, t_{h-1}$ and numbers $k_1, \dots, k_{h-1} \geq 0$ such that

- $M_H^1 \xrightarrow{t_1^{k_1}} M_H^2 \cdots M_H^{h-1} \xrightarrow{t_{h-1}^{k_{h-1}}} M_H^h$, where for every t we define $M' \xrightarrow{t^0} M$ iff $M' = M$.
- For every $1 \leq i \leq h-1$, there are exactly k_i trajectories $\tau \in H$ such that $\tau(i)\tau(i+1) = p_s p_d$, where p_s, p_d are the source and target places of t_i, and all other trajectories $\tau \in H$ satisfy $\tau(i) = \tau(i+1)$. Moreover, there is at least one trajectory τ in H such that $\tau(i)\tau(i+1) = p_o p_o$, where p_o is the observed place of t_i. We say that t_i *realizes* step i of H.

We say that $t_1^{k_1} \cdots t_{h-1}^{k_{h-1}}$ realizes H. Intuitively, at a step of a realizable history only one transition occurs, although perhaps multiple times, for different tokens. From the definition of realizable history we immediately obtain:

- $M' \xrightarrow{*} M$ iff there exists a realizable history with M' and M as initial and final markings.
- Every firing sequence that realizes a history of length h has accelerated length at most h.

From solution flow to firing sequence Let f be a solution flow, and $\mathcal{R}$ the final stable restrictions set of the algorithm. Intuitively, our construction of the solution flow makes sure the flow has value at least $|P|$ along each pair (p, q) which is allowed by $\mathcal{R}$. We use the procedure for reachability-based inference to construct a realizable history from this flow, such that for every pair (p, q) there are at most $f(v_p^i, v_q^f)$ trajectories from p to q.

Definition 4. *A place r' is an* initially-reachable child *of place r for pair (p, q) if r' was added to $R_i(p, q)$ because of some transition $r \xrightarrow{s} r'$. The notion of* initially-reachable descendant *is defined by transitive and reflexive closure over the initially-reachable child relation. The corresponding notions of* finally-reachable child *and* finally-reachable descendant *are defined symmetrically.*

The defined relations and functions can be computed by rerunning a reachability-based inference step with the stable set $\mathcal{R}$. Notice also that a reachability-based inference step on $\mathcal{R}$ provides no new restrictions, so $R_i(p, q) = R_f(p, q) = R(p, q)$.

We define three markings M_m, M_i and M_f. Let M_m be the marking such that $M_m(r)$ is equal to the cardinality of the set $\{(p, q)|r \in R(p, q)\}$ for all r. Let M_i be the marking such that $M_i(p) = \sum_q |R(p, q)|$. Note that as $|R(p, q)| \leq |P|$ we have $M_i(p) \leq f(i, v_p^i)$. Symmetrically, let M_f be the marking such that $M_f(q) = \sum_p |R(p, q)|$; we have $M_f(q) \leq f(v_q^f, o)$. We are going to construct a history from M_i to M_m and from M_m to M_f.

We construct the realizable history from M_i to M_m by running a reachability-based inference step. The first step of the history consists in trajectories of length 1 such that there is exactly one trajectory in p for each triple (p, q, r) such that $r \in R(p, q)$. We label each trajectory with its triple. This first step corresponds to the marking M_i. The idea is to extend each trajectory of M_i labeled (p, q, r) from p until it reaches place r, and then to do the same, working backwards, for each trajectory of M_f labeled (p, q, r) from q until r.

At each step of the construction of the history we maintain the fact that a (p, q, r)-labeled trajectory is in place $\hat{r}$ such that $\hat{r}$ is the latest place with r as initially-reachable descendant to be added to $R_i(p, q)$. In the first step described above, this holds by initialization of the R_i sets.

At each step, we pick the next pair (p, q) and place r' such that the reachability-inference step adds r' to $R_i(p, q)$. It is added because of a transition $\hat{r} \xrightarrow{s} r'$. For every place d which is a descendant of r', we extend trajectories labeled (p, q, d) with a step from $\hat{r}$ to r'. The rest of the trajectories in the history are extended with steps preserving their corresponding current places. By definition of a reachability-inference step, s is already in $R_i(p', q')$ for some p', q' so the history thus defined is realizable. Eventually all the trajectories reach the place r of their label (p, q, r).

We construct a realizable history from M_m to M_f in a symmetrical way. We concatenate these two histories (identifying the trajectories labeled (p, q, r) in them) to obtain a history from M_i to M_f with $|R(p, q)| \leq |P| \leq f(v_p^i, v_q^f)$ trajectories from p to q. We pick an arbitrary trajectory from p to q and increase its multiplicity in the multiset by $f(v_p^i, v_q^f) - |R(p,q)|$. This provides a realizable history from M to M'.

Finally, we extract a firing sequence from the realizable history from M to M' by associating a transition and an iteration count to each step of the history. Each step with k trajectories going from p_s to p_d with $p_s \neq p_d$ is associated to a transition t iterated k times from p_s to p_d, where t realizes the step. This is possible by realizability.

5.3 Algorithm Correctness

We recall the general structure of the algorithm. The algorithm initializes the set of restrictions to be empty, then alternates reachability-based and cut-based inference steps until a stable set of restrictions is reached (or an early termination occurs). The stable set of restrictions is then used to build a solution flow (or report a near miss); a solution flow can always be converted into a firing sequence from M to M'. We now prove that the algorithm runs in polynomial time and always returns a correct answer. In case of a near-miss, both reporting the near-miss and correctly resolving reachability is considered a correct answer.

Lemma 1. *The algorithm runs in polynomial time.*

The runtime analysis is straightforward, and can be found in the appendix. To prove that the algorithm is correct, we prove that the inference steps only produce correct restrictions and that the algorithm's reports of non-reachability and near-misses are correct.

Lemma 2. *The restrictions and non-reachability reports are correct.*

Proof (Sketch). We say that a restriction "no token can go from p to q via r" is correct if there exists no realizable history from M to M' with a trajectory from p to q passing through r. The proof follows from the fact that if M can reach M', there exists a realizable history from M to M' with trajectories from p to q for each $p, q \in P$ such that $M(p) > 0$ and $M'(q) > 0$, and which induces a flow of size $|M| = |M'|$ over the flow graph G of the cut-inference step. The existence of this flow of size $|M|$ also entails that the non-reachability reports are correct. The details are in the appendix.

The correctness proof of the near-miss reports is more involved, and interesting.

Lemma 3. *The near-miss reports are correct.*

Proof. The algorithm only reports a near-miss in the solution flow procedure, over a stable set of restrictions $\mathcal{R}$. Such a report entails that we have flow graph $G = (V, A, c)$ with the following properties:

– the maximal flow value is $|M|$,

- decreasing c by 1 along any allowed pair of $\mathcal{R}$ decreases the maximum flow value to $|M| - 1$ (by stability of $\mathcal{R}$),
- decreasing c by $|P|$ along all the b allowed pairs of $\mathcal{R}$ is either impossible or leads to a maximum flow value less than $|M| - b \times |P|$.

If decreasing c by $|P|$ along all the b allowed pairs of $\mathcal{R}$ is impossible, then either there is some place p such that the arc (i, v_p^i) has capacity less than $|P| \times |P|$, or there is some place q such that the arc (v_q^f, o) has capacity less than $|P| \times |P|$. This is equivalent to either $M(p) < |P| \times |P|$, which is a near-miss as $0 < |M(\{p\}) - M'(\varnothing)| < |P|^3$; or $M'(q) < |P| \times |P|$, which is a near-miss as $0 < |M(\varnothing) - M'(\{q\})| < |P|^3$.

Assume that instead, decreasing c by $|P|$ along all the b allowed pairs of $\mathcal{R}$ leads to a maximum flow value less than $|M| - b \times |P|$. We call c' the capacity post-decrease, and note $G' = (V, A, c')$. Theorem 1 on equality of the minimum cut and the maximum flow gives existence of a cut in G' with capacity less than $|M| - b \times |P|$. Consider such a cut (V_I, V_O) of capacity $\kappa' < |M| - b \times |P|$. We write κ the capacity of cut (V_I, V_O) in G before the decrease operation. Since the maximum flow, and thus minimum cut, of G is $|M|$, we have $\kappa \geq |M|$. Therefore there exists an allowed pair (p, q) such that the arcs (i, v_p^i) and (v_q^f, o) both cross the cut, as otherwise $\kappa' \geq |M| - b \times |P|$. We also know that as the cut-based inference did not produce new restrictions, decreasing by 1 along any allowed pair keeps any cut capacity in G bigger or equal to $|M| - 1$. Thus we have $\kappa > |M|$. By structure of G and G', the decreasing operation can reduce a cut capacity by at most $2b \times |P|$. So $\kappa - \kappa' \leq 2b|P|$, and using the inequalities above as well as the fact that there are at most $b \leq |P|^2$ allowed pairs, we get $|M| < \kappa < |M| + |P|^3$.

Consider the following two vertex sets based on cut (V_I, V_O). Let $X = V_I \cap \left\{v_p^i | p \in P\right\}$ and $Y = V_I \cap \left\{v_p^f | p \in P\right\}$. Our cut is finite, so only finite capacity arcs cross it, namely the arcs from the inlet to vertices v_p^i and from vertices v_p^f to the outlet. The capacity in G of this cut is thus $\kappa = M(P \setminus X) + M'(Y)$. Since $|M| < \kappa < |M| + |P|^3$ and $|M| = M(P)$, we know $0 < M(P \setminus X) + M'(Y) - M(P) < |P|^3$. By set considerations $M(P) - M(P \setminus X) = M(X)$, and so finally $0 < M'(Y) - M(X) < |P|^3$. The sets X, Y prove that M, M' are a near-miss.

If the algorithm does not report non-reachability or a near-miss, it returns a solution flow. We have shown that we can construct a firing sequence from M to M' from this flow, and it can be done in polynomial time.

6 Conclusion and future work

We have considered two restrictions of the IO net reachability problem with a promise for much simpler verification for some applications and established the reachability complexity in both these cases, which is NP-complete in one case and polynomial in the other.

We leave the question of complexity of set-set reachability under these restrictions for future research. Another related question is defining a notion of "approximate" reachability that would provide a reduction in complexity for IO nets, as merely bounding the maximum difference between token counts or the sum of differences preserves PSPACE-hardness of the reachability problem.

References

1. David Angeli, Patrick De Leenheer, and Eduardo D Sontag. A petri net approach to the study of persistence in chemical reaction networks. *Mathematical biosciences*, 210(2):598–618, 2007.
2. Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
3. Paolo Baldan, Nicoletta Cocco, Andrea Marin, and Marta Simeoni. Petri nets for modelling metabolic pathways: a survey. *Nat. Comput.*, 9(4):955–989, 2010.
4. E. Cardoza, Richard J. Lipton, and Albert R. Meyer. Exponential space complete problems for petri nets and commutative semigroups: Preliminary report. In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA*, pages 50–54. ACM, 1976.
5. Gheorghe Craciun, Yangzhong Tang, and Martin Feinberg. Understanding bistability in complex enzyme-driven reaction networks. *Proceedings of the National Academy of Sciences of the United States of America*, 2006.
6. Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 24–33. ACM, 2019.
7. René David and Hassane Alla. Petri nets for modeling of dynamic systems: A survey. *Autom.*, 30(2):175–202, 1994.
8. E. A. Dinits. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Sov. Math., Dokl.*, 11:1277–1280, 1970.
9. Javier Esparza, Mikhail A. Raskin, and Chana Weil-Kennedy. Parameterized analysis of immediate observation petri nets. In *Petri Nets*, volume 11522 of *Lecture Notes in Computer Science*, pages 365–385. Springer, 2019.
10. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399404, 1956.
11. Huimin Lin. Stratifying winning positions in parity games. In Kees M. van Hee and Rüdiger Valk, editors, *Applications and Theory of Petri Nets, 29th International Conference, PETRI NETS 2008, Xi'an, China, June 23-27, 2008. Proceedings*, volume 5062 of *Lecture Notes in Computer Science*, pages 9–11. Springer, 2008.
12. Mikhail A. Raskin, Chana Weil-Kennedy, and Javier Esparza. Flatness and complexity of immediate observation petri nets. *CONCUR 2020 (to appear)*, 2020.

A First restriction: transition enabling

We describe the reduction from SAT to the reachability problem for non-forgetting IO nets.

Theorem 4. *Reachability problem for non-forgetting IO nets is* NP*-hard.*

Proof. NP-hardness of reachability is proved by a reduction from the SAT problem. Consider a SAT instance represented as a circuit of binary "NAND" ($\neg(x \wedge y)$) operations (any propositional formula can be converted into such form in linear time). We construct a net with the following places.

- For each input x_i of the SAT circuit we add places $x_i^{\perp}$, x_i^0, x_i^1. Informally, marking these places corresponds to the input value being unknown, set to 0 and to 1 respectively.
- For each operation node n_j, we add places $n_j^{(\perp,\perp)}$, $n_j^{(\perp,1)}$, $n_j^{(1,\perp)}$, n_j^0, n_j^1. Informally, these places correspond to our knowledge about the inputs and the output value of the node n_j: we can know neither input, know that one of the inputs is 1, or know the output value of the node being 0 or 1 (if one output is 0, the node has the value 1 regardless of the other input).

The transitions are as follows.

- A token can move from a place $x_i^{\perp}$ to either of the places x_i^0 or x_i^1.
- A token in one of the places $n_j^{(\perp,\perp)}$, $n_j^{(\perp,1)}$, $n_j^{(1,\perp)}$ can observe a token in p_k^0 or p_k^1 where p_k is an input to n_j and move to the place corresponding to its updated information about the arguments.
- Let n_o be the output operation node. Any token can observe a token in n_o^1 and perform any move that would be allowed by some observation (ensuring the non-forgetting property), or move to n_o^1.

The initial marking puts one token into each $x_i^{\perp}$ and $n_j^{(\perp,\perp)}$.

Such a net is a non-forgetting IO net, and it is easy to see that any execution in this net from the initial marking corresponds to guessing some inputs and evaluating the circuit. In particular, the marking with all the tokens in n_o^1 is reachable iff the circuit is satisfiable. This completes the proof.

B Second restriction: token counts

Below are the omitted or sketched proofs of the correctness of the polynomial algorith for reachability of no-near-miss pairs.

Lemma 1. *The algorithm runs in polynomial time.*

Proof. A reachability-based inference step tries to increase the initially-reachable and finally-reachable place sets of each allowed pair (p, q). There are at most $|P|^2$ such pairs and $2|P|^2$ such sets, each of which have size at most $|P|$. Increasing a set takes polynomial time. A cut-based inference step computes the maximum flow of a flow graph at most $|P|^2$ times. A single computation of the maximum flow can be done in polynomial time, for example by the Dinic algorithm.

A set of restriction can have size at most $|P|^3$ so the number of inference steps which add to it is polynomial. Once we obtain a stable set of restrictions, we compute a solution flow by computing a maximum flow, taking polynomial time.

Having a solution flow, we construct a realizable history. To do so we rerun a reachability-based inference step with a polynomial slowdown caused by the need to update the trajectories. Increasing the multiplicity of one trajectory for pair of places takes just a polynomial number of arithmetic operations. Each step of converting a realizable history to a description of a firing sequence requires reading all the (distinct) trajectories and their multiplicities, finding the non-horizontal steps, selecting a transition realizing the step and writing it and its multiplicity. All these operations are feasible in polynomial time.

Lemma 2. *The restrictions and non-reachability reports are correct.*

Proof. We say that a restriction "no token can go from p to q via r" is correct if there exists no realizable history from M to M' with a trajectory from p to q passing through r. The restrictions set is initialized with restrictions for every (p, q, r) such that $M(p) = 0$ or $M'(q) = 0$. These restrictions are correct.

Reachability-based inference is correct Consider a pair of places (p, q). If there exists a realizable history from M to M' with a trajectory from p to q passing through r, it is straightforward to see that r is in the initially-reachable and finally-reachable sets for (p, q). Therefore if r is not in the reachable set $R(p, q)$, there is no realizable history containing a trajectory from p to q via r. If p or q are not in the reachable set $R(p, q)$, there is no realizable history containing a trajectory from p to q.

Cut-based inference is correct Consider a pair of places (p, q) forbidden by a cut-based refinement. Any realizable history from M to M' induces a flow of value $|M|$: the flow that saturates all the arcs with the finite capacities (i.e. the arcs from the inlet and to the outlet), and assigns to an infinite-capacity arc from some p' to some q' the number of trajectories from p' to q'. If there is a realizable history from M to M', then consider such a flow. if this flow includes any flow from v_p^i to v_q^f, i.e. if $f(v_p^i, v_q^f) > 0$, we decrease the capacity by 1 along the pair (p, q).' The remaining flow has value $|M| - 1$. But since the cut-based inference forbids this pair, the maximum flow value should be strictly less than $|M| - 1$. The contradiction proves that there is no flow from v_p^i to v_q^f, and correspondingly, there are no trajectories from p to q in the history.

Non-reachability reports are correct Reports of non-reachability are given when the algorithm finds a maximum flow value less than $|M|$ in the cut-inference step. As a realizable history induces a flow of value $|M|$, non-reachability cannot be reported if such a history exists, therefore all non-reachability reports are also correct.