Reachability Analysis via Net Structure

Harro Wimmel, Karsten Wolf

Universität Rostock, Institut für Informatik

Abstract. Exploitation of the structure of a Petri net is widely believed to be an inefficient approach to solving the reachability problem. We show that structure analysis can be combined with integer programming and partial order reduction to obtain a fast reachability solver.

Keywords: Petri net, reachability problem, integer programming, structure analysis.

1 Introduction

The *reachability problem* for Petri nets, i.e. if a final marking can be reached in a given net from the initial marking, is known to be decidable [May84,Kos82,Lam92] but EXPSPACE-hard [Lip76]. Efficient tools exist, but they cannot solve all instances of the problem (at least not in a lifetime). Model checkers, symbolic [CMS06] or with partial order reduction [Wol10], have been used successfully to solve quite large reachability problems.

Here we present an approach that is a mixture of several methods, the main ones being integer programming and structure analysis. The *marking equation*, a linear system of integer equations, is known to be a necessary condition for reachability. When a solution of the marking equation is found, it may represent the parikh vector of a firing sequence solving the reachability problem or not. If it does, the firing sequence needs to be found, otherwise the marking equation can be constrained to discriminate the found solution. The needed constraints are found by analysing the net structure. Consider the example net in Fig. 1.

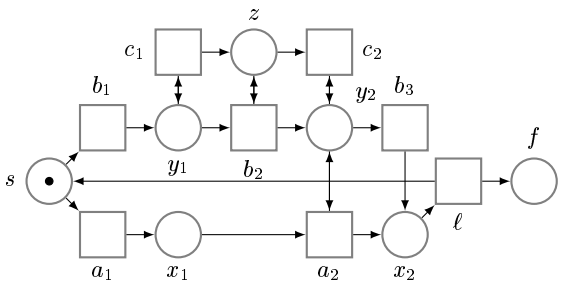

Fig. 1. An example Petri net N with initial marking s and final marking $s + 3f$

The marking equation $m_0 + Cx = m_f$ (where $m_0 = 1s$, $m_f = 1s + 3f$ are initial and final marking and C is the incidence matrix of the net N) can be solved by any integer programming tool. The smallest solution is $3a_1 + 3a_2 + 3\ell$, i.e. each of the three transitions should fire three times (in some unknown order). If we notice that a token is needed on y_2 to fire a_2, but none of the three transitions produces such a token, we might add a constraint "a token should be produced on y_2" to our marking equation to discriminate our first solution. A still viable solution is now $2a_1 + 2a_2 + b_1 + b_2 + b_3 + 3\ell$, as b_2 can produce that token. But now a token on z is missing. If required by a similar constraint, we come to $2a_1 + 2a_2 + b_1 + c_1 + b_2 + c_2 + b_3 + 3\ell$ but the token on y_2 is still not there when needed for a_2. Requiring a higher token production on y_2 will finally lead to the solution $3b_1 + c_1 + 3b_2 + c_2 + 3b_3 + 3\ell$ and now we "only" have to find a correct firing sequence, e.g. $b_1c_1b_2b_3\ell b_1b_2b_3\ell b_1b_2c_2b_3\ell$.

2 Some Basic Definitions

We expect the reader to be familiar with the basic Petri net terminology and some knowledge in linear algebra. All Petri nets here are general ones, i.e. they may have arbitrary multi-arcs including loops. Vectors are sometimes written as finite sums (multisets) over the vector's domain.

Definition 1 (Reachability problem). *A* reachability problem *is the question, when given a tuple (N, m, m') of a Petri net $N = (S, T, F)$ and two markings $m, m' \in \mathbb{N}^S$, whether m' can be reached from m, i.e. if $\sigma \in T^*$ with $m[\sigma\rangle m'$ exists.* The *reachability problem then is the set* $\mathsf{RP} = \{(N, m, m') \mid N = (S, T, F)$ *is a Petri net,* $m, m' \in \mathbb{N}^S$, $\exists \sigma \in T^*$: $m[\sigma\rangle m'\}$. *A reachability problem (N, m, m') is also called an* instance *of* RP*, to which the answer is "yes" if $(N, m, m') \in \mathsf{RP}$ and "no" otherwise.*

The reachability problem is decidable [May84] and making it solvable for as many instances as possible is our goal. It is well-known that a necessary condition for a positive answer to a reachability problem is the feasibility of the marking equation.

Definition 2 (Marking equation). *For a Petri net $N = (S, T, F)$ let $C \in \mathbb{N}^{S \times T}$, defined by $C_{s,t} = F(t, s) - F(s, t)$, be the* incidence matrix *of N. For two markings m and m' the system of linear equations $m + Cx = m'$ is the* marking equation *of N for m and m'. A vector $x \in \mathbb{N}^T$ fulfilling the equation is called a* solution.

For a firing sequence σ the *Parikh vector* $\wp(\sigma)$: $T \to \mathbb{N}$ is defined by $\wp(\sigma)(t) = \#_t(\sigma)$, where $\#_t(\sigma)$ is the number of occurrences of t in σ. If x is a solution of the marking equation $m + Cx = m'$, any firing sequence σ with $\wp(\sigma) = x$ and $m[\sigma\rangle$ positively solves the instance (N, m, m') of the reachability problem. From linear algebra the following is known:

Theorem 1 (Solution space). *For any marking equation $m+Cx = m'$ over a net $N = (S, T, F)$ there are finite sets of base vectors $B \subseteq \mathbb{N}^T$ and period vectors $P \subseteq \mathbb{N}^T$ such that all and only the solutions can be expressed as $b + \sum_i n_i p_i$ with $b \in B$, $p_i \in P$, and $n_i \in \mathbb{N}$.*

In other words, the solution space is a semilinear set over nonnegative integer vectors. Period vectors are nonnegative T-invariants where for a firing sequence σ with $\wp(\sigma)$ to be a T-invariant, $m[\sigma\rangle m$ must hold for all markings m enabling σ. Adding a T-invariant to a solution of the marking equation will produce another solution. Apart from multiples, our example net from Fig. 1 contains only one nonnegative T-invariant: $c_1 + c_2$. The base vectors take the form $i(a_1 + a_2) + (3 - i)(b_1 + b_2 + b_3) + 3\ell$ with $0 \leq i \leq 3$.

3 Traversing the Solution Space

Integer Programming (IP) solvers come in two flavors. Some can compute the whole solution space at once, but are too slow for practical purposes, others can only compute one solution. The latter, like *lp_solve* [BEN10], can be directed to compute a minimal solution (with respect to the sum over all values, leading to shortest firing sequences). Constraints can be used to discriminate a solution and force the IP solver to produce another (greater) one until no more solutions exist.

Definition 3 (Constraints). *We define two forms of constraints, both being linear inequations over transitions:*

- *a* jump constraint *takes the form $t < n$ with $n \in \mathbb{N}$ for a transition t.*
- *an* increment constraint *takes the form $\sum_{i=1}^{k} n_i t_i \geq n$ with $n_i \in \mathbb{N}$, $n \in \mathbb{N}$, and transitions t_i.*

Assume a linear system with a minimal solution b with $b(t) = n$, then an additional jump constraint $t < n$ discriminates b and leads to an incomparable solution. An increment constraint on the other hand may enforce a greater solution, adding some T-invariant to b. This idea is depicted in Fig. 2 where dashed arrows represent jumps and normal arrows the adding of T-invariants.

Since the solution space is semilinear we need jumps to get to other solution cones and increment constraints to go upwards in a cone. Jump and increment constraints can contradict each other, but it is possible to remove a jump constraint. Assume we have a solution a before and a solution b after adding a jump constraint. We construct one increment constraint per transition t with $t \geq b(t)$. This enforces at least the solution b. Removing the jump constraint now will not lead to an old (smaller) solution, especially not to a.

4 Building Constraints

Let us first argue that for a marking equation, any of the minimal solution vectors in B can be obtained by using jump constraints. For two solutions b and

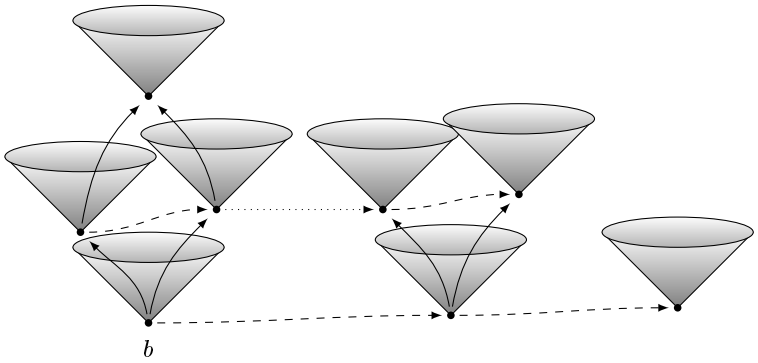

Fig. 2. Paths from the minimal solution b to any solution. Black dots represent solutions, cones stand for linear solution spaces over such solutions, which may or may not intersect or include each other. Normal arrows increment a solution by adding a T-invariant, dashed arrows are jumps to greater solutions. Such jumps can also occur on higher levels of linear solution spaces, shown by the dotted arrow

b' let $b \prec b'$ if we can change the solution our IP solver produces from b to b' by adding new constraints to a system consisting of the marking equation plus some (old) constraints.

Lemma 1 (Jumps to minimal solutions). *Let $b, b' \in B$ ($b \neq b'$) be base vectors of the solution space of the marking equation $m + Cx = m'$ plus some set of constraints $\mathcal{C}$. Assume b to be the minimal solution of the system. Then, we can obtain b' as output of our IP solver by consecutively adding jump constraints of the form $t_i < n_i$ with $n_i \in \mathbb{N}$ to $\mathcal{C}$.*

Proof. $b \prec b'$ and since b' is a minimal solution, $b \not\leq b'$. Thus, $\exists t \in T\colon b'(t) < b(t)$. Add the constraint $t < b(t)$ to $\mathcal{C}$, then b is not a solution anymore. Assume b'' to be our IP solver's new solution. As b' fulfills $t < b(t)$ it is still a solution, so from $b' \neq b''$ we conclude $b'' \prec b'$, and the same argument as above holds. Termination is guaranteed since there are only finitely many solutions $b'' \prec b'$.

Non-minimal solutions may not be reachable this way, since the argument "$b'(t) < b(t)$ for some t" does not necessarily hold. To determine those, increment constraints are necessary, and the latter can be obtained from partial solutions.

Definition 4 (Partial solution). *A* partial solution *of a reachability problem (N, m, m') is a tuple $(\mathcal{C}, x, \sigma, r)$ of*

- *a family of (jump and increment) constraints $\mathcal{C} = (c_1, \ldots, c_n)$,*
- *the $\prec$-smallest solution x fulfilling the marking equation of (N, m, m')* and *the constraints of $\mathcal{C}$,*
- *a firing sequence $\sigma \in T^*$ with $m[\sigma\rangle$ and $\wp(\sigma) \leq x$,*
- *a remainder r with $r = x - \wp(\sigma)$ and $\forall t \in T\colon (r(t) > 0 \Longrightarrow \neg m[\sigma t\rangle)$.*

The vectors x and r are included for convenience only, they can be computed from $\mathcal{C}$, σ, $\prec$, and the problem instance.

A full solution *is a partial solution $(\mathcal{C}, x, \sigma, r)$ with $r = 0$. In this case, σ is a firing sequence solving the reachability problem (with answer 'yes').*

If we obtain a partial solution with $r \neq 0$ there are not enough tokens on some places to fire the transitions in the remainder r. An underapproximation of tokens necessary can be computed from a graph G containing the transitions in r and the undermarked places, an edge from place s to transition t if there are not enough tokens on s to fire t, and an edge the other way if firing t increases the token count on s. From any strongly connected component SCC in G without incoming edges (= tokens produced by other components, a *source SCC*) we compute the minimal number k of tokens needed to activate any of its transitions. Our constraint now states that the number of tokens produced on the component's places should be increased by at least that number k. The tokens produced on a place p can be expressed as $\sum_{t:\, C(p,t)>0} C(p,t) \cdot x(t)$ for a solution vector x, so the constraint takes the form $\sum_{p \in SCC} \sum_{t \notin SCC:\, C(p,t)>0} C(p,t) \cdot t \geq k + \sum_{p \in SCC} \sum_{t \notin SCC:\, C(p,t)>0} C(p,t) \cdot x(t)$. Note that we sum up over transitions outside SCC only, as the transitions inside cannot produce tokens until the first of them gets activated, which is the aim of this constraint.

In our example net from Fig. 1 we start with the solution $x = 3a_1 + 3a_2 + 3\ell$. None of the transitions can fire three times, this leads us to the graph G: $y_2 \to a_2 \to x_2 \to \ell \to s \to a_1 \to x_2 \to a_2$. The only source SCC of G consists of just y_2 leading to the constraint $1 \cdot b_2 \geq k + 1 \cdot x(b_2) = k = 1$. Now the smallest solution becomes $x' = 2a_1 + 2a_2 + b_1 + b_2 + b_3 + 3\ell$ and only b_1 can fire as often as wanted. Our constructed graph G' now has z as its only source SCC, from which we obtain the constraint $1 \cdot c_1 \geq 1 + 1 \cdot x'(c_1) = 1$, leading to $x'' = 2a_1 + 2a_2 + b_1 + b_2 + b_3 + c_1 + c_2 + 3\ell$. After the sequence $b_1 c_1 b_2 c_2 b_3 \ell a_1$ we find a remainder $r = a_1 + 2a_2 + 2\ell$ and get again the graph G, but now with a constraint $1 \cdot b_2 \geq k + 1 \cdot x''(b_2) = 1 + 1 = 2$. After the next solution $x''' = a_1 + a_2 + 2b_1 + 2b_2 + 2b_3 + c_1 + c_2 + 3\ell$ with the graph $y_2 \to a_2 \to x_2 \to \ell \to s$ (a_1 not being in the remainder anymore) the constraint is increased to $1 \cdot b_2 \geq k + 1 \cdot x'''(b_2) = 1 + 2 = 3$. We now obtain the final solution of $3b_1 + 3b_2 + 3b_3 + c_1 + c_2 + 3\ell$. Note that the first step can also be done by a jump $a_1 < 3$, but this is impossible for the second step since $x'' \geq x'$.

5 Finding Partial Solutions

Finding maximal firing sequences σ for a solution x produced by the IP solver to obtain partial solutions can be done by a brute force tree search. The execution time may grow exponentially with the size of the solution, though. Partial order reduction, e.g. the stubborn set method [KSV06], can be applied to reduce the execution time. Other reductions can be thought of; even with stubborn sets a marking may appear more than once, either on a single path (the firing sequence is not minimal) or on permutated paths. Finding such spots allows to avoid going through subtrees unnecessarily.

6 Conclusion

The algorithm has been implemented in a tool named Sara [Wim10] and tested well with some small examples and a challenge posed by H. Garavel [Gar03] in 2003 so far. Four other tools, using different approaches, managed to solve the challenge, with run times from about 10 minutes to more than an hour (in 2003). A proof to the challenge consists of nearly 800 firing sequences of different length; those tools giving such a proof provided (much) longer firing sequences than our algorithm. Our implementation takes about 20 seconds on a simple linux PC and twice that much on a standard Windows PC under Cygwin (in 2010).

Due to the underapproximation of needed tokens when building constraints, our algorithm runs into trouble when the Petri net has high arc weights. A comparison to LoLA [Wol10] suggests an exponential loss with growing arc weights for some specialised examples.

Overall, we have the hope that our implementation is able to compete with other tools using state space exploration or symbolic model checking instead of structure analysis. Of course, more tests are necessary before a stable statement can be made. The algorithm presented here only tries to make a semi-decision for the positive case, but we are working on an extension to also make correct negative semi-decisions in many cases.

References

[BEN10] M. Berkelaar, K. Eikland, P. Notebaert: Lp_solve Reference Guide, `http://lpsolve.sourceforge.net/5.5/`, 2010.

[CMS06] G. Ciardo, R. Marmorstein, R. Siminiceanu: The saturation algorithm for symbolic state space exploration, *Software Tools for Technology Transfer* **8**:1, pp.4-25, 2006.

[Gar03] H. Garavel: Efficient Petri Net tool for computing quasi-liveness, `http://www.informatik.uni-hamburg.de/cgi-bin/TGI/pnml/getpost?id=2003/07/2709`, 2003.

[Kos82] S.R. Kosaraju: Decidability of reachability in vector addition systems, *Proceedings of the 14th Annual ACM STOC*, pp.267–281, 1982.

[KSV06] L.M. Kristensen, K. Schmidt, A. Valmari: Question-guided Stubborn Set Methods for State Properties, *Formal Methods in System Design* **29**:3, pp.215–251, Springer, 2006.

[Lam92] J. Lambert: A structure to decide reachability in Petri nets, *Theoretical Computer Science* **99**, pp. 79–104, 1992.

[Lip76] R.J. Lipton: *The Reachability Problem Requires Exponential Space*, Research Report 62, Yale University, 1976.

[May84] E. Mayr: An algorithm for the general Petri net reachability problem, *SIAM Journal of Computing* **13**:3, pp.441–460, 1984.

[Wim10] H. Wimmel: *Sara – Structures for Automated Reachability Analysis*, `http://www.informatik.uni-rostock.de/~nl/wiki/tools/download`, 2010.

[Wol10] K. Wolf: *LoLA – A low level analyzer*, `http://www.informatik.uni-rostock.de/~nl/wiki/tools/lola`, 2010.