Penrose: Putting Compositionality To Work For Petri Net Reachability

Paweł Sobociński and Owen Stephens

ECS, University of Southampton, UK

Abstract. Recent work by the authors introduced a technique for reachability checking in Petri Nets, exploiting compositionality to increase performance for some well-known examples. We introduce a tool that uses this technique, `Penrose`, discuss some design details in its implementation, and identify potential future improvements.

1 Introduction

The famous example of Dining Philosophers has n philosophers around a dining table, contending for the use of shared forks, in order to eat. A Petri net[1] representation of three dining philosophers is given in Fig. 1. The graphical notation is non-standard, with "directed" places and undirected links[2].

Independent sets of transitions of a (1 bounded) Petri net can *fire* if the current marking contains tokens in the source place(s) and none in the target place(s). *Reachability* is the problem of determining if a particular *marking*—a set of places that contain a token—can be reached by firing transitions, starting from some initial marking. In this paper we introduce `Penrose`[3] a tool, written in Haskell, for solving reachability in Petri nets, via an algebraic approach.

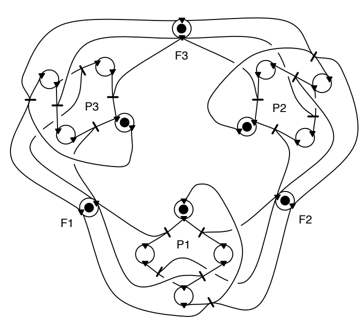


Fig. 1: Petri net representing a table of three dining philosophers.

[1] Here we consider 1-bounded Petri nets, aka C/E nets or Elementary Net Systems.

[2] Places in the graphical presentation have separate in/out ports, and distinct transitions are marked with a stroke. For details see [12].

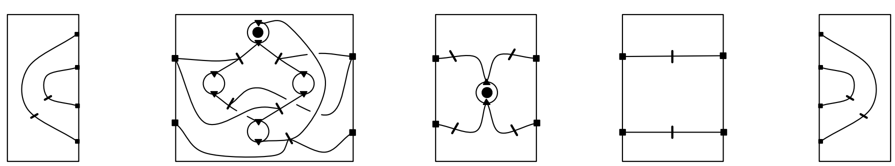
[3] Available for download: `http://users.ecs.soton.ac.uk/os1v07/Penrose_CALCO13`

Each Petri net determines a transition system with states the markings of the net and transitions that witness the simultaneous firing of an independent set of net transitions. For reachability, we consider the net's transition system as a non-deterministic finite automaton (NFA) over a unary alphabet: the initial and final states are, respectively, the initial and desired markings. Deciding reachability then coincides with emptiness of the NFA's language. This NFA is known as the *reachability graph* of a net; one algorithm is thus to construct the transition system, and determine if there is a path from the initial marking to the final one. State explosion makes this approach untenable: the number of markings (and thus, the statespace) is exponential in the number of places of the net.

1.1 A local approach: `Penrose`

Most standard approaches (e.g. [8,13]) to checking reachability are *monolithic* in that they consider a net as a whole. `Penrose` takes a different approach: decompose the net into small components (or take a decomposition as input), *locally* check reachability, and use the local information to reconstruct a global result. Our methodology is thus reminiscent of compositional model checking [5].

We use the algebra of *nets with boundaries* [11,3]. These extend Petri nets by adding left and right boundaries to a net, to which, transitions of the net can connect. They inherit the algebra of monoidal categories: composition can be "sequential", written ';', where two nets are synchronised, having their common boundary connected, or "tensor", written '⊗', where two nets are placed "on top" of one another and considered as a single net. If N is a net with boundaries, we write $N : k \to l$ if N has a left boundary of size k and right boundary of size l.



$d_2 : 0 \to 4$ $\quad$ $ph : 2 \to 2$ $\quad$ $fk : 2 \to 2$ $\quad$ $i_2 : 2 \to 2$ $\quad$ $e_2 : 4 \to 0$

Fig. 2: Component nets with boundaries of Dining Philosophers.

Using nets with boundaries, we can give a decomposition of the net in Fig. 1, in terms of 5 simple component nets, illustrated in Fig. 2: $PhRow_n$ is a row of n alternating philosophers and forks, and Ph_n a table of n dining philosophers:

$$PhRow_1 \stackrel{\text{def}}{=} ph \,;\, fk \qquad\qquad Ph_n \stackrel{\text{def}}{=} d_2 \,;\, (i_2 \otimes PhRow_n) \,;\, e_2$$
$$PhRow_{k+1} \stackrel{\text{def}}{=} ph \,;\, fk \,;\, PhRow_k$$

The Ph_n construction "seals" the row of philosophers into a table, by wiring the last fork to the first philosopher, using d_2, i_2 and e_2.

Consider the deadlocked configuration, where all forks are picked up, but no philosophers are eating; this situation corresponds to a marking with fork places empty, and the "eating" places unmarked in each philosopher. A net with boundaries $N : k \to l$ determines an NFA in a similar fashion to a Petri net, however, the labels of the NFA's transitions are now important—they record the interaction of underlying sets of net transitions on the boundaries, as a pair of k-bit and l-bit binary strings[4]. The minimal DFA for $PhRow_2$ with the desired (local) marking is shown in Fig. 3 (the error state has been omitted, to increase readability.) The DFA illustrated is actually a fixed point: for $n \geq 2$, the minimised DFA of $PhRow_n$ is the same[5]. Similar observations have been made about dining philosophers modelled using the algebra of Span(Graph) [7].

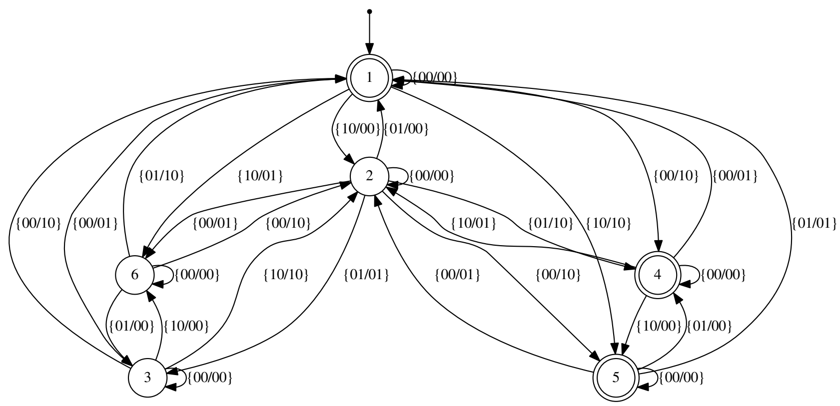


Fig. 3: Minimal DFA for $PhRow_2$.

The automata representing the underlying net components are ';'- or '$\otimes$'-composed through modifications of the standard product construction.

The high-level algorithm of `Penrose` is:

1. Take as input a reachability problem, comprised of a Petri net, considered as a net with boundaries, and initial and final markings.
2. Transform the net into a *wiring decomposition* [12], or take one as input[6]. A wiring decomposition is a binary tree, with nodes composition operators, and leaves nets with boundaries (with local marking information).

[4] A '1' in the i^{th} position indicates the presence of an interaction with the i^{th} boundary port, on the corresponding side.

[5] In general, when we talk about *fixed points* we are referring to a situation in which there exists $k \in \mathbb{N}$ such that $\mathsf{Min}(N_k) = \mathsf{Min}(N_{k+1})$, where N_n is a recursively defined net built up from component nets with boundaries, and Min is some minimisation operation (DFA minimisation, quotienting by weak bisimilarity, etc.).

[6] Many *real* Petri nets have recursive specifications that are readily translated to the language of nets with boundaries.

3. Traverse the wiring decomposition, avoiding duplicate work via memoisation:
   – convert leaves (once per unique net) into corresponding NFAs and construct minimal DFAs to discard irrelevant local statespace.
   – at internal nodes, combine DFAs (once per unique pair) using either form of composition, and then minimise the resulting DFA.

DFA minimisation can be prohibitively expensive. On the other hand, it is a very coarse equivalence that allows us to prune the statespace aggressively, and in some examples allows us to reach a fixed point quickly, when using a finer equivalence would not suffice, as explained in Sec. 2. Properties of nets with boundaries [11,3] ensure correctness, see [12] for proofs.

2 Applicability and Performance

State explosion is common in model checking of concurrent systems; by *minimising* the automata of component nets, we obtain the minimal characterisation of their "protocol": how they must interact with the environment in order to reach a desired configuration. Prior to minimisation, we ϵ-close[7] the NFA, since only actions that interact with the net's boundaries affect its protocol. We then determinise and minimise, using Brzozowski's [4] algorithm. This algorithm is conceptually very simple: the NFA's transition relation is twice reversed and determinised (using the subset construction).

The performance of `Penrose` depends on two factors:

1. The *structure* of the input net: can we identify repeated "small" components?
2. The *semantics* of a decomposition: does the statespace explode or grow slowly as the net is reconstructed; do we reach a fixed point?

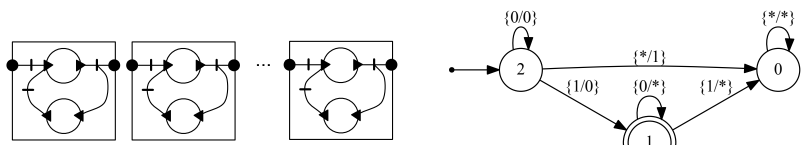
An initial investigation into structural issues has been carried out in [10]. Through memoisation, we can avoid duplicate work if repeated structure of a net is exposed and leads to a fixed point; for example, given the decomposition $PhRow_n$, no extra work is required to check reachability for $n \geq 3$, since a fixed point is reached at $n = 2$. This leads to performance that sometimes asymptotically outperforms monolithic approaches, see [12] for experiemental results. Characterising the underlying semantic issues is an open research question: *why* do Dining Philosophers reach a fixed point at $n = 2$?

2.1 Minimisation and fixed points

Minimisation using Brzozowski's algorithm is potentially very expensive, since the subset construction is performed twice. Indeed, `Penrose` performs well only if small automata are minimised. The advantage to minimising w.r.t. language equivalence is that statespace is pruned aggressively—in particular, branching is discarded—and thus the likelihood of finding a fixed point[8] is greater.

[7] ϵ-transitions are those with labels 0^*, indicating internal behaviour.
[8] `Penrose` finds fixed points via memoisation.

One alternative would be to quotient NFAs by (weak-) bisimilarity, obtaining smaller, equivalent NFAs without exponential blowup. However, on many examples bisimilarity is too fine an equivalence and fixed points do not exist because of branching, which is irrelevant for reachability. Indeed, quotienting by weak bisimilarity results in a fixed point only in deterministic variants[9] of the Dining Philosophers. We give another simple example of this phenomenon for the “replicators” of Fig. 4a. A replicator component can output an unbounded



(a) Chain of n Replicators

(b) Minimal DFA for n Replicators.

number of tokens on the right after receiving a single token as input on the left. Consider a chain of n replicators, with the desired marking having a token only in the upper place of each; the chain’s protocol is simple, and furthermore, is identical irrespective of n: after a single token has been received by the first replicator, it can be percolated through the chain with no interaction on the outermost boundary ports. This protocol is a fixed point reached at $n = 1$ and is the DFA shown in Fig. 4b. Quotienting by weak bisimilarity does not induce a fixed point. Therefore, in this example, the initial cost of determinisation pays off, whereas quotienting by bisimilarity is prohibitively expensive for large n.

3 Representing transition functions with BDDs

Recall that a net with boundaries N determines an NFA L, with labels binary strings of length $l = |\text{boundaries}(N)|$. A simple data structure for such an NFA is a set of pairs (s, f), where $s \in \mathsf{states}(L)$ and $f : \{0, 1\}^l \to 2^{\mathsf{states}(L)}$, that is, a source state and function from labels to sets of (target) states.

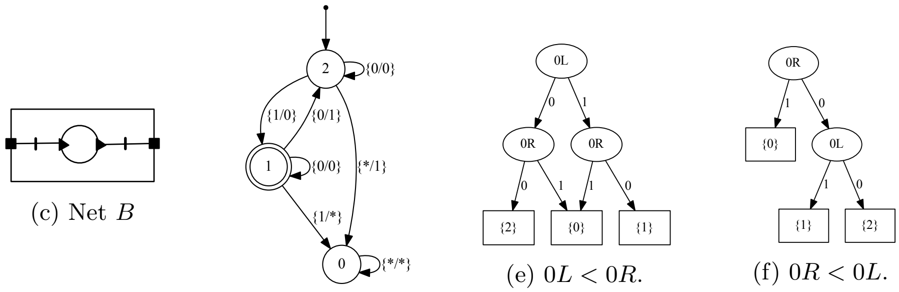
Reduced Ordered Binary Decision Diagrams (ROBDDs, or commonly just BDDs) are a compact representation of n-ary binary functions [1]. `Penrose` uses a generalisation of BDDs, Multi-Terminal BDDs [6], encoding functions with codomain the Boolean algebra of subsets $2^{\mathsf{places}(N)}$, rather than the Booleans.

As an example, consider the one place buffer net, B, and the reachability problem $(B, \langle absent \rangle, \langle present \rangle)$; we show B, the corresponding NFA and two alternative BDDs representing state 2’s transition function, in Fig. 4.

BDDs are not necessarily minimal, with their size sensitive to variable ordering; for us, the lexicographical order[10] on boundary ports. Indeed, by reversing

[9] For example, philosophers that always take their left fork first.

[10] The ordering gives an interleaving left-right, top-bottom, on boundary ports (i, s) where $i \in \mathbb{N}, s \in \{L, R\}$ and $L < R$.



(c) Net B (d) NFA for B (e) $0L < 0R$. (f) $0R < 0L$.

Fig. 4: One place buffer net with boundaries B, NFA for B with the initially empty, finally full marking and non-minimal and minimal BDDs for state 2.

the variable ordering of Fig. 4e we obtain a smaller BDD, illustrated in Fig. 4f. For larger BDDs, the effect of reordering variables can be more dramatic. Computing optimal variable ordering is NP-Complete [2].

`Penrose` represents NFA transitions with a collection of BDDs, one for each state, but doing so loses potential sharing of common targets. For example, in Fig. 4d, the BDDs for state 2 and state 1 transitions will both have leaves for $\{ 0 \}$. One possible solution is to use a data-structure similar to that of Minato et al. [9], where a single BDD is referenced by multiple "pointers" to nodes of the BDD graph, thereby retaining sharing.

4 Future Work

`Penrose` is under active development, currently supporting basic reachability-checking functionality outlined in this paper. We have obtained encouraging experimental results, sometimes asymptotically improving on monolithic approaches. Future work will investigate applying our decomposition technique to model checking problems other than reachability; optimising BDD representation, particularly w.r.t. the performance of the NFA to minimal DFA procedure; and further development of an algorithm for *automatic* decomposition of nets.

References

1. H. R. Andersen. An Introduction to Binary Decision Diagrams. Technical University of Denmark, 1997.
2. B. Bollig and I. Wegener. Improving the Variable Ordering of OBDDs Is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.
3. R. Bruni, H. C. Melgratti, U. Montanari, and P. Sobociński. Connector algebras for C/E and P/T nets' interactions. *LMCS*, 2013. To appear.
4. J. Brzozowski. Canonical Regular Expressions and Minimal State Graphs for Definite Events. In *Mathematical Theory of Automata*, volume 12 of *MRI Symposia*, pages 529–561. Polytechnic Institute of Brooklyn, 1962.
5. E. M. Clarke, D. Long, and K. McMillan. Compositional Model Checking. In *LiCS'89*, pages 352–362, 1989.
6. M. Fujita, P. C. McGeer, and J. C.-Y. Yang. Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation. *Formal Methods in System Design*, 10(2-3):149–169, 1993.
7. P. Katis, N. Sabadini, and R. Walters. Compositional Minimization in Span(Graph): Some examples. *ENTCS*, 104C:181–197, 2004.
8. K. McMillan. A Technique of a State Space Search Based on Unfolding. *Form Method Syst Des*, 6(1):45–65, 1995.
9. S.-i. Minato, N. Ishiura, and S. Yajima. Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation. In *Proc. DAC '90*, pages 52–57. ACM Press, 1990.
10. J. Rathke, P. Sobociński, and O. Stephens. Decomposing Petri nets. arXiv:1304.3121v1, 2013.
11. P. Sobociński. Representations of Petri net interactions. In CONCUR '10, pages 554–568. Springer, 2010.
12. P. Sobociński and O. Stephens. Reachability via compositionality in Petri nets. arXiv:1303.1399v1, 2013.
13. P. Starke. Reachability Analysis of Petri Nets using Symmetries. *Syst. Anal. Model. Sim.*, 4/5:292–303, 1991.