In Proc. 18th German Workshop on Algorithms and Tools for Petri Nets (AWPN 20101), Fernuniv. Hagen, September 2011.

Atomic Fragments of Petri Nets – Extended Abstract –

Monika Heiner[1], Harro Wimmel[2], and Karsten Wolf[2]

[1] Brunel University Uxbridge/London, on sabbatical leave from
Brandenburgische Technische Universität Cottbus
Institut für Informatik
[2] Universität Rostock
Institut für Informatik

Abstract. An atomic fragment of a Petri net is a maximal set of places or a maximal set of transitions such that the nodes in the set cannot be distinguished by any invariant and they induce a connected subnet. Atomic fragments are quite useful in understanding models, e.g. in biochemical reaction networks. We introduce a technique based on linear programming that efficiently finds the atomic fragments.

1 Introduction

Consider a place/transition net N with is incidence matrix C. Let a place invariant i be an (integer) solution of the system $iC = 0$ and a transition invariant an (integer) solution j of $Cj = 0$. Let an invariant be semipositive iff all its entries are greater or equal to 0.

In this paper, we consider a relation $\equiv$ between nodes of the net that is induced by its invariants. For two places (or two transitions) x and y, let $x \equiv y$ iff, for all semipositive place (transition, resp.) invariants i, $i(x) = 0$ if and only if $i(y) = 0$. This relation is obviously an equivalence. It is moreover evident that equivalent nodes must have a close functional connection. This is even more true if equivalent nodes are close to each other in the net topology. So let an *atomic fragment* of the net be a maximal set A of nodes of the same type such all nodes in A are pairwise equivalent (in the aforementioned sense) and connected (i.e. there is an undirected path between each pair of nodes in A that only passes nodes in A and nodes of the opposite type). Indeed, atomic fragments can be a useful tool for understanding models.

This extended abstract is organised as follows. In Sect. 2, we briefly discuss the use of atomic fragments in a biochemical setting. Then, in Sect 3, we present our approach for computing atomic fragments. Finally, we report on an implementation and initial experiments in Sect. 4.

2 Background

Petri nets represent a natural choice for modeling and analyzing biochemical networks as both share the bipartite and concurrent paradigm. Places usually

model species, or any kind of chemical compounds, e.g. proteins or proteins complexes, playing the role of precursors, products, or enzymes of chemical reactions. Complementary, transitions may stand for any kind of chemical reactions, e.g. association, dissociation, phosphorylation, or dephosphorylation, transforming precursors into products, possibly controlled by enzymes. The arcs go from precursors to reactions (ingoing arcs), and from reactions to products (outgoing arcs), while arc weights reflect known stoichiometry. Opposite to technical networks, biochemical networks tend to be very dense and apparently unstructured making the understanding of the full network of interactions difficult and, therefore, error-prone.

Getting a survey on the current state of knowledge about a particular pathway requires a lot of reading and search through data bases, including the creative interpretation of various graphical representations. These pieces of separate understanding are then assembled to get a comprehensive as well as consistent knowledge representation. The transformation from an informal to a formal model involves the resolution of any ambiguities, which must not necessarily happen in the right way. Therefore, the next step in a sound model-based technology should be devoted to model validation.

Place and transition invariants are a popular validation technique for technical as well as biochemical networks. One approach of acquiring a deeper understanding of the network behavior consists in understanding all its basic executions, which correspond to the minimal transition invariants. Another approach which has been proposed in [Hei09] goes one step further by looking for invariants-induced connected subnetwork defining a partition – the atomic fragments[3].

Atomic fragments permit a structured representation of invariants by network coarsening, and the automatically derived hierarchical representation of a given network supports its comprehension. Most remarkably, the approach requires static analysis only. Consequently, it can also be applied to unbounded systems.

In this paper we discuss the efficient computation of atomic fragments. For more details and examples demonstrating the hierarchical structuring principle see [Hei09].

3 Computation

Consider first the implementation of the equivalence $\equiv$ with respect to the invariants. Staying close to the definition, we can check the equivalence of x and y by considering two linear programming (LP) problems: $Ci = 0, i \geq 0, 0, i(x) > 0, i(y) = 0$ and $Ci = 0, i \geq 0, i(x) = 0, i(y) > 0$. Feasibility of either of the two systems means that x and y are not equivalent while nonfeasibility of both systems proves x and y to be equivalent.

Being based on an equivalence, atomic fragments partition the set of all nodes of the net. Consequently, we can use Tarjan's Union/Find data structure

[3] called abstract dependent transition sets in [Hei09]

[Tar75] for managing the fragments. This means that the test for equivalence needs only be performed for pairs of nodes in different fragments. If the test reveals equivalence, the two fragments can be unified. The adjacency criterion can be implemented by checking only for equivalence of two nodes x and y that are in different fragments and are adjacent (i.e. there is a node z such that $\{x, y\} \subseteq {\bullet}z \cup z{\bullet}$.

For increasing efficiency, we also use a dual partition of the nodes that is initially set to a singleton set consisting of all nodes. Whenever one of the considered LP problems returns with a solution i , we split every set in this partition into those nodes x where $i(x) > 0$ and those where $i(x) = 0$. This way, we converge to our final solution both bottom-up and top-down. Additionally, we tried to reduce that number of calls to an LP solver by considering the net topology. If there is a z such that $z{\bullet} = \{x\}$ and ${\bullet}z = \{y\}$ then x and y must be equivalent. Generalizing this pattern, x and y are also equivalent if there is a z such that $z{\bullet} = \{x\}$, $y \in {\bullet}z$, and all nodes in ${\bullet}z$ are equivalent. Dually, x and y are equivalent if there is a z such that ${\bullet}z = \{x\}$, $y \in z{\bullet}$, and all nodes in $z{\bullet}$ are equivalent. Furthermore it is clear that nodes adjacent to nodes with empty preset or empty postset are not covered by any invariant.

The complexity of our implementation is best assessed in terms of the number of calls to an LP solver as all other efforts are negligible. We distinguish between successful calls (i.e. calls yielding equivalence) and unsuccessful calls (i.. calls yielding non-equivalence). Let n be the number of nodes of the considered type and f be the number of resulting fragments. Each successful call is actually the result of two calls to an LP solver. Upon each such call, two fragments are unified. Initially, we have n singleton fragments. That is, the number of successful calls to an LP solver is $2(n-f)$ which is linear in n in worst case. Unsuccessful calls need to be considered for each pair of resulting fragments. Observe that unsuccessful calls remain valid even if the compared fragments change: Let $x \in F_1$, $y \in F_2$ and $x \not\equiv y$. If F_1 is later on unified with F_3 and F_2 with F_4, we know without an additional test that $F_1 \cup F_3$ cannot be unified with $F_2 \cup F_4$. Thus, we have $\frac{f(f-1)}{2}$ unsuccessful instances requiring up to $f(f-1)$ calls to an LP solver. IN consequence, depending on the ration between the number of nodes and the number of resulting fragments, the number of calls to an LP solver is between linear and quadratic in the number of nodes of the net.

In face of the well-known fact that integer LP solving is NP-complete, we have proposed a solution that can be executed in polynomial space. This is a clear advantage compared to the naive solution of the problem which consists of enumerating the (up to exponentially many) minimal semipositive invariants and to subsequently partition the set of nodes. Furthermore, our solution has an excellent potential for multicore or distributed implementation since the various LP problems can be solved independently. Last but not least, the present day technology for LP solving is quite advanced and not necessarily a blocker for practically relevant problem instances.

4 Tool and Experiments

Our approach is implemented in the tool Petia freely available at the web page `www.service-technology.org`. The tool is open source and available under an AGPL license. The tool reads a net, generates the various LP problems and ships them to the publicly available lp solver `lp-solve` [BEN]. Then it records the results in the two mentioned partitions and finally outputs the final partition. Some initial experiments have been conducted on a Macbook Pro with 2.2 GHz. For the n dining philosophers in the standard scenario (PHn), nets have $5n$ places and $4n$ transitions. Atomic fragments are all singletons, so a quadratic number of LP problems needs to be solved. Petia runs 2 seconds for $PH100$, 200 seconds for $PH500$, and almost 30 minutes for $PH1000$. For a data access protocol where processes can concurrently read and exclusively write to a data item (DATA n for n processes), atomic fragments are again all singletons. Petia solves the partition in 9 seconds for 100 processes and almost 2 minutes for 200 processes.

Additionally, we checked 1386 business process models with up to some 200 places and transitions (refer to [FFK+11] for more information on the models. These models have only few nontrivial classes, several models (typically acyclic ones) end up with a single class consisting of all nodes. Run time for a single model is not measurable Processing of the whole model library takes less than 10 seconds.

Currently we are in the process in applying Petia to our benchmark collection of biochemical networks.

5 Conclusion

We presented a method for determining the atomic fragments of a Petri net which can be used as a tool for better understanding a model and gaining new insights by hierarchical structuring. This structuring technique is especially helpful for analyzing biochemically interpreted Petri nets, where it supports model validation of biochemical reaction systems reflecting current comprehension and assumptions of what has been designed by natural evolution.

The proposed method requires only polynomial space, compared to an exponential naive solution. The prototype is efficient enough to solve academic problems beyond 1000 nodes and real-life problems. The method has an excellent potential for speed-up through parallel computing. The various LP problems can be solved independently, and almost no substantial data traffic is required between the processes. Thus, we conclude that the method is applicable for routine use.

References

[BEN] M. Berkelaar, K. Eikland, and P. Notebaert. lpsolve: Open source (mixed integer) linear programming system. Technical report, Eindhoven University of Technology.

[FFK+11] Dirk Fahland, Cédric Favre, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Analysis on demand: Instantaneous soundness checking of industrial business process models. *Data Knowl. Eng.*, 70(5):448–466, 2011.

[Hei09] M Heiner. *Understanding Network Behaviour by Structured Representations of Transition Invariants – A Petri Net Perspective on Systems and Synthetic Biology*, pages 367–389. Natural Computing Series. Springer, 2009.

[Tar75] R.E. Tarjam. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.