Decomposition into open nets

Stephan Mennicke and Olivia Oanea and Karsten Wolf

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{stephan.mennicke, olivia.oanea, karsten.wolf}@uni-rostock.de

Abstract. We study the decomposition of an arbitrary Petri net into open nets. This means that shared places can be seen as message channels between components. We show that there exists a unique decomposition into atomic components which can be efficiently computed. We further show that every composition of components yields a component and that every component can be built from atomic components. Finally, we briefly discuss potential applications.

1 Introduction

Several authors [6, 1] proposed open nets as a means for open components or web services. An open net is Petri net with an interface consisting of places. Each place in the interface is either an input or an output place.

We study the problem of decomposing an arbitrary Petri net into open nets. This means that we want to find open nets such that the whole net is structurally isomorphic to the composition of the components. The actual challenge is that a shared place must belong to exactly two components where it serves as input place for one component and as output place for the other.

We show that there are elegant and efficient techniques for handling a decomposition into open nets. In fact, every component constitutes of atomic components which can be computed in little more than linear time.

An implementation of the technique teaches as that practically relevant Petri nets decompose into many and small components. Hence, decomposition into open nets can be used for divide-and-conquer algorithms. A potential candidate is Commoner's property [4, 3] since Petri net structures such as siphons and traps relate quite regularly to open net components. Another interesting application could be the decomposition of business process models into web services where the atomic components can be aggregated into components of useful size by applying available role annotations in the activities.

2 Definitions

We use the standard notation $[P, T, F]$ for a (place/transition) Petri net. We require for all nets $|T| > 0$ and disallow isolated nodes. Arc weights are ignored for simplicity; their insertion would not cause any problem.

An open net $[P, T, F, \mathit{In}, \mathit{Out}]$ consists of a Petri net $N = [P, T, F]$, and an interface $I = [\mathit{In}, \mathit{Out}]$ with $\mathit{In} \cup \mathit{Out} \subseteq P$ and $\mathit{In} \cap \mathit{Out} = \emptyset$. In the interface, *In* is the set of input places and *Out* is the set of output places. We require ${}^\bullet\mathit{In} = \emptyset$ and $\mathit{Out}^\bullet = \emptyset$. A set M of open nets is composable iff their sets of transitions is disjoint and each place belongs to at most two nets in M. Thereby, it must be an input place in one net and an output place in the other. The composition $\oplus M$ of a composable set M of open nets is obtained by building the union of the respective constituents.

A decomposition of a Petri net $N = [P, T, F]$ is a set M of open nets such that $\oplus M = [P, T, F, I, O]$, for some I and O. $\{[P, T, F, \emptyset, \emptyset]\}$ is the trivial decomposition of $[P, T, F]$. A decomposition of an open net

$N = [P, T, F, \mathit{In}, \mathit{Out}]$ is a set M of open nets such that $\oplus M = N$. $\{N\}$ is the trivial decomposition of open net N.

Decomposition M_1 of N is finer than decomposition M_2 of N if, for each $N \in M_2$, there is a subset of M_1 which is a decomposition of N. M is a finest decomposition of N if it is finer than every decomposition of N.

3 Existence of a finest decomposition

Throughout this section, we consider a Petri net $N = [P, T, F]$. We prove the existence of a finest decomposition by construction. To this end, we inductively define an equivalence relation R on $P \cup T$. For every decomposition of N, different nodes that are related by R occur as inner nodes in the same component. In the following definition, let X^* be the reflexive and transitive closure of relation X.

Definition 1 (Relation R). *Let $R = \bigcup_{i \in \mathit{Nat}} R_i$ where*
(Base:) $R_0 = (\{[t_1, t_2] \mid \exists p : p \in P \cap {}^\bullet t_1 \cap {}^\bullet t_2 \text{ or } p \in P \cap t_1^\bullet \cap t_2^\bullet\})^*$.
(Step:) $R_{i+1} = (R_i \cup \{[p, t_1], [t_1, p] \mid \exists t_2 : p \in P \cap {}^\bullet t_1 \cap t_2^\bullet, [t_1, t_2] \in R_i\})^*$.

It is easy to see that all the R_i, and thus R, are equivalence relations. Note that the pairs $[p, t_2]$ and $[t_2, p]$ are inserted in the step as well, due to the transitive closure.

Lemma 1. *If $[x, y] \in R$ and $x \neq y$ then x and y occur as internal nodes in the same component in every decomposition of N.*

Proof. (Base:) Assume that $\exists p : p \in P \cap {}^\bullet t_1 \cap {}^\bullet t_2$. Then t_1 and t_2 cannot occur in different components since otherwise p must be an interface place where both components consume tokens, in contradiction to the definition of a decomposition. Both t_1 and t_2 are internal as transitions cannot be part of the interface. The case $p \in P \cap t_1^\bullet \cap t_2^\bullet$ is analogous. Building the reflexive and transitive closure does not destroy the property as "occurrence in the same component" is naturally a transitive relation.
(Step:) By the inductive assumption, $[t_1, t_2] \in R_i$ implies that t_1 and t_2 occur as internal nodes in the same component. As t_1 consumes tokens from p while t_2 produces tokens on p, p cannot be an interface place. Consequently, it must be an internal place in the same component as t_1 and t_2. Again, building the transitive closure is harmless. □

The following few lemmas prepare the actual construction of the finest decomposition.

Lemma 2. *Every non-singleton equivalence class contains transitions.*

Proof. No pairs between two places are inserted into R (except for transitive closures). □

Next we show that classes which contain transitions are not adjacent.

Lemma 3. *If $[p, t] \in F$ or $[t, p] \in F$ ($p \in P, t \in T$) such that $p \not R t$, then $\{p\}$ is an equivalence class w.r.t. R.*

Proof. Assume that p is part of a nontrivial class. Then p is connected to other nodes only in the step of the construction of R. This means that the equivalence class of p contains both a pre-transition and a post-transition of p. By the inductive base, however, one of these transitions must be equivalent to t, so, by transitivity, p, and t would be equivalent, in contradiction to the initial assumption. □

In consequence, every class that contains transitions is surrounded by singleton place classes. Next we show that a singleton place class $\{p\}$ has at most one class from where tokens are produced to p, and at most one class from where tokens are consumed from p.

Lemma 4. *Let $\{p\}$ be an equivalence class w.r.t. R. Then all transitions in ${}^\bullet p$ are equivalent and all transitions in $p^\bullet$ are equivalent.*

Proof. This is evident from the construction of R_0. □

The above lemmas show that open nets can be constructed by letting a class that contains transition serve as the set of inner nodes while the adjacent singleton place classes are used as interface.

Definition 2 (Open net from equivalence class). *Consider an equivalence class E according to the above relation R that contains at least one transition. The corresponding open net $[P', T', F', I, O]$ is determined by $T' = E \cap T$, $P' = T'^\bullet \cup {}^\bullet T'$, $F' = F \cap (P' \cup T')$, $I = (P' \setminus E) \setminus T'^\bullet$, and $O = (P' \setminus E) \setminus {}^\bullet T'$.*

Lemma 5. *The just defined structure is indeed an open net.*

Proof. We have $|T'| > 0$ since E contains a transition. No node is isolated as only pre-and post-places of T' are considered, and each element of T' has at least one pre- or post-place. I and O are valid sets of input and output places as problematic transitions are removed. □

Lemma 6. *The set of constructed open nets are composable.*

Proof. This is an immediate consequence of Lemma 4. □

Now we are ready to prove the main result of this section.

Theorem 1. *Every Petri net has a unique finest decomposition into open nets.*

Proof. Consider relation R and the just constructed set of open nets. Each node occurs in a class. If the class contains a transition, then all its elements form the inner of some constructed open net. If the class does not contain transitions, it is a singleton place class. Since this place p is not isolated in N, it is connected to at least one class that contains a transition. So, p appears in the interface of one or two of the constructed open nets. Each arc of N appears in one of the constructed nets as well. The constructed set of open nets is composable. Consequently, the composition of this set exists and contains all nodes and arcs of N (and no other nodes or arcs). Hence, the set of open nets is a valid decomposition.

Consider any other decomposition and one of its components C. By Lemma 1, C contains, with every node, the whole equivalence class of that node. If it contains a class which contains transitions, it must also contain all adjacent singleton place classes since otherwise the arcs from or to the place in the singleton class cannot be represented in any composition involving C. Consequently, C is the union (in other words, the composition) of some open nets as constructed above. □

Corollary 1. *The finest decomposition is unique.*

Proof. The relation "finer as" can easily be identified as a partial order. Partial orders, however, cannot have more than one minimum. □

The main result can be interpreted as follows: Every decomposition can be obtained by first building the finest decomposition and then composing some disjoint subsets to larger open nets. The last result of this section considers the same idea the other way round:

Theorem 2. *Let $\mathcal{C}$ be the finest decomposition of N. Let $\mathcal{P}$ be a partition of $\mathcal{C}$. Then $\{\oplus P \mid P \in \mathcal{P}\}$ is a decomposition of N.*

Proof. This can be easily verified as composition is plain union of the ingredients, and composability is not affected by the constructions. □

The results of this section justify the name *atomic components* for the members of the finest decomposition.

4 Constructing a finest decomposition

The main task in constructing the finest decomposition is to compute the equivalence relation R. We represent R as the corresponding partition of $P \cup T$. Initially, the partition consists of singleton nodes of the given net only. We access the partition using two operations. $Find(x)$ takes a node as input and returns the (unique) class in the partition that contains x. $Union(C_1, C_2)$ takes two (not necessarily different) classes as argument and modifies the partition by replacing C_1 and C_2 by $C_1 \cup C_2$. The result is a (coarser) partition. Tarjan [7] showed that a partition can be organised such that an arbitrary sequence of n union and find operations can be executed in $O(nlog * n)$ time where $log*$ is an extremely slowly growing but asymptotically unbounded function. Our considerations result in the procedure sketched in Table 1.

Consequently, the finest decomposition can be computed in little more than linear time.

5 Potential applications

Divide-and-conquer techniques for structural analysis

Some traditional Petri net structures, in particular siphons (a set of places S where ${}^\bullet S \subseteq S^\bullet$) and traps (a set of places R where $R^\bullet \subseteq {}^\bullet R$) behave quite regularly w.r.t. the proposed way of decomposition. If S is a siphon in $N = [P, T, F]$ and $C = [P', T', F', \mathit{In}, \mathit{Out}]$ is a component of some decomposition, then $S \cap P'$ is a siphon in C. The other way round, if $N = N_1 \oplus N_2$, S_1 is a siphon in N_1, S_2 is a siphon in N_2 and both S_1 and S_2 contain the same interface places, then $S_1 \cup S_2$ is a siphon in N. Traps behave similarly. In future work, we aim at exploiting this observation for a divide-and-conquer algorithm for deciding Commoner's property (every siphon contains an initially marked trap) which allows nice implications to liveness of free-choice nets or deadlock-freedom of arbitrary Petri nets.

Extracting services from business process models

Open nets have been studied as models for web service behavior. Business processes can be modeled as Petri nets as well. A decomposition of a business process model into open nets thus yields a separation of the business process into web service. The finest decomposition is not necessarily a useful decomposition. However, building larger components by composing, for instance, components with identical role annotations may lead to the identification of useful portions of the net which could be separated as a web service.

Table 1. Construction of finest decomposition

Input: Net $[P, T, F]$
Output: Set of open nets M

Var U: SET of SET of Nodes

Init: $U = \{\{x\} \mid x \in P \cup T\}$

FOR ALL $p \in P$,$t_1, t_2 \in T$ DO
  IF $p \in t_1^\bullet \cap t_2^\bullet$ THEN $union(find(t_1), find(t_2))$;
  IF $P \in {}^\bullet t_1 \cap {}^\bullet t_2$ THEN $union(find(t_1), find(t_2))$;
END
REPEAT UNTIL nothing changes
  FOR ALL $p \in P$,$t_1, t_2 \in T, t_1 \neq t_2$ DO
    IF $find(t_1) = find(t_2)$ AND $p \in {}^\bullet t_1 \cup t_2^\bullet$ THEN $union(find(t_1, find(p)))$;
  END
END
$M := \emptyset$;
FOR ALL $X \in U$ DO
  IF $X \cap T \neq \emptyset$ THEN
    $TT := X \cap TT$;
    $PP := TT^\bullet \cup {}^\bullet TT$;
    $FF := F \cap (PP \cup TT) \times (PP \cup TT)$;
    $II := (PP \setminus X) \setminus TT^\bullet$;
    $OO := (PP \setminus X) \setminus {}^\bullet TT$;
    $M := M \cup \{[PP, TT, FF, II, OO]\}$;
  END
END

6 Examples

Applying our decomposition to academic examples leads to widely varying results. A particular version of the n dining philosophers net yields $2n$ open nets in the finest decomposition. Figure 1 shows the decomposition of the dining philosophers net for $n = 3$. Each open net consists of just two transitions (with a same label) and five interface places. Some standard net that grants concurrent read access and exclusive write access to some data base decomposes into no more than two open nets, regardless of the number of reading and writing processes. In realistic examples, our technique tends to decompose into many and small components, as the following numbers show. Using the implementation available online under `www.service-technology.org/diane`, we checked more than 1700 workflow nets provided by IBM Zurich. The nets had up to 273 places, 284 transitions, and 572 arcs. The resulting open net in the decomposition had at most 66 places, 12 transitions, and 66 arcs. In fact, many resulting open nets consisted of only one transition and a couple of interface places.

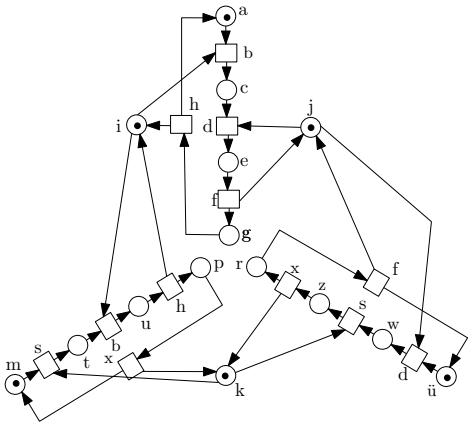

Fig. 1. Dining philosophers net and its decomposition

7 Related Work and Conclusion

Decomposition of Petri nets is a well-studied domain [2, 8]. [5] considers decomposition of nets where cut places and transitions are given and compositional verification of Commoner property for asymmetric choice nets. The idea to cut a net such that the border places form an asynchronous message passing interface, however, appears to be new. The closest approach seems to be the decomposition of a net into conflict clusters. In fact, every conflict cluster of a net is local to a dingle open net component (cf. the base in the inductive definition of R). In contrast to conflict clusters, our components are also closed under backward conflict clusters (see again the definition of R). Given these observations as well as the sketched potential applications, this kind of decomposition could eventually pay off.

References

1. P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional modeling of reactive systems using open nets. In *CONCUR*, volume 2154 of *LNCS*, pages 502–518. Springer, 2001.
2. G. Berthelot. Transformations and decompositions of nets. In *APN 1986*. Springer-Verlag, 1987.
3. F. Commoner. *Deadlocks in Petri Nets*. Applied Data Research, Inc., Wakefield, Massachusetts, Report CA-7206-2311, 1972.
4. M.H.T. Hack. Analysis of Production Schemata by Petri Nets. Master's thesis, MIT, Dept. Electrical Engineering,, Cambridge, Mass, 1972.
5. Li Jiao. Decomposition of nets and verification in terms of decomposition. In *CIMCA/IAWTIC*, pages 804–809. IEEE Computer Society, 2005.
6. E. Kindler, A. Martens, and W. Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *BPM*, volume 1806 of *LNCS*, pages 235–253. Springer, 2000.
7. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.
8. W. Vogler and B. Kangsah. Improved decomposition of signal transition graphs. *Fundam. Inform.*, 78(1):161–197, 2007.