Data under control

Niels Lohmann and Karsten Wolf

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{niels.lohmann, karsten.wolf}@uni-rostock.de

Abstract. Controllability is a fundamental sanity check for open systems such as services. Existing approaches to check controllability consider Petri net models in which data aspects are usually neglected or abstracted from. This paper investigates controllability of algebraic Petri nets and sketches a symbolic analysis approach.

1 Introduction

The paradigm of *service-oriented computing* advocates replacing large monolithic systems by a composition of loosely coupled components, called *services*. The behavior of services can be faithfully modeled with open nets [7], a class of Petri nets with dedicated interface places. Composition of services then boils down to gluing these interface places.

Controllability [12] is a fundamental sanity check for open nets. An open net N is controllable iff there exists another open net M (called *partner* of N) such that their composition $N \oplus M$ can always reach a distinguished final marking. Controllability of N can be verified constructively by *synthesizing* such a partner M. So far, partner synthesis has only been investigated for uncolored open nets. Apart from obvious advantages such as decidability and efficient implementations [6], this has the drawback that data can only be modeled to a very limited extend. In this paper, we study controllability of *algebraic open nets*; that is, an extension of open nets with an algebraic specification.

To reason about the behavior of an algebraic open net N, its algebraic specification needs to be *interpreted*. This results in a family of colored open nets that share the structure of N, but provide concrete semantics for the symbols used in the specification. Partner synthesis for colored open nets may unfortunately be undecidable [8], as infinite color domains may yield infinite state systems. Alternatively, *symbolic methods* treat the algebraic specification syntactically and replace interpreting by term rewriting and equivalence checks. Similar to parametrized reachability graphs for algebraic nets [4,11], we introduce a parametrized partner synthesis approach for algebraic open nets.

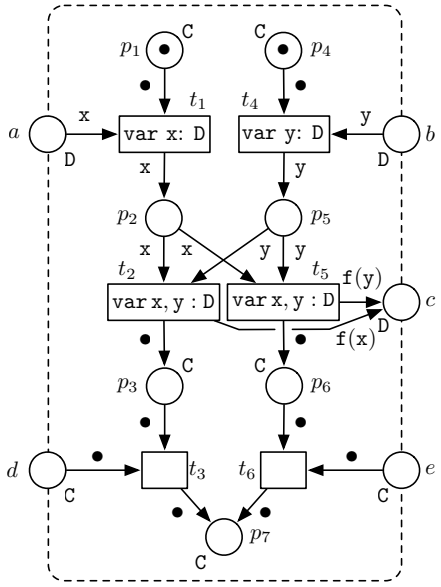
The rest of this paper is organized as follows. The next section introduces algebraic open nets and related concepts in terms of an example we use throughout this paper. Because of space constraints, we refrain from a formal definition and refer to standard literature for Petri nets and algebraic specifications [10]. Section 3 sketches partner synthesis for algebraic open nets and its application to the running example. Section 4 discusses related approaches. Section 5 concludes the paper and describes open issues with this approach.

```
SPECIFICATION S;

SORT D, C;

OPN f :  D → D;
    • :    → C;
```

(a) specification S



(b) algebraic open net N

Fig. 1. Running example

2 Algebraic open nets

An algebraic open net extends an open net with an algebraic specification, consisting of an signature (containing sorts and operations), variables, and a set of equations. For this paper, we consider the specification S, cf. Fig. 1(a). We distinguish a sort for data (D) and a sort for the control flow (C) of the service. For the latter sort, we are only interested in the constant symbol • which has the role of standard "black" tokens. Other values of C are not used. The unary function f is used to exemplify the treatment of terms. In this example, we do not employ a set of equations.

The specification then is "linked" to an open net by annotating places with sorts, transitions with variable declarations, and arcs with multiterms; Figure 1(b) depicts an example. This algebraic open net may concurrently receive two messages of sort D from the channels a and b, respectively. Then, $\mathtt{f}(v)$ is sent to channel c where v is nondeterministically chosen to be either the value that was received from channel a (t_2) or from channel b (t_5). Depending on this choice, a message of sort C is expected from either channel d or e. To reach the final marking $[p_7.\bullet]$, a partner service M apparently has to be able to deduce the choice of N from the value sent to channel c. This in turn depends on the operation f as well as the values that were initially sent from M to channel a and b. In the remainder of this paper, we show an algebraic partner can be synthesized together with a set of constraints that also need to be satisfied by any interpretation.

3 Symbolic partner synthesis

3.1 Partner synthesis in a nutshell

We begin with a description of the partner synthesis approach for open nets without algebraic specification [12]. This approach bases on the fact that the behavior of an open net can only be influenced indirectly by sending and receiving messages — the concrete marking of the open net cannot be observed. As a consequence, a partner can only make assumptions on the actual marking of the open net. These assumptions are formalized by a *set* of markings that cannot be distinguished by the partner.

The synthesis approach uses these sets as states of the partner and therefore synthesizes an automaton rather than a Petri net model. This automaton can be translated into a Petri net employing region theory [2] in a later step. Each of the partner's transition is labeled by a communication event: $!a$ and $?a$ for sending a message to and receiving a message from channel a, respectively. In a postprocessing step, states that contain markings from which no final marking can be reached are iteratively removed. Unless all states are removed, we can conclude controllability of the open net. The termination of this approach can be shown in case of bounded behavior of the open net and the restriction to a bounded number of messages that may be pending at all times. Under these assumptions, the number of marking set is finite. The interested reader is referred to [12] for a detailed description and formalization.

3.2 Symbolic synthesis

This paper aims at extending the synthesis toward algebraic open nets. Thereby, we follow a symbolic approach. This means that we refrain from enumerating the concrete data sets, but replace concrete data values (e. g., for messages) by variables and calculate with these variables. As a result, we need to cope with *parametrized* markings. This brings three challenges.

- First, the *equality* of markings and marking sets is more difficult to check. Rather than comparing multisets, we need to compare multiterms. Such equivalence checks may yield high worst-case complexity or may be even undecidable.
- Second, we need to be able to express *dependencies* between different states. A final marking may, for example, only be reachable for a certain constellation of data values. Such a constellation needs to be expressed in terms of constraints. In addition, states that violate these constraints need to be removed.
- Third, the *origin* of the data plays an important role. Whereas the environment has control over messages values that are sent to the service, it only has indirect influence on the values that are received. This limited scope of action needs to be taken into account to avoid spurious and unrealizable synthesized partners.

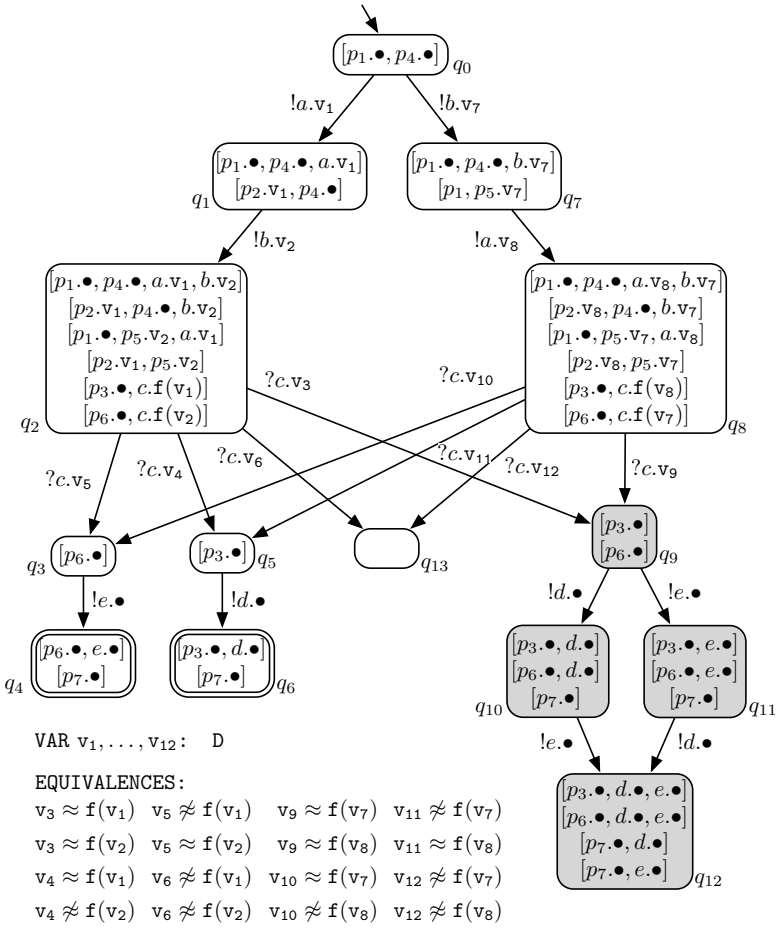


Fig. 2. Symbolic partner synthesis

To synthesize a partner for the open net N (cf. Fig. 1(b)) while taking the above mentioned challenges into account, we proceed as follows. We discuss the symbolic partner synthesis approach in terms of the example depicted in Fig. 2.

Variables. Whenever a data value is produced or consumed, we introduce a fresh variable (typed with the appropriate sort) and use it as placeholder in the marking sets. As an example, consider the transition labeled "$!a.\mathtt{v}_1$" reaching the state q_1 in which the value $\mathtt{v}_1$ is used in two markings. This value corresponds to the value that would be bound to variable `x` when firing transition t_1 in the algebraic open net N (cf. Fig. 1(b)). We see that after sending a message with value $\mathtt{v}_2$ to channel b, these values are also used as arguments for the function `f` in state q_2.

Equivalences. Consider the state q_2 that contains the markings $[p_3.\bullet, c.\mathtt{f}(\mathtt{v}_1)]$ and $[p_6.\bullet, c.\mathtt{f}(\mathtt{v}_2)]$. These markings model the outcome of the conflict between transition t_2 and t_5. This conflict is not visible to the environment of N. Conclusions about the current marking of N depend on assumptions on the relationship between the two values $\mathtt{f}(\mathtt{v}_1)$ and $\mathtt{f}(\mathtt{v}_2)$ on channel c. This yields four different successor states:

1. A value $\mathtt{v}_3$ is received which is equivalent to both values. Consequently, the environment cannot derive the outcome of the choice and in state q_3 it remains unknown whether p_3 or p_6 is marked. As each marking requires a different continuation (i. e., sending a message to channel c or d) the final marking cannot be safely reached. As a result, the gray shaded states are removed meaning that N cannot be controlled if $\mathtt{f}(\mathtt{v}_1)$ and $\mathtt{f}(\mathtt{v}_2)$ are equivalent.
2. A value $\mathtt{v}_4$ is received that is equivalent only to $\mathtt{f}(\mathtt{v}_1)$. In this case, state q_5 is reached and the environment can be sure that only p_3 is marked and sending a message to channel d reaches the final marking.
3. Analogous to the previous case for value v_5 which is equivalent to $\mathtt{f}(\mathtt{v}_2)$.
4. A value $\mathtt{v}_6$ is received that is neither equivalent to $\mathtt{f}(\mathtt{v}_1)$ nor to $\mathtt{f}(\mathtt{v}_2)$. This situation cannot occur and an empty state q_{13} is reached.

The resulting equivalences follow straightforwardly from a case distinction.

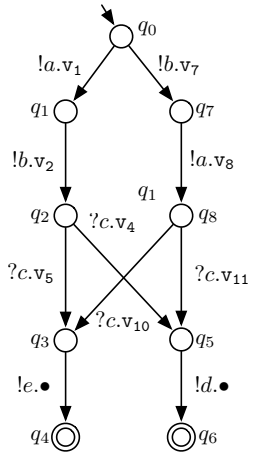
Constraints. After removing the gray states q_9–q_{12}, the final marking $[p_7.\bullet]$ is always reachable. Consequently, N is controllable and Fig. 3 depicts a partner. It is annotated with constraints that not only express the equivalences we derived earlier, but that also formulate the restrictions that follow from the origin of the message values: The values $\mathtt{v}_1$ and $\mathtt{v}_2$ can be freely chosen by the environment as long as it is guaranteed that they can be distinguished after applying function $\mathtt{f}$. This is formalized by according quantification of the variables.

To summarize, N is controllable for every interpretation of the specification $\mathtt{S}$ such that the constraints are satisfiable. An example would be interpreting $\mathtt{D}$ as the set of integers and $\mathtt{f}$ as the function $\mathtt{f}(x) = 2x$ for all $x \in \mathbb{Z}$. Then the constraints are satisfied by any distinct pair of integers as $2x \neq 2y$ for all $x \neq y$.

4 Related work

The introduction of fresh variable names is motivated by a compiler technique called *static single assignment form* [1]. Moser et al. [9] extended this technique to also cope with concurrency and derived data dependencies in WS-BPEL processes. This extension was later used to restructure WS-BPEL processes before translating them into uncolored open nets [3]. A similar approach is also followed by Lohmann et al. [5].

To the best of our knowledge, the presented partner synthesis in this paper is the only approach in which data is treated symbolically.



CONSTRAINTS:

$$\exists \mathtt{v}_1, \mathtt{v}_2 \in \mathtt{D} : \forall \mathtt{v}_4 \in \mathtt{D} : \big(\mathtt{v}_4 \approx \mathtt{f}(\mathtt{v}_1) \wedge \mathtt{v}_4 \not\approx \mathtt{f}(\mathtt{v}_2)\big)$$
$$\exists \mathtt{v}_1, \mathtt{v}_2 \in \mathtt{D} : \forall \mathtt{v}_5 \in \mathtt{D} : \big(\mathtt{v}_5 \not\approx \mathtt{f}(\mathtt{v}_1) \wedge \mathtt{v}_5 \approx \mathtt{f}(\mathtt{v}_2)\big)$$
$$\exists \mathtt{v}_7, \mathtt{v}_8 \in \mathtt{D} : \forall \mathtt{v}_{10} \in \mathtt{D} : \big(\mathtt{v}_{10} \approx \mathtt{f}(\mathtt{v}_7) \wedge \mathtt{v}_{10} \not\approx \mathtt{f}(\mathtt{v}_8)\big)$$
$$\exists \mathtt{v}_7, \mathtt{v}_8 \in \mathtt{D} : \forall \mathtt{v}_{11} \in \mathtt{D} : \big(\mathtt{v}_{11} \not\approx \mathtt{f}(\mathtt{v}_7) \wedge \mathtt{v}_{11} \approx \mathtt{f}(\mathtt{v}_8)\big)$$

Fig. 3. Symbolic partner for the open net N

5 Open issues and concluding remarks

This paper sketched a symbolic partner synthesis approach for algebraic open nets which extends the idea of parametrized reachability graphs. We showed the principal application in terms of a running example. Even this small example demonstrated some arising challenges that need to be studied in future work.

In Fig. 2, we can see that state q_2 and q_8 yield the same successor states. In fact, these two states only differ in the order in which the environment sent the messages a and b. Whereas it would be easy to proof equivalence of these states in this example, a generic method deserves further attention. In particular, the folding of cyclic behavior is a challenging tasks, because the equivalence of markings containing different terms needs to be shown. Existing results in the area of parametrized reachability graphs seem to be a promising starting point to tackle such problems.

Acknowledgments. The authors thank Christoph Wagner for inspiration on the running example.

References

1. Alpern, B., Wegman, M.N., Zadeck, F.K.: Detecting equality of variables in programs. In: POPL 1988. pp. 1–11 (1988)
2. Badouel, E., Darondeau, P.: Theory of regions. In: Advanced Course on Petri Nets. pp. 529–586. LNCS 1491, Springer (1996)
3. Heinze, T.S., Amme, W., Moser, S.: Process restructuring in the presence of message-dependent variables. In: ICSOC 2010 Workshops. pp. 121–132. LNCS 6568, Springer (2010)

4. Lindquist, M.: Parameterized reachability trees for predicate/transition nets. In: PETRI NETS. pp. 301–324. LNCS 674, Springer (1991)
5. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WS-BPEL processes using flexible model generation. Data Knowl. Eng. 64(1), 38–54 (2008)
6. Lohmann, N., Weinberg, D.: Wendy: A tool to synthesize partners for services. In: PETRI NETS 2010. pp. 297–307. LNCS 6128, Springer (2010)
7. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
8. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidablity of partner existence for open nets. Inf. Process. Lett. 108(6), 374–378 (2008)
9. Moser, S., Martens, A., Görlach, K., Amme, W., Godlinski, A.: Advanced verification of distributed WS-BPEL business processes incorporating CSSA-based data flow analysis. In: IEEE SCC 2007. pp. 98–105. IEEE Computer Society (2007)
10. Reisig, W.: Petri nets and algebraic specifications. Theor. Comput. Sci. 80(1), 1–34 (1991)
11. Schmidt, K.: Parameterized reachability trees for algebraic Petri nets. In: PETRI NETS. pp. 392–411. LNCS 935, Springer (1995)
12. Wolf, K.: Does my service have partners? LNCS T. Petri Nets and Other Models of Concurrency 5460(2), 152–171 (2009)