Partner Synthesis for Data-Dependent Services

Christoph Wagner

Institut für Informatik, Humboldt Universität zu Berlin,
Unter den Linden 6, 10099 Berlin, Germany
cwagner@informatik.hu-berlin.de

Abstract. A service is *controllable*, if there exists a service with which it can interact properly. We sketch an approach to decide controllability for a certain class of services. Controllability is decided by synthesizing a service that controls the given service. For a class of services which abstracts from data, the synthesis problem is already solved. In this paper, we present an approach for a class of services that deals with data explicitly.

1 Introduction

A service is designed with the goal that it can interact with another service. A service may interact properly with one service but not interact properly with an other service. For example, two services may end up blocking each other, making any further interaction impossible. The possibility of the occurrence of an error depends on both interacting services and in general can not be attributed to one service alone. However, at the time of design of a service, the other services the service will be interacting with typically are not known in advance. Nevertheless, we can check a fundamental property called *controllability* [7] when considering one service in isolation. A service is called *controllable*, if there is a least one other service it can interact with properly. Controllability can be decided by *synthesizing* a service that interacts with the given service properly.

The synthesis problem is solved for the finite automaton based service model used in [3,7]. This service model abstracts from data, i. e. messages are not distinguished by their content. For a more realistic model which takes data into account, the synthesis problem is still open. The goal of our work is to solve the synthesis problem for a service model that includes data.

In Sect. 2, we introduce the concept of a *partner* of a service. Section 3 presents an approach to synthesize a partner for a High-Level Petri net based service model. Section 4 concludes our work.

2 Partners

We represent a service by an *open net* [3] (Fig. 1). An open net is a Petri net with distinguished *interface places* that represent channels for asynchronous message exchange. We use High-Level Petri nets [2] so that data can be represented by coloured *tokens*.

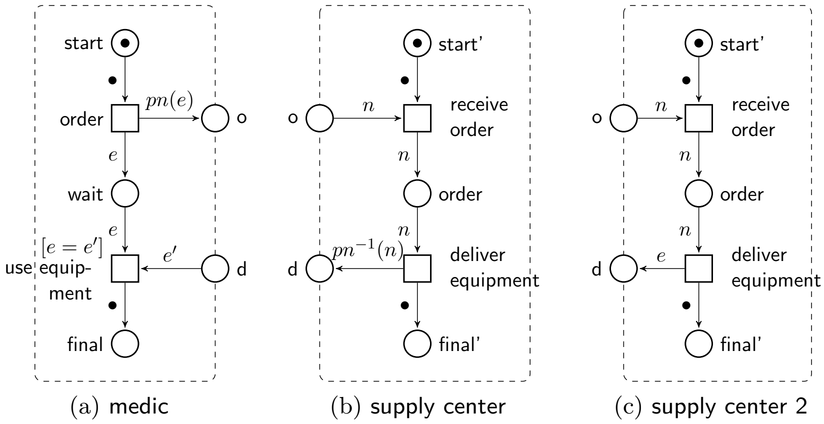


Fig. 1: Some open nets

The following example illustrates the interaction of a medic with a medical supply center from which the medic orders medical equipment. Both the medic and the supply center are services represented by the open nets in Fig. 1a and Fig. 1b. The composition (medic$\oplus$supply center) of the open nets medic and supply center is the Petri net we obtain by fusing the interface places o and d. The medic orders medical equipment by *firing* the transition order. Thereby the token • is removed from place start. The variable e is assigned the equipment the medic orders, e. g., a syringe. Therefore, a token with value syringe is produced on place wait. The term $pn(e)$ evaluates to the product number of the syringe. A token with that value is produced on the output place o. This corresponds to sending a message to supply center.

The medic waits for an incoming delivery on place d. supply center receives the order message from o by firing receive order and n is assigned the product number of syringe. The transition deliver equipment takes the product number from place order and produces the equipment corresponding to that number (in this case syringe) on interface place d. Since the tokens on wait and d are equal, the *guard* $e = e'$ of transition use equipment is fulfilled. Therefore use equipment can fire and (medic$\oplus$supply center) reaches its *final marking*, i. e. the marking in which both places final and final' are marked with the token •.

Two open nets N_1 and N_2 are called *partners*, if from each reachable marking of $N_1 \oplus N_2$ a final marking of $N_1 \oplus N_2$ is reachable. The medic and supply center are partners. The medic and supply center 2 are not partners: When deliver equipment fires, any arbitrary equipment may be assigned to e which in general does not correspond to the product number assigned to n. Therefore, there is a marking reachable in (medic$\oplus$supply center) with a token with value syringe on wait and a token with another value (e. g. stethoscope) on d. This marking is a *deadlock*, because $e = e'$ is not satisfied and use equipment can not fire. On acyclic open nets, the reachability of a final marking is equivalent to deadlock freedom.

An open net N_1 is called *controllable*, if it has a least one partner. medic is controllable. In this example, we assumed that function pn has an inverse pn^{-1}. However, if we replace pn by a function pn' which is not bijective, medic becomes uncontrollable. In that case, a supply center can not infer the equipment the medic is waiting for from the product number. Therefore, it is not possible to guarantee that $e = e'$ is always satisfied and the final marking will be reached.

In the next section, we sketch an approach to synthesize a partner for a specific class of open nets.

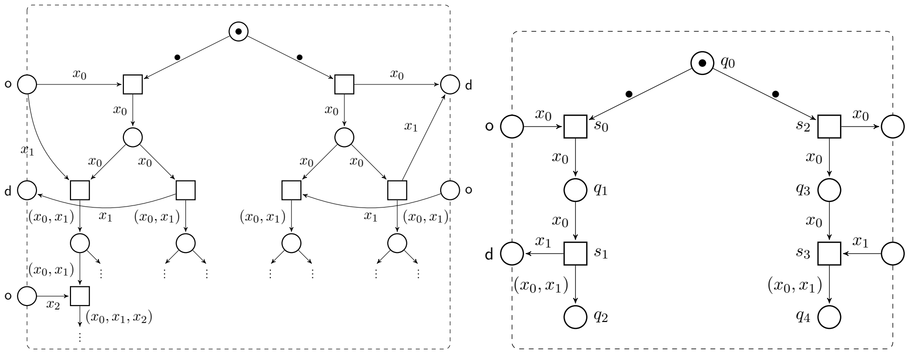
3 Partner Synthesis

In this section, we sketch a partner synthesis algorithm for a given open net N. We consider a class of Petri nets where the domain of the variables and the colours of the tokens is infinitely large. The guards are denoted in a subset of first order logic that contains boolean algebra and quantifiers. For computational reasons, we assume that this subset is decidable (like e. g. Presbuger arithmetic [5]). We also assume that N is acyclic and the number of tokens on each place is at most one. Therefore, the length of each path in the state space of N is bounded by some number k.

Due to infinitely many colours, the state space of N is infinitely large. Therefore the synthesis algorithm for finite state services given in [7] can not be applied to our service model. Nevertheless, conceptually, our synthesis algorithm follows a similar approach. Therefore, we briefly outline the approach used for finite state services: First, an *over-approximation* of the partner of N that will be synthesized, i. e., a service that is guaranteed to contain a partner as a sub-graph, is generated. Then certain states are removed from the over-approximation iteratively. The iteration is repeated as long as the composition of the given service and the over-approximation contains a deadlock. Eventually, two cases may occur: 1. The composition is deadlock-free. Then the remaining sub-graph of the over-approximation is a partner. 2. Every state has been removed. Then N is not controllable.

Now we give an overview of our synthesis algorithm. The details will be explained later by example. First, we construct an over-approximation S_0 of the partners of N. S_0 is a prefix of depth k of an infinite tree-like open net U we call *universal environment*. Then we iteratively add guards to S_0. Adding a guard corresponds to removing states from the over-approximation in the finite case. The iteration is repeated until no deadlock is reachable in the composition of N and S_0. Each time a guard is added, some of the deadlocks of the composition become unreachable. Each iteration step may also introduce new deadlocks which will then be eliminated in the next iteration step. If (and only if) N is controllable, the composition will eventually be deadlock free and the modified S_0 is a partner.

Now we show the derivation of a partner of medic from Fig. 1a. We assume that the colours of medic are integers and pn is the bijective function with $pn(x) = x + 1$. Therefore, medic can be expressed with Presburger arithmetic.



(a) Universal environment U (b) Finite prefix S_0 of U

Fig. 2: Universal environment and its prefix. Interface places o and d are depicted multiple times to improve readability.

Fig. 2a shows the universal environment of medic. U is an infinite open net that has the inverse interface of N. U has a regular tree-like structure and can send and receive any (possibly infinite) sequence of messages. Therefore, U is an over-approximation of every partner of N. U stores every message sent or received from N. Each of the variables $x_0, x_1, \ldots$ corresponds to the value of a message. These variables will be used in the guards which we will add later on. Since U is infinitely large, it is not suited for computational methods. Due to the acyclicity of N, the number of messages N can send and receive is bounded by a number k. Therefore, the prefix of U of depth k is still an over-approximation of the partners of N.

In the example, k is 2. In this particular case, we can even remove the branches of U which send or receive two messages on the same interface place without destroying the over-approximation property. This is possible because medic sends or receives at most one message on each interface place. Thus, we get the prefix S_0 of U in Fig. 2b.

Now we iteratively derive a guard for each transition of S_0. The transitions are processed in bottom-up order. Thereby, we obtain a sequence $S_0, S_1, S_2, \ldots$ of open nets with successively smaller reachability graphs.

Intuitively, a guard forbids a transition s of S_0 to fire in a certain firing mode if there is the possibility to reach a deadlock after s has fired in that mode. That way, deadlocks are successively removed from the composition.

Since a transition has infinitely many firing modes, we need a syntactical representation of all firing modes that may not lead to a deadlock. We derive the guard predicate that is assigned to each transition using a technique outlined in [6]. The technique is based on the *symbolic reachability graph* (SRG) of a High-Level Petri net. The symbolic reachability graph is a compact representation of the reachability graph that allows to represent a possibly infinite set of markings by a *symbolic marking*. Fig. 3 shows the SRG of the composition medic $\oplus$ S_0. In a symbolic marking M, every value is represented by a term. Attached to M is a

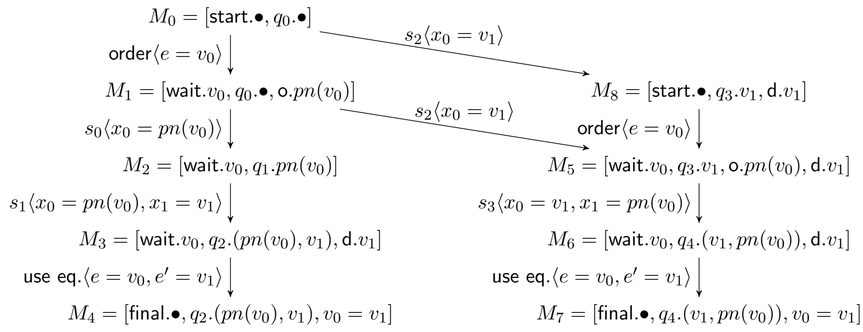


Fig. 3: Symbolic reachability graph of medic $\oplus S_0$

condition $COND(M)$ which restricts the set of valid assignments to the variables that occur in M. A marking m is reachable if and only if there is a symbolic marking M that evaluates to m for an assignment that satisfies $COND(M)$. Technically, during the construction of the SRG, $COND(M)$ is formed by the conjunction of the effects of every guard on a path to M. Each edge of the SRG is inscribed by a transition t and a *symbolic firing mode*. A symbolic firing mode of t assigns a term to each variable of t.

In our example, the integer that is chosen non-deterministically by order for e is represented by the variable v_0 in marking M_1. Analogously, the integer sent by s_1 on d is represented by v_1. The effect of the guard $e = e'$ of use equipment is represented by the condition $v_0 = v_1$ of the markings M_4 and M_7. Every other symbolic marking has the condition *true*.

The method we use to derive the guard of a transition s of S_0 is inspired by Dijkstra's predicate transformer semantics [1]. The guard predicate can be regarded as the *weakest pre-condition* so that after firing of the transition s a specific post-condition holds. Here, the post-condition describes the assignments for which no subsequent symbolic marking evaluates to a deadlock. For each symbolic marking M we define a predicate $DF(M)$ that describes for which assignments of the variables $v_0, v_1, \ldots$ of the SRG M does not evaluate to a deadlock. $DF(M)$ is formed by the disjunction of the conditions of the successors of M. The SRG and the DF predicates are recalculated in each iteration step. We denote the iteration step by a subscript.

M_4 and M_7 are final markings. Therefore, $DF_0(M_4) \equiv DF_0(M_7) \equiv$ *true*. Since M_3 has a successor marking only for those assignments of v_0 and v_1 with $v_0 = v_1$, we get $DF_0(M_3) \equiv v_0 = v_1$. Analogously, $DF_0(M_6) \equiv v_0 = v_1$. For every other symbolic marking M we get $DF_0(M) \equiv$ *true*.

From every predicate $DF(M)$, we derive a predicate $DF'(M)$ which expresses the condition that M does not evaluate to a deadlock in terms of the variables used by the transition s of S_0 that precedes M in the SRG (s is unique because S_0 is a tree). By assigning $DF(M)$ as a guard to s, every evaluation of M which is a deadlock becomes unreachable. The relationship between the variables $x_0, x_1, \ldots$

of S_0 and the variables $v_0, v_1, \ldots$ of the SRG is established by the the symbolic firing mode of s.

In the example, M_3 is reachable via exactly one path of the SRG. The last transition of S_0 on this path is s_1 with the symbolic firing mode $\langle x_0 = pn(v_0), x_1 = v_1\rangle$. As stated above, $DF_0(M_3) \equiv v_0 = v_1$. With $x_0 = pn(v_0)$, $x_1 = v_1$ and the assumption that pn is injective follows that $v_0 = v_1$ is equivalent to $pn^{-1}(x_0) = x_1$. Formally, we express this transformation by universal quantification of v_0, v_1:

$$\begin{aligned} DF'_0(M_3) &\equiv \forall v_0, v_1 : x_0 = pn(v_0) \wedge x_1 = v_1 \implies v_0 = v_1 \\ &\equiv x_1 = pn^{-1}(x_0) \end{aligned}$$

In words: $DF'_0(M_3)$ describes all assignments of x_0, x_1 for which the post-condition $v_0 = v_1$ is guaranteed to hold after the firing of s_1 in mode $\langle x_0 = pn(v_0), x_1 = v_1\rangle$, regardless of which integers might have been non-deterministically chosen for v_0, v_1.

The general form of a DF' predicate (for the special case that there is only one path in the SRG to M) is

$$\forall v_0, \ldots, v_j : COND(M) \wedge x_0 = T_0 \wedge \ldots x_n = T_n \implies DF(M)$$

where $\langle x_0 = T_0, \ldots, x_n = T_n\rangle$ is the symbolic firing mode of the last transition s of S_0 on the path to M.

For M_4, we obtain $DF'(M_4) \equiv true$. We add the conjunction $DF'(M_3) \wedge DF'(M_4)$ as a guard to the transition s_1 that precedes both M_3 and M_4 in the SRG. After adding this guard $x_1 = pn^{-1}(x_0)$ to s_1, every deadlock in which q_2 is marked becomes unreachable.

We repeat this procedure for s_3. M_6 is reachable via two paths of the SRG (which differ only insignificantly). The last transition of S_0 on both paths is s_3 with firing mode $\langle x_0 = v_1, x_1 = pn(v_0)\rangle$. Analogously, we get

$$\begin{aligned} DF'_0(M_6) &\equiv \forall v_0, v_1 : x_0 = v_1 \wedge x_1 = pn(v_0) \implies v_0 = v_1 \\ &\equiv x_1 = pn(x_0) \end{aligned}$$

In a more general case, we may get a different predicate for each path on which a marking M is reachable. Then $DF'(M)$ is the conjunction of these predicates. With $DF'_0(M_7) \equiv true$ we get the conjunction $DF'_0(M_6) \wedge DF'_0(M_7) \equiv x_1 = pn(x_0)$. After assigning $x_1 = pn(x_0)$ to s_3 as a guard, no deadlock is reachable in which q_4 is marked.

Let S_1 be the open net derived from S_0 by adding the two guards to s_1 and s_3. These guards introduce new deadlocks in which q_2 and q_4 are not marked, e. g. $[\mathsf{wait}.0, q_3.3, \mathsf{o}.1, \mathsf{d}.3]$ is a deadlock in $\mathsf{medic} \oplus S_1$ but not a deadlock of $\mathsf{medic} \oplus S_0$. These new deadlocks will become unreachable in the next iteration step. In the SRG of $\mathsf{medic} \oplus S_1$ (Fig. 4), M_3 has the condition $v_0 = v_1$ and M_6 has the condition $pn(v_0) = pn(v_1)$ due to the guards that were added. Therefore we obtain the predicates

$$\begin{aligned} DF'_1(M_2) &\equiv \forall v_0 : x_0 = pn(v_0) \implies \exists v_1 : v_0 = v_1 \equiv true \\ DF'_1(M_5) &\equiv \forall v_0, v_1 : x_0 = v_1 \implies pn(v_0) = pn(v_1) \equiv false \end{aligned}$$

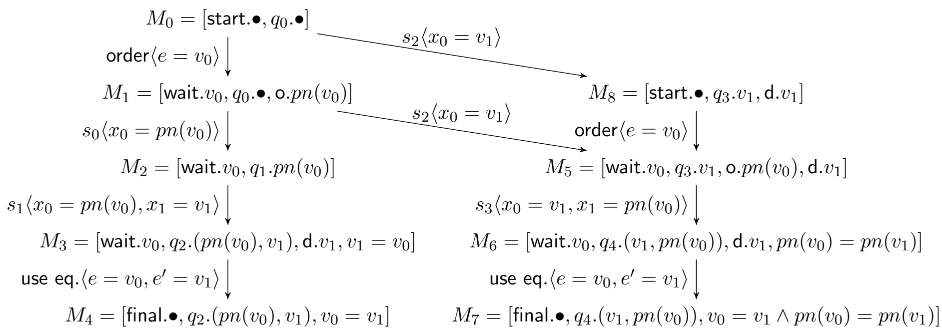
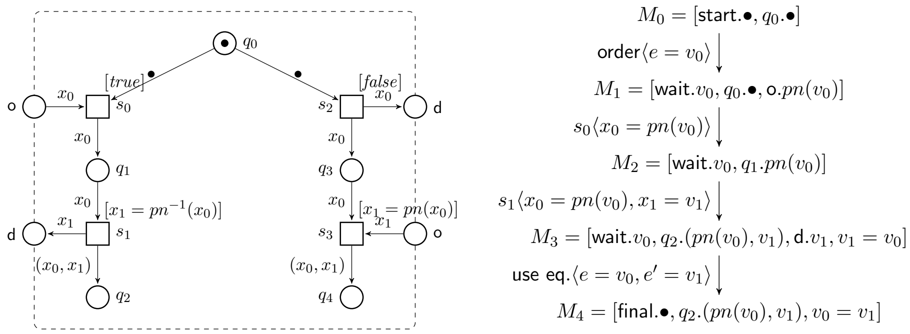


Fig. 4: Symbolic reachability graph of $\mathsf{medic} \oplus S_1$.

Variable v_1 is existentially quantified, because v_1 is not yet defined in M_2 but will be created and chosen appropriately by s_1 in the step from M_2 to M_3. Please note that existential quantifiers may only appear as a part of a DF' predicate.



(a) S_2 (b) SRG of $\mathsf{medic} \oplus S_2$

Fig. 5: Last iteration step of the partner synthesis

Eventually, by assigning the predicate $DF'_1(M_2) \equiv \mathit{true}$ to s_0 and $DF'_1(M_5) \equiv \mathit{false}$ to s_2, we obtain the open net S_2 shown in Fig. 5a. By repeating our calculation on the SRG of $\mathsf{medic} \oplus S_2$ (Fig.5b), we obtain $DF_2(M_0) \equiv \mathit{true}$. This indicates that no reachable marking in which q_0 is marked is a deadlock. Therefore, every deadlock has been eliminated from $\mathsf{medic} \oplus S_2$ and S_2 is a partner of medic by definition. Please note that S_2 is very similar to N_2 from Fig. 1b. In general, the last open net S_i of the sequence is a partner of N iff for every symbolic marking M in which the root place q_0 is marked the predicate $DF_i(M)$ is fulfilled for every assignment that fulfils $COND_i(M)$.

4 Conclusion

We sketched an algorithm to synthesize a partner of a service represented by a High-Level Petri net. Currently, our approach is limited to acyclic services. We show a systematic approach to derive the relations between the values of incoming and outgoing messages that a service has to adhere to in order to be a partner of the given service. Since relations are denoted in first-order logic which is undecidable in the general case, we rely on an oracle to decide controllability. For a decidable theory like presburger arithmetic [5], controllability can be effectively computed. The technical details of the approach are not yet fully worked out.

Lohmann et al. [4] sketch a different approach to synthesize a partner for a High-Level Petri net based service. They use a symbolic representation for markings similar to ours. Their work focuses on the construction of the structure of the partner and only briefly discusses the derivation of the predicates. They give an ad-hoc explanation of the construction of the predicates for a particular example but do not describe a general derivation method. In particular, values that are non-deterministically chosen by the service are not treated.

In contrast to their approach, our approach does not consider structural aspects of the partner synthesis at all. All information concerning the behaviour of the partner is encoded by the guards. The structure of the partner is chosen in a generic way. However, our approach can handle values that are non-deterministically chosen by the service.

In our future work we aim at extending our approach to cyclic services. In this scenario it may be advantageous to chose a specific structure for the synthesized partner. E. g. case distinctions for specific values should result in a distinct node for each case. With a structure reflecting the behaviour of the service, it will be easier to identify isomorphic branches of the structure and nodes that can be combined without changing the behaviour. In order to ensure termination of the synthesis algorithm in the cyclic case, it is necessary to guarantee that there will be only finitely many nodes after the nodes have been combined.

References

1. Dijkstra, E.W.: A Discipline of Programming. Prentice Hall, Inc. (October 1976)
2. Jensen, K., Kristensen, L.M.: Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer (2009)
3. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: ICATPN 2007, Siedlce, Poland, June 25-29, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4546, pp. 321–341. Springer-Verlag (2007)
4. Lohmann, N., Wolf, K.: Data under control. In: AWPN 2011 (Sep 2011)
5. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du Premier Congrès des Mathématicienes des Pays Slaves. Warsaw (1929)
6. Wagner, C.: A data-centric approach to deadlock elimination in business processes. In: ZEUS 2011, Karlsruhe, Germany. CEUR-WS.org (2011)
7. Wolf, K.: Does my service have partners? LNCS ToPNoC 5460(II), 152–171 (Mar 2009), special Issue on Concurrency in Process-Aware Information Systems