MODELING AND CONTROLLER SYNTHESIS FOR RESOURCE BOOKING PROBLEMS USING BDDS

Petter Falkman, **Arash Vahidi** and **Bengt Lennartson**

Control and Automation Laboratory, Department of Signals and Systems, Chalmers University of Technology, Skeppsgränd 2, SE-412 96 Göteborg, Sweden

Abstract: Mixed Process algebra and Petri Nets (MPPN) combine Petri net constructs and Process algebra. The modeling language implies a compact representation of complex systems. For general resource booking problems MPPN is used for routing specifications and modeling of resources. These models are formally converted into ordinary Petri nets. The Petri net representations are easily converted into efficient Binary Decision Diagram (BDD). The BDDs are then used used for controller synthesis. It is shown that in specific cases very few boolean variables are required in the BDD representation based on MPPN. This is crucial in order to calculate a controller efficiently.

Keywords: Discrete event systems, Petri nets, process algebra, binary decision diagrams, controller synthesis.

1. INTRODUCTION

In this paper we focus on concurrent systems that may be modeled as discrete event systems (DES), and especially on routing and resource booking problems. Such systems may be described as a set of shared resources and a set of objects. The objects are described by the routing specification which involves a set of operations that are to be executed in a certain order involving certain resources. The objects may be products in a flexible production system, data packets in a communication network or vehicles in a traffic control system. In order to synchronize the objects utilization of the available shared resources, a *controller* is often required.

Mixed Process algebra and Petri Nets (MPPNs), is as a specification and modeling language, which combine both Petri net constructs and Process algebra. Process operators for alternative, synchronization and arbitrary order are defined in order to realize a compact representation. Additional operators for join and conditional event are defined in order to formally convert the MPPN into ordinary Petri nets models, which are easily represented as Binary Decision Diagrams (BDDs).

The huge number of states in a resource booking system normally give vast computations in formal verification and controller synthesis. For this purpose symbolic tools such as BDDs have been used for efficient computations,(Bryant, 1992). For supervisory control (Ramadge and Wonham, 1987) BDDs have also been used by e.g. Gunnarson (1997).

The transformation of both Process algebra and Petri nets, respectively, into symbolic representations has been investigated by several researchers, e.g. (Corno *et al.*, 1995; Camurati *et al.*, 1993; Bloom *et al.*, 1997). To successfully use BDDs it is important to optimize the BDD tree. Apart from *variable ordering*, another important issue is the *number* of boolean variables, (Bloom *et al.*, 1997).

The purpose of this paper is to introduce operators in order to formulate the conversion between a high level representation into ordinary Petri nets. It is also shown that an efficient BDD representation is created due to a compact resource representations.

2. MIXED PROCESS ALGEBRA AND PETRI NETS (MPPN)

In order to achieve a powerful tool for routing and booking/unbooking specifications, process operators for standard alternative, synchronization and arbitrary order have been introduced in Falkman and Lennartson (2001). Operators for join and conditional event is defined which makes it possible to convert the MPPN models into ordinary PNs in a formalized manner. These are then easily converted into BDD representations.

Petri Nets and processes The transition between two Petri net places in an MPPN is a process P. A process $P = a_1 \to P_1$ describes that, first the event a_1 occurs, then it behaves like a process P_1. In our case a process is ended after a sequence of events $P = a_1 \to a_2 \to \ldots \to a_n$, see Figure 1.

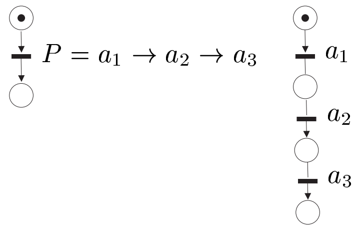

Fig. 1. In the model to the left there is a process P at the transition between two Petri net places. Converted into an ordinary Petri net it creates a new place for each event.

This implies that process P has completed its execution when the last event a_n in the sequence has occurred.

Alternative The alternative (split) operator $+$ specifies that there is a choice between two processes. Let two processes be defined as $P = a_1 \to P_1$ and $Q = b_1 \to Q_1$. Then an alternative between these two processes is described as

$$P + Q = a_1 \to P_1 \mid b_1 \to Q_1$$

using Hoare's (Hoare, 1985) choice symbol $|$. This implies that either event a_1 occurs followed by process P_1 or event b_1 occurs followed by process Q_1.Note that the alternative operator $+$ implies that one explicit place for each alternative event is created when converting the MPPN models into ordinary PNs.

Conditional event The conditional event a^b describes that event a can only occur if event b has occurred in an earlier transition. When a specific event has been conditioned it must occur again before the the condition is fulfilled. This can be exemplified as

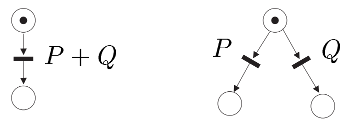

Fig. 2. In the model to the left there is an alternative between two processes P and Q. Converted into an ordinary Petri net it splits the Petri net into two sequences.

$$a \to b^a \to c^a + d \quad \Leftrightarrow \quad a \to b \to d$$

Join operator The join operator $\oplus$ specifies that there have been an alternative operation in an earlier transition and there is one or more parallel sequences that are to be joined into a single sequence.

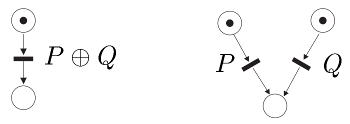

Fig. 3. In the model to the left there is two alternative processes P and Q. Converted into an ordinary Petri net it joins the Petri net into two sequences based on earlier alternative processes.

Synchronization The nonstandard synchronization operator &, which was suggested in Lennartson *et al.* (1998), implies that processes are to be synchronized without introducing common events; compare with Full Synchronous Composition (FSC) (Hoare, 1985). Similar ideas for event synchronization can be found in Fabian (1995) and in Arnold (1994). Again, consider two processes $P = a_1 \to P_1$ and $Q = b_1 \to Q_1$. The synchronization operator & can be described as

$$P\&Q = a_1\&b_1 \to P_1\&Q_1$$

This means that a_1 in P occurs at the same time as b_1 in Q. This synchronized event is denoted $a_1\&b_1$. For a more detailed description, (Falkman and Lennartson, 2001).

The synchronization operator & is useful when *flexibility* and *reuseability* are desired. Consider the following example: the issue is to synchronize the processes P and Q at one moment and later on P with another process $R = c_1 \to R_1$. To specify these synchronizations by FSC it is required to introduce specific event labels for these two different situations in the corresponding models. By the suggested synchronization operator & a specification process S including $a_1\&b_1$, later followed by $a_1\&c_1$, implies that the processes P, Q and R do not need to be modified (relabeled).

Arbitrary order Arbitrary order describes that all processes involved are to be executed, but the order does not matter. We define an arbitrary order operator $\ominus$ for two processes as

$$P \ominus Q = P \rightarrow Q + Q \rightarrow P$$

and more generally expressed $\ominus_{i=1}^{n} P_i = \mathrm{perm}(P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n)$ with perm denoting the sum of all different permutations of the sequence $(P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n)$. This means that $\mathrm{perm}(P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n)$ describes n! sequences, with an alternative of which sequence that will be executed.

Precedence The process operators above are executed with the following precedence, increasing from left to right

$$\ominus \ \preceq \ +, \ \oplus \ \preceq \rightarrow \preceq \ \&$$

This implies e.g. that

$$P \ + \ Q\&R = P \ + (Q\&R)$$

i.e. either process P occurs or processes Q and R occurs synchronized.

Example – Converting MPPNs into ordinary PNs
To the left in Figure 4 there is an MPPN involving two transitions. In the first transition there is a choice between two events a and b and the next transition involves a choice between two conditional events $c(a)$ and $d(b)$, describing that in order for the event c to occur a has to have occurred before.

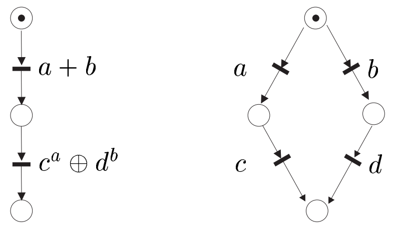

Fig. 4. To the left an MPPN model, and to the right the corresponding ordinary Petri net model with explicit places for the alternative.

□

3. ROUTING SPECIFICATIONS AND RESOURCES

In this section a formal description of the different building blocks in a resource booking system is presented. These building blocks are a set of resource models, a set of routing specifications and a controller, which synchronizes the individual objects utilization of the shared resources.

Routing specification The routing specification S_i describes which operations a particular object is to undergo and the sequence these operations are to execute. For each operation it is specified what resource or which resources that are required. The routing specification also describes whether there are alternative resources. The routing specification may be described on two levels:

- a *high level routing specification* (HRS) that describes which operations an object are to undergo, in which order these operations are to be executed, and which resource(s) that may be used for each individual operation.
- a *booking and unbooking specification*, which describes on a more detailed level how the shared resources are to be booked and unbooked, based on the HRS, to obtain the desired route through the resource system.

Resource model The resources may be modeled in several different ways. Here they are modeled with only two events; the booking event and the unbooking event using parameterized events which were introduced in Lennartson *et al.* (1998). The free parameter x in the general resource model to the left in Figure 5 is replaced by the name of the corresponding routing specifications S_i in the evaluated to the right in Figure 5.

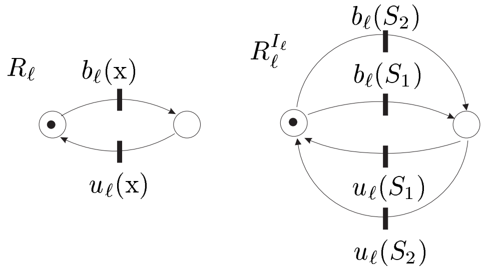

Fig. 5. To the left a general resource model with parameterized events, and to the right the corresponding evaluated resource model with index set $I_\ell = \{1, 2\}$.

The resource model to the right in Figure 5 is specified to be used by the routing specifications S_i, where i ϵ I_ℓ. I_ℓ is the index set corresponding to the routing specifications that utilize the resource R_ℓ. The evaluated resource model is then denoted $R_\ell^{I_\ell}$. To the right in Figure 5 the index set is $I_\ell = \{1, 2\}$.

Controller synthesis The purpose of the controller is to synchronize the objects utilization of the common, available resources. Unnecessary restrictions must be avoided, and as much flexibility as possible be given to the system, without danger of running into blocking states or other forbidden configurations.

As a first step to obtain a controller, the parameterized resource models are transformed based on the current routing specifications. The transformed resource models are synchronized with the routing specifications. Assume that m specifications $S_1, \ldots, S_m$ are given which altogether use n resources $R_1, \ldots, R_n$. Then the model

$$S = S_1 || \ldots || S_m$$

is a first *specification* of the desired behavior of the plant

$$R = R_1^{I_1} || \ldots || R_n^{I_n}$$

Here $||$ is CSP's full synchronous composition (FSC) (Hoare, 1985). Also note that the specifications S_i run independently of each other, since our modeling approach implies that the specification alphabets are disjunct. The two models R and S give together what is called the *global specification*

$$S_{sp} = R || S \qquad (1)$$

This is a first candidate for a possible controller C, and in fact it is a model of the controlled closed loop system. This model may however be blocking or noncomplete, (Ramadge and Wonham, 1987). The first aspect has to do with liveness and the second one is related to uncontrollable events, which we do not consider in this paper.

The global specification S_{sp} therefore has to be manipulated to result in an appropriate controller. This is formally expressed by the operator $\mathcal{N}B$, which removes blocking states from S_{sp} to make it nonblocking. The synthesized controller is now expressed as

$$C = \mathcal{N}B(S_{sp}) = \mathcal{N}B(R_1^{I_1} || \ldots || R_n^{I_n} || S_1 || \ldots || S_m) \quad (2)$$

4. BINARY DECISION DIAGRAMS

Symbolic model checkers based on Binary Decision Diagrams (BDDs) have been used to automatically verify properties of systems with as many as 10^{120} states (Burch *et al.*, 1991).

BDDs are a very common data structure for representing sets and functions/relations on finite domains. A BDD is an acyclic IF-THEN-ELSE digraph on a set of Boolean variables with two terminal nodes zero and one.

4.1 *BDDs for controller synthesis*

Based on the resource booking problem, with the global specification S_{sp} 1, the boolean state vector s is composed of many partial boolean state vectors:

$$s = \langle s_0, \ldots, s_n, r_0, \ldots, r_m \rangle \qquad (3)$$

where s_j is the state vector for the sub-model representing routing specification j, and r_k is the state vector for resource k (Vahidi *et al.*, 2001).

The model used in this paper is a simplified Boolean function model $\langle I, T \rangle$, where $I(s)$ is true iff s is an initial state. $T(s, s^{'})$ is true iff there exists one or more transitions from s to $s^{'}$.

The initial states $I(s)$

$$I(s) = \langle I(s_0), \ldots, I(s_n), I(r_0), \ldots, I(r_m) \rangle$$

implies that all objects are ready to start and that all resources are free and may be booked. The marked states $M(s)$

$$M(s) = \langle M(s_0), \ldots, M(s_n), M(r_0), \ldots, M(r_m) \rangle$$

assumes that all objects are finished and the resources are back to there initial state. The marked states may be seen as *final* or *desired states.*

5. CONVERTING MPPNs INTO BDDs

In order to convert the MPPN models into relations and BDDs it is important to keep the number of required boolean state variables to a minimum, to ensure a efficient BDD representation (Bloom *et al.*, 1997). This implies that it is better to have few large models with large number of states then many small models. For instance, a small model involving five states has to be modeled by three boolean variables. On the other hand a model involving 127 states only requires six variables. This means that only two small models with a total of states states requires the same number of variables as one large model with 127 states.

In the resource booking system there may be alternative shared resources that may be booked by a number of routing specifications. It is then necessary to know which resource that was booked by which routing specification. Alternative, may be specified either in the resource model or in the routing specification.

If many resources are shared by a smaller number of routing specifications it may be advantageous to specify, alternative, in the routing specifications with explicit states rather then the resource model. The resource model will then only involve two states *free* and *booked*, see to the right in Figure 5. It is only in exceptional cases advantageous to model alternative in the resource models as to the left in Figure 5.

5.1 *Converting MPPNs into ordinary PNs*

Alternative bookin/unbooking In this example there is an object that is to undergo two operations.

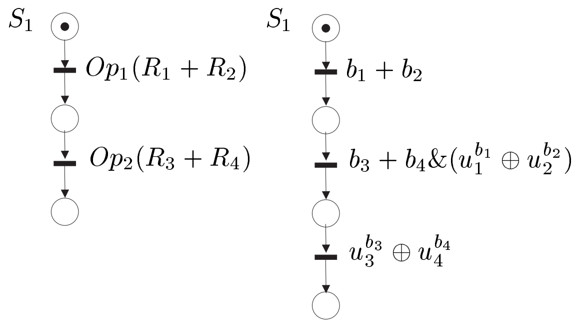
S_1
$Op_1(R_1 + R_2)$
$Op_2(R_3 + R_4)$
S_1
$b_1 + b_2$
$b_3 + b_4\&(u_1^{b_1} \oplus u_2^{b_2})$
$u_3^{b_3} \oplus u_4^{b_4}$

Fig. 6. Routing specification for an alternative resource booking sequence, represented as an High level Routing Specification (HRS) to the left and to the right as an MPPN with booking/unbooking events.

Operation Op_1 require either resource R_1 or R_2 and operation Op_2 requires either resource R_3 or R_4. In Figure 6 a routing specification S_1 is represented both as an High level Routing Specification (HRS), to the left, and an MPPN with booking/unbooking events, to the right.

The resource booking sequence described in Figure 6 represented as an MPPN model is converted into a booking/unbooking model represented as an ordinary Petri net in Figure 7. This is formally done using the join operator together with the condition event.

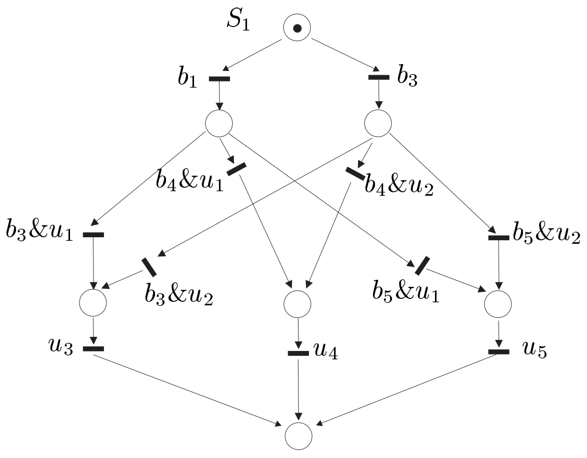

Fig. 7. Routing specification for an alternative resource booking sequence represented as an ordinary Petri net with explicit places for every alternative.

Operations in arbitrary order In this example there is an object that is to undergo four operations. Operation Op_1 requires either resource R_1 or R_2 and operation Op_3, Op_4 and Op_5 requires resource R_3, R_4 and R_5, respectively. Operation Op_3, Op_4 and Op_5 are to be executed in arbitrary order, which is described by the High level Routing Specification (HRS) to the left in Figure 8.

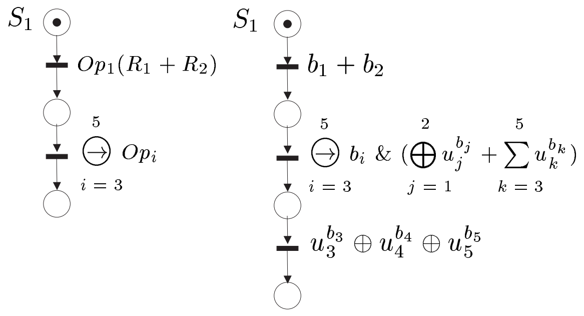
S_1
$Op_1(R_1 + R_2)$
$\bigodot_{i=3}^{5} Op_i$
S_1
$b_1 + b_2$
$\bigodot_{i=3}^{5} b_i \ \& \ (\bigoplus_{j=1}^{2} u_j^{b_j} + \sum_{k=3}^{5} u_k^{b_k})$
$u_3^{b_3} \oplus u_4^{b_4} \oplus u_5^{b_5}$

Fig. 8. Routing specification for an alternative and arbitrary resource booking sequence, represented as an HRS to the left and to the right as an MPPN with booking/unbooking events.

In the second transition in the MPPN model in Figure 8 the arbitrary booking of resources along with the unbooking of the previously booked resource is described.

The MPPN model to the right in Figure 8 is formally converted into a booking/unbooking model represented as an ordinary Petri net in Figure 9 with explicit places for each alternative.

Note that the formal description in Figure 8 do not result in an Petri net model with minimal number of states. When converting an arbitrary order transition into an ordinary Petri net description using the suggested formalism it results in n duplicated states, where n is the number of operations that are to be executed in arbitrary order. This is not a big problem due to what was discussed in the beginning of this Section.

6. CONTROLLER SYNTHESIS

The idea behind a Discrete Event System Controller is to enable and disable events in a system, and in this way force the closed loop system to execute a certain way since disabling some events will remove some execution paths. A common application is to disable all events which may lead to a blocking state or a path leading to a blocking state via uncontrollable events.

To generate a controller the first thing to do is therefore to find a set of blocking states. Since we in this paper only consider controllable events, this only require reachability tests in order to find undesirable states.

Backword reachability is used in the synthesis procedure in order to verify which states that are

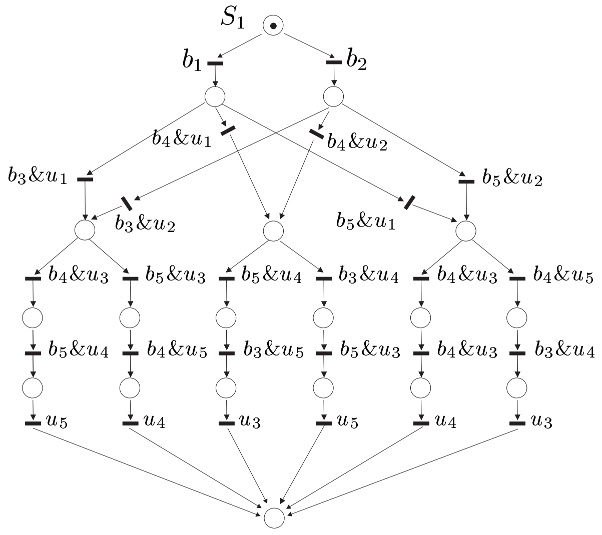

Fig. 9. Routing specification for an alternative and arbitrary represented resource booking sequence as an ordinary Petri net with explicit places for every alternative.

visited in a path that ends in a marked state and may be expressed as:

$$\begin{aligned} B_0(s) &= M \\ B_{k+1}(s) &= R_k(s) \vee (\exists s^{'} .\, T(s, s^{'}) \wedge B_k(s^{'})) \end{aligned}$$

The backward reachability algorithm will answer with a set N which contains all non-blocking states. The Controller is the expressed as

$$T_{controller}(s, s^{'}) = T(s, s^{'}) \wedge N(s^{'})$$

which is the BDD representation of 2. With the suggested specification representation it is not required to model the resources which gives a reduction of required variables. This may together with smoothing result in an efficient calculation of a controller (Vahidi *et al.*, 2001).

7. CONCLUSION

A combination of Petri nets and process algebra is introduced as specification and modeling language for general resource booking problems. The Mixed Process algebra Petri Net (MPPN) implies a compact representation of complex systems.

Process operators have been introduced in order to formally convert the MPPN models into ordinary Petri nets. These Petri net models are efficiently represented as Binary Decision Diagrams and used for synchronization and Controller synthesis. With this representation it is not required to model the resources which gives a reduction of required variables.

8. REFERENCES

Arnold, A. (1994). *Finite Transition Systems: Semantics of Communicating Systems.* International Series in Computer Science. Prentice–Hall International. Englewood Cliffs, NJ.

Bloom, B., A. Cheng and A. Dsouza (1997). Using a protean language to enchance expressiveness in specification. *IEEE Transactions on Software Engineering* **23**(4), 224–234.

Bryant, R.E. (1992). Symbolic boolean manipulation with ordered binary decition diagrams. *ACM computing Surveys.*

Burch, J., E. Clarke and D. Long (1991). Symbolic model checking with partitioned transition relations. In: *Proc. of VLSI'91.* Edinburgh, Scotland.

Camurati, P., F. Corno and P. Pronetto (1993). Exploiting symbolic traversal techniques for efficient process algebra manipulation. In: *Proc. of 1993 CHDL, IFIP Conference on Hardware Description Languages.* Ottawa, Canada. pp. 21–34.

Corno, F., M. Cusinato, M. Ferrero and P. Pronetto (1995). Proving testing preorders for process algebra descriptions. In: *Proc. of 1995 European Design and Test Conference, ED&TC.* pp. 333–337.

Fabian, M. (1995). On Object- Oriented Non-deterministic Supervisory Control. Phd thesis. Control and Automation Laboratory, Chalmers University of Technology. Göteborg, Sweden. Technical report 282.

Falkman, P. and B. Lennartson (2001). Combined process algebra and petri nets for specification of resource booking problems. In: *Proc. of 2001 IEEE American Control Conference.* Arlington, VA, USA.

Gunnarson, J. (1997). Symbolic Methods and Tools for Discrete Event Dynamic Systems. Phd thesis. Linköping, University. Division of Automatic Control, Department of Electrical Engineering.

Hoare, C.A.R. (1985). *Communicating Sequential Processes.* International Series in Computer Science. Prentice–Hall International. Englewood Cliffs, NJ.

Lennartson, B., M. Fabian, M. Tittus and A. Hellgren (1998). Modeling primitives for supervisory control. In: *Proc of WODES '98.* Cagliari, Italy.

Ramadge, P.J. and W.M. Wonham (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.* **25**(1), 206–230.

Vahidi, A., B. Lennartson, Dennis Arkeryd and M. Fabian (2001). Efficient application of symbolic tools for resource booking problems. Proc of ACC'01, Washington.