

Testing Modularity of Local Supervisors: An Approach Based on Abstractions

Patrícia N. Pena, José E.R. Cury and Stéphane Lafortune

Proceedings of 8th International Workshop on Discrete Event Systems - WODES'06
10-12 of July, 2006, Ann Arbor, MI, US.

Testing Modularity of Local Supervisors: An Approach Based on Abstractions

Patrícia N. Pena
DAS - Federal University of
Santa Catarina, Brazil
Currently at EECS - The University
of Michigan, Ann Arbor, MI, USA
Email: pat@das.ufsc.br

José E.R Cury
Department of Automation and
Systems (DAS) - Federal University
of Santa Catarina,
Florianópolis, SC, Brazil
Email: cury@das.ufsc.br

Stéphane Lafortune
Department of Electrical Engineering
and Computer Science (EECS)
The University of Michigan,
Ann Arbor, MI, USA
Email: stephane@eecs.umich.edu

Abstract—This paper presents an efficient way to detect conflict in composed systems controlled by local supervisors designed using the Supervisory Control Theory of discrete event systems. The idea is to apply the required modularity test not over the languages implemented by the supervisors, but over abstractions of the supervisors with some specific characteristics. The concept of observer and some constraints on the set of relevant events are the basis for the approach. An illustrative example is presented.

I. INTRODUCTION

Discrete-Event Systems (DES) are dynamical systems with state changes that are driven by discrete events. Some examples of DES are manufacturing systems and communication networks. In the early 1980's, Ramadge and Wonham started an effort to develop a control theory for DES, under the formalism of finite state machines (FSMs) and formal languages. The resulting body of work is known as Supervisory Control Theory [?]. Despite the significant advances in recent years, the formal techniques are not being widely employed in industry. One of the main obstacles in industrial applications is the complexity of supervisor synthesis as it involves composition of the specifications and the global plant. This factor is very restrictive as it may cause state explosion in large scale systems. Several authors have tried to overcome this restriction by exploiting aspects of the system, such as modularity [?] [?] and symmetry [?], [?] among others. It is worthwhile mentioning references [?], [?], [?] and [?] that address the problem of controlling concurrent (usually termed modular) systems through local supervisors. However, those papers consider only prefix-closed languages.

Most large scale systems are modeled through the composition of many smaller subsystems usually representing concurrent operations and, in general, there are many specifications that restrict only parts of the global plant. In many cases, the specifications only intend to synchronize concurrent subsystems. Modular control, introduced in [?], is a natural solution to deal with such systems because it divides the overall task into subtasks and assigns them to different controllers. The “local modular approach” in [?], an extension of modular control, not only divides the tasks in subtasks, but also expresses each specification (and the corresponding supervisor) only in terms of its “local

plant”. The local plant of a specification is the composition of the subplants that are affected by the specification. The advantage of this approach is that it is not necessary to compose the entire plant in the synthesis of the supervisors, thereby mitigating the state explosion problem. However, as there are multiple supervisors, there is the possibility of getting into a situation of blocking when the supervisors are combined. In such a case, the supervisors are said to be “conflicting”. To check if the local supervisors obtained by modular synthesis are nonconflicting a modularity test is required. The condition of being nonconflicting is also termed “modular” in [?], which is the terminology that is used in this paper. Two or more languages are said to be modular if whenever they share a prefix, they also share a word containing this prefix.

Though the local modular approach has solved the problem of state explosion in the synthesis of supervisors, the verification of modularity is still problematic. To verify the local modularity property all the local supervisors must be composed and this composition may itself cause state explosion. The problem of conflict occurrence is addressed in the literature in different ways. Some authors have developed approaches where the supervisors are nonconflicting by construction [?]. Others have solved the blocking problem by using coordinators [?], [?], [?]. Recently, there have been some works that address the problem of detecting efficiently the occurrence of conflict [?], [?] and [?].

This paper presents an efficient way to detect the presence of conflict in systems controlled by local supervisors. The idea is to apply the modularity test not over the languages implemented by the supervisors, but over abstractions of the supervisors. In the worst case, the test based on abstractions has the same complexity as the original one. In practice, we have found that the test based on abstractions leads to computational savings.

In Section ?? we present a review of some basic concepts of languages and automata theory and of Supervisory Control of DES. Section ?? presents the main results of the paper followed by an illustrative example in Section ??. Section ?? presents the conclusions of the paper. Due to space constraints, proofs of technical results are omitted; they are available in [?].

II. PRELIMINARIES

In this section, we recall some concepts and notations as well as the basic ideas behind the local modular supervisory control approach. The paper is set in the supervisory control framework of Ramadge and Wonham [?]. We refer the reader to [?] or [?] for a detailed introduction to the theory. In this framework, a DES is modelled as an FSM $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states, Σ is the set of events, δ is the transition function, q_0 is the initial state, and Q_m is the set of marked states. Σ^* is the set of all finite strings of elements in Σ , including the empty string ϵ . A language is a subset of Σ^* . The behavior of G , modelled as a language $\mathcal{L}(G) \subseteq \Sigma^*$, is the set of finite strings that G can generate. G can model a second language, $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$, that is the set of strings that represent completed tasks (or, equivalently, that end in marked states).

The prefix closure of a language L (represented by \bar{L}) is the set of all prefixes of strings in L . A language is said prefix-closed if $\bar{L} = L$.

A. Natural Projections and Observers

The natural projection $P_i : \Sigma^* \rightarrow \Sigma_i^*$ is a map with the following characteristics:

$$\begin{aligned} P_i(\epsilon) &= \epsilon \\ P_i(s\sigma) &= \begin{cases} P_i(s) & \text{if } s \in \Sigma^*, \sigma \notin \Sigma_i \\ P_i(s)\sigma & \text{if } s \in \Sigma^*, \sigma \in \Sigma_i. \end{cases} \end{aligned}$$

In words, the projection erases the events of Σ that are not in Σ_i . The concept of natural projection can be extended to languages as follows:

$$P_i(L) = \{u_i \in \Sigma_i^* \mid u_i = P_i(u) \text{ for some } u \in L\}.$$

The inverse projection is, then, defined as

$$P_i^{-1}(L_i) = \{u \in \Sigma^* \mid P_i(u) \in L_i\}.$$

Given a set of languages L_i over event sets Σ_i , $i \in \{1, \dots, n\}$ with $\Sigma = \bigcup_{i=1}^n \Sigma_i$, the notion of inverse projection is used to give a formal definition of the synchronous product (or parallel composition) of languages, as follows:

$$\prod_{i=1}^n L_i = \bigcap_{i=1}^n P_i^{-1}(L_i).$$

The property of distributivity of projection over synchronous product, considered in [?] and extensively used in the proofs of this paper, is presented below as a proposition.

Proposition 1 [?] *Let $L_i \subseteq \Sigma_i^*$, $i \in I = \{1, \dots, n\}$, $\Sigma = \bigcup_{i=1}^n \Sigma_i$, $\Sigma_r \subseteq \Sigma$, $P_{\Sigma \rightarrow \Sigma_r} : \Sigma^* \rightarrow \Sigma_r^*$ and $P_{\Sigma_i \rightarrow (\Sigma_i \cap \Sigma_r)} : \Sigma_i^* \rightarrow (\Sigma_i \cap \Sigma_r)^*$.*

$$P_{\Sigma \rightarrow \Sigma_r} \left(\prod_{i=1}^n L_i \right) = \prod_{i=1}^n P_{\Sigma_i \rightarrow (\Sigma_i \cap \Sigma_r)}(L_i)$$

if $\Sigma_c \subseteq \Sigma_r$ where $\Sigma_c = \cup(\Sigma_j \cap \Sigma_l)$, $\forall j, l \in I$ with $j \neq l$.

The property of projections known as *observer property* will be used in this paper. It was introduced in [?] and is presented as a definition.

Definition 1 [?] *Let $S \subseteq \Sigma^*$ be a language, $\Sigma' \subseteq \Sigma$ an event set and $\theta : \Sigma^* \rightarrow \Sigma'^*$ the natural projection of strings in Σ^* to strings in Σ'^* . If*

$$\begin{aligned} (\forall a \in \bar{S})(\forall b \in \Sigma'^*) \theta(a)b \in \theta(S) \implies \\ (\exists c \in \Sigma^*) \theta(ac) = \theta(a)b \text{ and } ac \in S \end{aligned}$$

then the projection is said to have the observer property.

It is known from [?] that the time complexity of computing projections is at worst exponential and that the size of the state space of the FSM that represents the projected language can increase exponentially with the number of states in the original system. However, if the projection possesses the observer property, it is guaranteed that the FSM that represents the projection always has a number of states not greater than that of the minimal generator for the original language and that it can be obtained in polynomial time [?].

The main theoretical result of this paper is based on natural projections that possess the observer property, as will be seen in Section ??.

B. Supervisory Control of DES

In order to synthesize supervisors, the models of the plant and the specifications for the closed-loop behavior have to be obtained. The set of events used to model the plant are divided into controllable (the ones that can be disabled by the supervisor) and uncontrollable (the ones that cannot be disabled, usually representing the spontaneous events of the plant). The action of the supervisor over the plant is to inhibit the occurrence of controllable events in order to achieve the desired behavior. Sometimes the desired behavior cannot be achieved. In such cases, the supervisor will implement the supremal controllable sublanguage of the desired language, named $Sup\mathcal{C}(K, \mathcal{L}_m(G))$, where K is the desired language and $\mathcal{L}_m(G)$ is the open-loop behavior of the plant. Monolithic supervisory control of DES consists of obtaining one plant and one specification by the composition of all subplants and specifications, respectively, and subsequent calculus of a unique supervisor that implements $Sup\mathcal{C}(K, \mathcal{L}_m(G))$.

C. Blocking and Modularity Test

The concept of blocking is related to the idea of not being able to reach a marked state from some state of the FSM. A FSM G is said to be nonblocking if $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. The fundamental point is that the conjunction of two (or more) nonblocking machines may lead to a blocking FSM. In such a case, we say that the two (or more) FSMs are conflicting.

The modularity test was introduced in [?] to check if supervisors obtained through the modular approach are non-conflicting. Let S_i be a set of languages, $i = \{1 \dots n\}$. The modularity test consists of checking if the equality below holds:

$$\bigcap_{i=1}^n \overline{S_i} = \overline{\bigcap_{i=1}^n S_i}. \quad (1)$$

In words, if two or more languages share a prefix, they must also share a word containing this prefix.

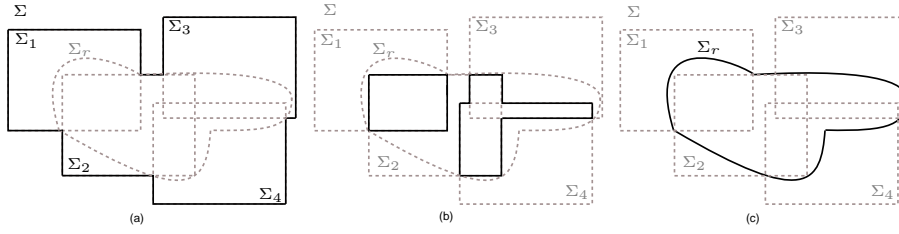


Fig. 1. Venn Diagrams of the inclusion of the event sets for $n = 4$: (a) $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4$ and $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$ (b) Initial $\Sigma_r = \Sigma_c = \cup(\Sigma_j \cap \Sigma_l)$, $\forall j, l \in I = \{1 \dots 4\}$ with $j \neq l$; (c) Final Σ_r , after extending initial Σ_r of (b).

The modularity test in equation (??) can only be applied when all the supervisors have the same event set. In a more general case, where the supervisors' event sets are different, the modularity test need to be adapted to cope with this situation. The adapted modularity test, named local modularity test by the authors in [?], is shown in equation (??):

$$\prod_{i=1}^n \overline{S_i} = \overline{\prod_{i=1}^n S_i}. \quad (2)$$

The local modularity test can be used to check the non-conflict property in sets of supervisors obtained through any method whenever their languages represent the closed-loop language of their system composed by plant+supervisor. The test if equation (??) becomes the one in equation (??) when the event sets of all supervisors are the same.

As can be seen in equation (??), to perform the test, all supervisors have to be composed, what may lead to state explosion.

D. Notation

Consider the event set $\Sigma = \bigcup_{i=1}^n \Sigma_i$, the languages $S_i \subseteq \Sigma_i$ and, $\Sigma_r \subseteq \Sigma$ as being the set of events considered relevant to the conflict. The natural projections that lead to the abstractions are named $\theta_i : \Sigma_i^* \rightarrow \Sigma_r^*$ where $\Sigma_i^* = \Sigma_i \cap \Sigma_r$ and $\theta : \Sigma^* \rightarrow \Sigma_r^*$. So, we have that $\theta_i(S_i) \subseteq \Sigma_i^*$ represents the abstraction of S_i . The event set Σ_c is the set of events that are shared by any two or more supervisors, namely $\Sigma_c = \cup(\Sigma_j \cap \Sigma_l)$, $\forall j, l \in I = \{1 \dots n\}$ with $j \neq l$.

III. MAIN RESULTS

One important outstanding problem in the application of modular supervisory control to actual systems is the computational complexity of the modularity test in equation (??). This section presents a novel modularity test performed over abstractions of the supervisors obtained by the natural projection of these supervisors to a subset of their events. More precisely, the objective is to identify sufficient conditions for the desired abstractions, denoted by $\theta_i(S_i)$ for supervisor S_i , so that the following property holds:

$$\prod_{i=1}^n \overline{\theta_i(S_i)} = \overline{\prod_{i=1}^n \theta_i(S_i)} \iff \prod_{i=1}^n \overline{S_i} = \overline{\prod_{i=1}^n S_i}. \quad (3)$$

The abstractions $\theta_i(S_i)$, in our approach, have the characteristics listed below:

- 1) common events are in Σ_r i.e., $\Sigma_c \subseteq \Sigma_r$;

- 2) observer property.

Hereafter, we call an abstraction satisfying the observer property an *OP-abstraction*. All the results presented in this paper rely in a set Σ_r that contains all events that are shared by more than one supervisor. Figure ?? presents an illustration of those sets of events.

In order to show equation (??), we first establish the following lemma. The proofs of Lemma ?? and Theorems ?? and ?? are omitted in the paper, but are available in [?].

Lemma 1 *Let $S_i \subseteq \Sigma_i^*$, $s \in \Sigma^*$, $t \in \Sigma_r^*$, the projections θ_i , θ , $P_{\Sigma \rightarrow \Sigma_i}$ be defined as before. Assume that $\Sigma_j \cap \Sigma_l \subseteq \Sigma_r$, $\forall j, l \in I = \{1, \dots, n\}$ with $j \neq l$. If $\theta(s)t \in \prod_{i=1}^n \theta_i(S_i)$ then $\exists t_i \in \Sigma_i^*$, $\forall i \in I$, such that the following statements are true:*

- i. $\theta(s)t \in \prod_{i=1}^n \theta_i(P_{\Sigma \rightarrow \Sigma_i} s)t_i$
- ii. $\theta_i(P_{\Sigma \rightarrow \Sigma_i} s)t_i \in \theta_i(S_i)$
- iii. $\prod_{i=1}^n \theta_i(P_{\Sigma \rightarrow \Sigma_i} s)t_i \subseteq \prod_{i=1}^n \theta_i(S_i)$.

Theorem ?? presents the main theoretical result of the paper.

Theorem 1 *Using the definitions presented before, if the natural projections $\theta_i(S_i)$, $\forall i \in I$, are OP-abstractions and if $\Sigma_j \cap \Sigma_l \subseteq \Sigma_r$, $\forall j, l \in I$ with $j \neq l$, then*

$$\prod_{i=1}^n \overline{\theta_i(S_i)} = \overline{\prod_{i=1}^n \theta_i(S_i)} \iff \prod_{i=1}^n \overline{S_i} = \overline{\prod_{i=1}^n S_i}.$$

Theorem 1 shows that it is indistinctive to take the modularity test over the supervisors or over their abstractions. Since the abstractions are *OP-abstractions*, they will have, in general, state space smaller than the original supervisors. The composition of the abstractions is not necessarily smaller than the composition of the original supervisors though. However, Theorem ?? shows that the composition of *OP-abstractions* will be also an *OP-abstraction*.

Theorem 2 *Using the definitions presented before, if, $\forall i \in I$, the natural projections $\theta_i(S_i)$ are OP-abstractions and if $\Sigma_j \cap \Sigma_l \subseteq \Sigma_r$, $\forall j, l \in I$ with $j \neq l$ then $\theta(\prod_{i=1}^n S_i)$ is also an OP-abstraction.*

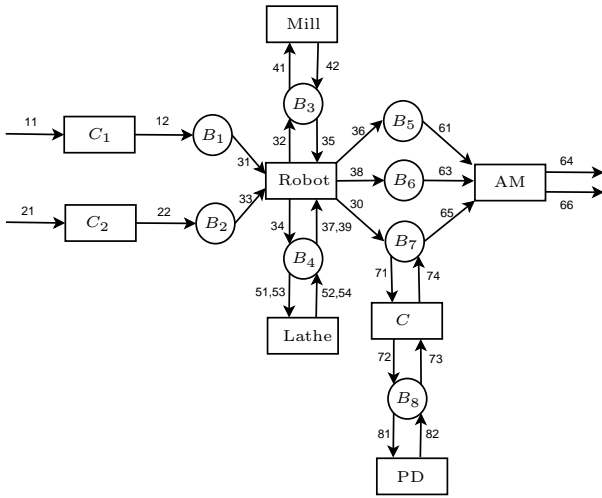


Fig. 2. Flexible Manufacturing System

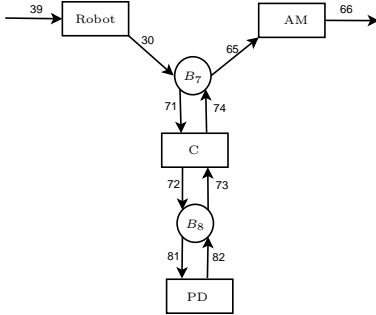


Fig. 3. Diagram of the partial problem

Theorem ?? together with Proposition ?? shows that if all abstractions $\theta_i(S_i)$ are *OP-abstractions*, their composition $\prod_{i=1}^n \theta_i(S_i) = \theta(\prod_{i=1}^n S_i)$ is also an *OP-abstraction*. From [?], we can say that the minimum automaton that represents the language $\theta(\prod_{i=1}^n S_i)$ has, in the worst case, the same number of states as the minimum automaton that represents the language $\prod_{i=1}^n S_i$. Namely, the modularity test over abstractions will have, in the worst case, the same size as the original modularity test.

The problem of obtaining abstractions with the observer property is addressed in [?] and [?]. The first paper presents a method to refine a causal reporter map to obtain an optimal observer (with the smallest refinement) in polynomial time. This algorithm requires some relabeling of events, what is not permitted in natural projections. Reference [?] adapts the algorithm presented in [?] to deal with natural projections. It shows that the problem of finding a minimal extension to a given initial set of relevant events (in this context, sometimes referred to as observable events) to attain an *OP-abstraction* is NP-hard. Nevertheless, it proposes a polynomial (with respect to states and transitions) algorithm that returns a reasonable extension, not necessarily minimal, of the set of observable events, such that the required observer property holds.

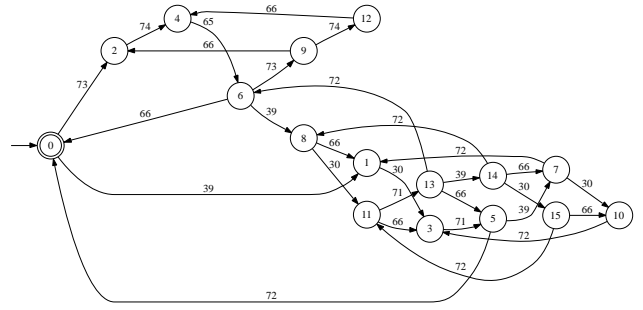


Fig. 4. Supervisor S_1

Thus, in order to apply the result of Theorem ??, we propose the following procedure:

- 1) Define as the initial set of relevant events, the set of events that are common to more than one supervisors;
- 2) Define the initial local relevant events for each supervisor S_i as the intersection of the set obtained in Step 1 with Σ_i ;
- 3) Apply the algorithm presented in [?] to each supervisor in order to get extensions of those initial observable events sets, leading to *OP-abstractions*.

All the events that are rendered observable by applying the algorithm in [?] to a particular supervisor are necessarily not shared with any other supervisor. Thus, the algorithm can be applied to all supervisors *independently* and *concurrently*. The final set of relevant events is composed of the initial set of relevant events (obtained in Step 1) and the extensions returned by the algorithm (Step 3) when applied to each supervisor.

The next section presents an example where the modularity test over abstractions is performed and compared with the original modularity test. It shows a reasonable reduction of complexity when testing modularity over abstractions.

IV. EXAMPLE

Consider the Flexible Manufacturing System (FMS) described in [?]. It produces two types of products from raw blocks and raw pegs: a block with a conical pin on top (Product A) and a block with a cylindrical painted pin (Product B). The FMS consists of eight devices: three conveyors (C_1 , C_2 and C), a Mill, a Lathe, a Robot, a Painting Device (PD) and an Assembly Machine (AM) (Fig. ??). The devices are connected through buffers B_k , $k = \{1, \dots, 8\}$, each with capacity for one part. The control problem is to give maximal degree of freedom to the FMS while avoiding overflow or underflow of parts in the buffers. The liveness specifications are: the supervisor should not prevent the manufacture of Product A and Product B nor should it prevent the Lathe from operating simultaneously with the Mill. Furthermore, the supervisor should not prevent the buffers B_i , $i = \{1, \dots, 6\}$ from becoming empty and the buffers B_7 and B_8 from becoming simultaneously empty. The controllable events are labeled by odd numbers.

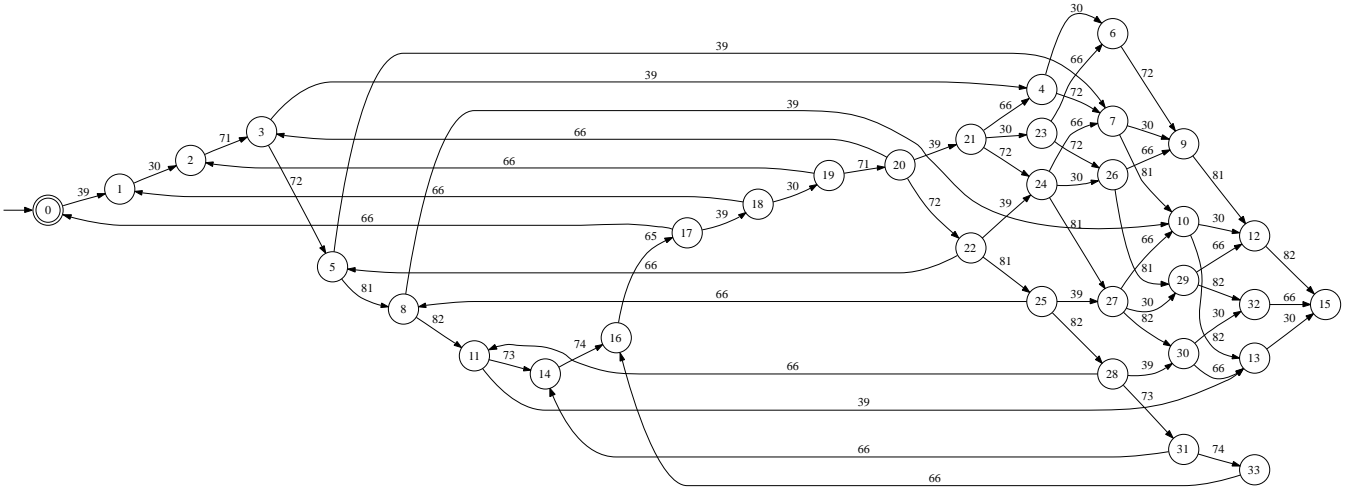


Fig. 6. Modularity test ($S_1||S_2$)

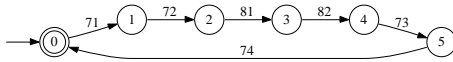


Fig. 5. Supervisor S_2

It turns out that the supervisors built using the local modular approach [?] lead to a situation of conflict. Upon inspection, it is determined that the conflict is caused by the supervisors responsible for controlling underflow and overflow of buffers B_7 and B_8 . To illustrate the result of this paper on an example of moderate scale, we consider only those two supervisors, hereafter named S_1 (for buffer B_7) and S_2 (for buffer B_8). The FMS is reduced to the system shown in Fig. ???. The supervisors S_1 and S_2 are presented in Figs. ?? and ??, respectively.

The local modularity test consists of checking if the parallel composition of S_1 and S_2 is nonblocking. In Fig. ?? we show that it is blocking (from the rightmost state (15), it is not possible to reach a marked state).

To perform the test with the abstractions, we first should determine what are the events that must be in the relevant set Σ_r . As mentioned before, Σ_r must contain all events that are common to more than one supervisors ($\Sigma_c = \{71, 72, 73, 74\} \subseteq \Sigma_r$). It was determined that the projection of S_1 to the subset $\{30, 39, 65, 71, 72, 73, 74\}$ is an *OP-abstraction*. This abstraction of supervisor S_1 is presented in Fig. ??.

It was not possible to find an abstraction with the required characteristics for S_2 . Fortunately, that is not a problem since we can just treat S_2 as an observer of itself, meaning that $\theta_2(S_2) = S_2$.

The modularity test over the abstractions consists of taking the parallel composition of $\theta_1(S_1)$ and $\theta_2(S_2)$ (or equivalently $\theta_1(S_1)||S_2$) and checking if it is nonblocking. Figure ?? shows the result of the test. It can be seen that it is not possible to reach a marked state from some states (for example state 15, the upper rightmost state).

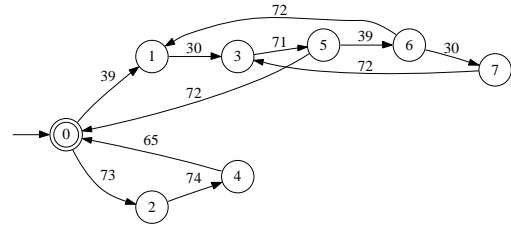


Fig. 7. Abstraction $\theta_1(S_1)$

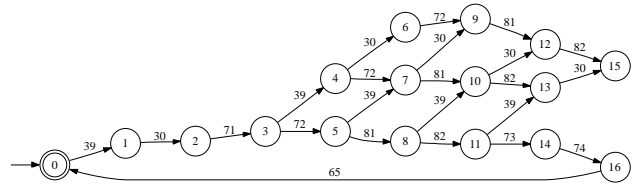


Fig. 8. Modularity test over the abstractions ($\theta_1(S_1)||S_2$)

Notice that it was possible to detect the conflict occurrence in a simpler (less states and transitions) modularity test. Comparing Fig. ?? with Fig. ??, a reduction in the size of the test from 34 states and 62 transitions to 17 states and 23 transitions can be noticed.

V. CONCLUSIONS

In this paper, we presented a novel modularity test based on abstractions of the supervisors (obtained through natural projections) with some specific characteristics, such as being *OP-abstractions*.

Obtaining an abstraction with the observer property is a difficult problem, especially if the original supervisor has a large state space. However, if the supervisor is built using approaches such as the local modular control, the resulting supervisor will tend to be small. In addition, the complexity of obtaining the abstractions does not necessarily increase

when adding more subsystems and specifications, since they are obtained locally.

ACKNOWLEDGMENT

The first author is supported by CNPq, and the second is supported in part by CNPq grant 300953/93-3. The research of the third author is supported in part by NSF grant CCR-0325571 and by ONR grant N00014-03-1-0232.