Supervisory Control of Finite State Machines

A. Aziz[1]*, F. Balarin[2], R. K. Brayton[1], M. D. DiBenedetto[3], A. Saldanha[2], A. L. Sangiovanni-Vincentelli[1]

[1] Dept. of EECS, University of California, Berkeley, CA, USA
[2] Cadence Berkeley Laboratories, Berkeley, CA, USA
[3] Dpt. di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Rome, Italy

Abstract. We address a problem of finding a finite state machine (FSM), which composed with a given FSM, satisfies a given specification. The composition we use is the standard synchronous automata composition restricted to cases which correctly model hardware interconnection. For the satisfaction relation, we use language containment. We present a procedure that will generate a solution (if one exists) which is maximal, i.e. contains behaviors of all other solutions.

1 Introduction

We consider the problem of finding a *controller* Q for a given *plant* P such that *put together*, P and Q *conform* to some specification $\mathcal{M}$. The plant and the controller are (possibly incompletely specified) finite state machines. The specification $\mathcal{M}$ is given by a deterministic finite state automaton. "Putting together" corresponds to the standard synchronous composition of automata, and the notion of conformance is language containment. Our main result is the procedure that either returns the empty set (if the problem does not have a solution), or returns a finite state machine that is maximal, in the sense that it includes behaviors of all solutions to the problem.

Similar problems have been considered in the control community under the label *"model matching"* [5, 6], in the discrete event system (DES) community under the label *"supervisory control"* [18, 15], in concurrency theory they appear as *"scheduler synthesis"* [17] and *"equation solving"* [12, 10], and in the logic synthesis community as *"interacting FSM synthesis"* [16, 1]. Compared to model matching approaches [5, 6] we limit somewhat the choice of possible controllers. The limitation is not serious in hardware context, because it rules out only those circuits that form a loop of combinational gates when composed with P (avoiding combinational loops is considered good design practice). On the positive side, we allow more general specifications of P and $\mathcal{M}$, and provide a

* Supported by SRC under contract 95-DC-324A.

uniform methodology that is applicable to various model matching problems. This general framework also strictly subsumes the problem considered in [16]. Compared to supervisory control of DES [15], our approach offers the advantage of being compatible with finite state machine techniques that have seen continuous developments in the past three decades (e.g. [11, 16]), provides more natural model of reactive system (an argument also made in [2]), and allows significantly simpler development of results.

We have chosen language containment as a satisfaction relation because it allows loose specifications, where a range of behaviors may be acceptable. Here we differ from most of the previous approaches in the process algebra settings, where a much stronger relation, typically some form of bisimulation equivalence is used [12, 10]. The exception is [9] which offers a general framework where the satisfaction relation is not set a priori, but can be defined by a formula in a logic that can express, among other relations, both simulation and bisimulation. However, the procedure presented in [9] generates only a single solution. We believe that it is advantageous to separate the solution process in two stages: first, all the possible solutions are characterized (which is the topic of this paper), and then one is chosen according to some optimality criteria (e.g. minimum state using [11]).

In the rest of this paper, we first review relevant elements of the theory of finite state automata (section 2). Then, we define finite state machines in section 3. In section 4 we develop our results in a simplified setting, and then we generalize them in section 5.

2 Deterministic Finite State Automata

A *deterministic finite state automaton* (DFA) over some finite *alphabet* $\mathcal{X}$ is a 4-tuple (S, s_0, δ, F), where S is some (finite and nonempty) *set of states*, $s_0 \in S$ is the *initial state*, $\delta : (S \times \mathcal{X}) \mapsto S$ is a partial function called the *transition function*, and $F \subseteq S$ is the set of *final states*. The transition function is extended to strings in $\mathcal{X}^*$ as follows (where ϵ denotes an empty string):

$$\begin{aligned} \delta(s, \epsilon) &= s \qquad\qquad \forall s \in S \ , \\ \delta(s, Xx) &= \delta(\delta(s, X), x) \ \forall s \in S, \ \forall X \in \mathcal{X}^*, \ \forall x \in \mathcal{X} \ , \end{aligned}$$

where $\delta(s, Xx)$ is defined only if both $\delta(s, X)$ and $\delta(\delta(s, X), x)$ are.

As a notational convention, we use calligraphic letters to denote alphabets, capitals to denote strings, and lower case letters to denote symbols in the alphabet. Also, given two strings $X = (x_1 x_2 \cdots x_n) \in \mathcal{X}^*$ and $Y = (y_1 y_2 \cdots y_n) \in \mathcal{Y}^*$, we write (X, Y) to denote the string:

$$((x_1, y_1)(x_2, y_2) \cdots (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^* \ .$$

The language of some DFA $A = (S, s_0, \delta, F)$ over $\mathcal{X}$ (denoted $\mathcal{L}(A)$) is the set of all strings $X \in \mathcal{X}^*$ satisfying $\delta(s_0, X) \in F$. Given some language $\mathcal{M} \subseteq (\mathcal{X} \times \mathcal{Y})^*$, the *projection* of $\mathcal{M}$ on $\mathcal{X}$ is defined by:

$$\mathcal{M} \downarrow_{\mathcal{X}} = \{X \in \mathcal{X}^* \mid \exists Y \in \mathcal{Y}^* \text{ such that } (X, Y) \in \mathcal{M}\} \ .$$

If $A = (S, s_0, \delta, F)$ is a DFA over $\mathcal{X}$ and $S' \subseteq S$ is some subset of states, then the *restriction of A to S'* is the automaton $A|_{S'} = (S' \cup \{s_0\}, s_0, \hat{\delta}, S' \cap F)$, where (for all states $s \in S' \cup \{s_0\}$ and all $x \in \mathcal{X}$):

$$\hat{\delta}(s,x) = \begin{cases} \delta(s,x) & \text{if } s \in S' \text{ and } \delta(s,x) \in S' \\ \text{undefined} & \text{otherwise,} \end{cases}$$

Note that $\mathcal{L}(A|_{S'})$ is always contained in $\mathcal{L}(A)$, and that $\mathcal{L}(A|_{S'})$ is empty if S' does not contain s_0.

3 Finite State Machines

Definition 1. A DFA $A = (S, s_0, \delta, F)$ defined over an alphabet $\mathcal{X} \times \mathcal{Y}$ is said to be a *finite state machine (FSM)*, over input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}$, if:

1. all the states in S are reachable and final (i.e. $F = S$), and
2. for all states $s \in S$ and all input letters $x \in \mathcal{X}$ there exists an output letter $y \in \mathcal{Y}$ such that $\delta(s, x, y)$ is defined.

For brevity, we say that a FSM is defined over $\mathcal{X} \mapsto \mathcal{Y}$ whenever its input alphabet is $\mathcal{X}$ and its output alphabet is $\mathcal{Y}$. Not every regular language is definable by a FSM, as stated in the following proposition:

Proposition 2. *A regular language $\mathcal{M} \subseteq (\mathcal{X} \times \mathcal{Y})^*$ is the language of some FSM defined over $\mathcal{X} \mapsto \mathcal{Y}$ if and only if:*

1. *$\mathcal{M}$ is prefix-closed, and*
2. *for all strings $(X, Y) \in \mathcal{M}$ and all input letters $x \in \mathcal{X}$ there exists an output letter $y \in \mathcal{Y}$ such that $(Xx, Yy) \in \mathcal{M}$.*

The condition 2 above is similar to the notion of *controllability* in [15]: at any point of time, for any input (which we cannot control), there must be some output defined.

Note that even though all FSM's are DFA's, some of them are considered non-deterministic in the FSM literature (called *pseudo non-deterministic* in [16]), because for any given state and input, there may exist more than one choice of outputs and next states. To avoid confusion, we use the term *"complete specification"* for this stronger notion of determinism.

Definition 3. A FSM (S, s_0, δ, S) is said to be a *completely specified finite state machine (CSFSM)* if for all states $s \in S$ and all input letters $x \in \mathcal{X}$ there exists a unique output letter $y \in \mathcal{Y}$ such that $\delta(s, x, y)$ is defined.

CSFSM's have straightforward interpretations as sequential circuits that include combinational gates and latches. Of special interest is a class of circuits known as *Moore machines* where no purely combinational paths exist from the

inputs to the outputs. This is important because it ensures that connecting sequential circuits together can never create a loop consisting only of combinational gates. Even though such loops can sometimes be used to optimize circuits, in general it is considered good design practice to avoid them, because cyclic circuits are harder to analyze and can have undesired oscillatory behaviors.

It turns out that DFA composition is a good model of an interconnection of sequential circuits, as long as the interconnection introduces no combinational loops. To model the behavior of such loops accurately, more complicated models have to be developed (e.g. [4]). Guided by the design practice, we decided to avoid combinational loops: we recognize a class of FSM's that corresponds to Moore circuits, and define a restricted notion of pair-wise composition where at least one of the FSM's has to be Moore (at least with respect to those inputs that actually participate in the composition).

Definition 4. A FSM (S, s_0, δ, S) over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ is said to be *Moore in* $\mathcal{Y}$ if for every state $s \in S$ and all $(x, y, v) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{V}$: $\delta(s, x, y, v)$ is defined if and only if $\delta(s, x, y', v)$ is defined for all $y' \in \mathcal{Y}$.

Intuitively, in every state, the output value is independent of y inputs, i.e. if v is a possible output value on input x, then it must be so regardless of the value of y. Of course, the next state may depend on y. Since the output does not depend on y inputs, it is possible to implement such a behavior with a Moore circuit, i.e. without a combinational path form y inputs, to the outputs. We will assume that the implementation of Moore FSM's is restricted to such circuits. Again, it is easy to characterize languages definable by Moore FSM's.

Proposition 5. *A regular language* $\mathcal{M} \subseteq (\mathcal{X} \times \mathcal{Y} \times \mathcal{V})^*$ *is the language of some FSM defined over* $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ *which is Moore in* $\mathcal{Y}$ *if and only if:*

1. $\mathcal{M}$ *is prefix-closed, and*
2. *for all strings* $(X, Y, V) \in \mathcal{M}$, *and all* $(x, y) \in \mathcal{X} \times \mathcal{Y}$ *there exists* $v \in \mathcal{V}$ *such that:*
$$(Xx, Yy, Vv) \in \mathcal{M} \ ,$$
3. *for all strings* $(X, Yy, V) \in \mathcal{M}$, *and all* $y' \in \mathcal{Y}$*:*
$$(X, Yy', V) \in \mathcal{M} \ .$$

Now, we are ready to define the composition of FSM's, which (when defined) corresponds to the standard DFA composition:

Definition 6. Given a FSM $A = (S, s_0, \delta_A, S)$ defined over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ and a FSM $B = (Q, q_0, \delta_B, Q)$ defined over $\mathcal{V} \mapsto \mathcal{Y}$, such that either A is Moore in $\mathcal{Y}$ or B is Moore in $\mathcal{V}$, the *composition* $A \otimes B$ is an automaton over $\mathcal{X} \times \mathcal{Y} \times \mathcal{V}$ defined as follows:

$$A \otimes B \ = \ (S \times Q, (s_0, q_0), \delta_{A \otimes B}, S \times Q) \ ,$$

where $\delta_{A\otimes B} : (S \times Q \times \mathcal{X} \times \mathcal{Y} \times \mathcal{V}) \mapsto (S \times Q)$ is defined by:

$$\delta_{A\otimes B}(s,q,x,y,v) = (\delta_A(s,x,y,v),\ \delta_B(q,v,y))$$

whenever $\delta_A(s,x,y,v)$ and $\delta_B(q,v,y)$ are both defined.

It is easy to check that the composition has the *language intersection* property:

$$\mathcal{L}(A \otimes B) \ = \ \{(X,Y,V) \,|\, (X,Y,V) \in \mathcal{L}(A),\ (Y,V) \in \mathcal{L}(B)\} \qquad (1)$$

Another important property is stated in the following lemma:

Lemma 7. *The restriction of $A \otimes B$ to reachable states is a FSM over $\mathcal{X} \mapsto (\mathcal{Y} \times \mathcal{V})$.*

Proof. Condition 1 in Definition 1 is trivially satisfied. To prove condition 2, consider the following two cases:

1. A is Moore in $\mathcal{Y}$. Given a state (s,q) of $A \otimes B$ and an input letter $x \in \mathcal{X}$ let $v \in \mathcal{V}$ be such that $\delta_A(s,x,y',v)$ is defined for some $y' \in \mathcal{Y}$ and let $y \in \mathcal{Y}$ be such that $\delta_B(q,y,v)$ is defined (by Definition 1, such y and v always exist). By Definition 4, $\delta_A(s,x,y,v)$ is also defined, and thus so is $\delta_{A\otimes B}(s,q,x,y,v)$.
2. B is Moore in $\mathcal{V}$. Given a state (s,q) of $A \otimes B$ and an input letter $x \in \mathcal{X}$ let $y \in \mathcal{Y}$ be such that $\delta_B(q,y,v')$ is defined for some $v' \in \mathcal{V}$ and let $v \in \mathcal{V}$ be such that$\delta_A(s,x,y,v)$ is defined. By Definition 4, $\delta_B(q,y,v)$ is also defined, and thus so is $\delta_{A\otimes B}(s,q,x,y,v)$.

□

Note that without a restriction that one of the FSM's be Moore, it would be possible for a composition not to be a FSM. That could happen in the case where connecting corresponding circuits creates a combinational cycle. By adding this restriction we make sure that whenever composition is defined, it has a clear physical interpretation.

4 Supervisory Control

In this section we consider the control problem shown in Figure 1. In this case, the controller Q sets all the inputs to the plant P and can observe (but not modify) all the outputs. The desired behavior is specified in terms of x and y (as indicated by the dashed box in Figure 1). This formulation corresponds to the *strong model matching* problem [6]. A more general formulation is analyzed in section 5.

Definition 8. Given a FSM P over $\mathcal{V} \mapsto \mathcal{Y}$ and a language $\mathcal{M} \subseteq (\mathcal{X} \times \mathcal{Y})^*$, P is said to be *$\mathcal{M}$−controllable* if there exists a FSM Q defined over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ (called a *$\mathcal{M}$−controller* of P) such that:

$$\mathcal{L}(P \otimes Q) \downarrow_{\mathcal{X}\times\mathcal{Y}} \subseteq \mathcal{M} \ . \qquad (2)$$

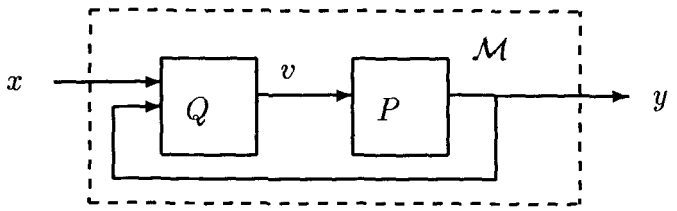

Fig. 1. A simple supervisory control problem.

Implicit in (2) is that $P \otimes Q$ is defined, i.e. that either P is Moore in $\mathcal{V}$ or Q is Moore in $\mathcal{Y}$. Conditions similar to (2) appear in different formalisms as well. In some, one also has to check the "lower bound", i.e for any sequence of inputs there exists some sequence of outputs in the language of $P \otimes Q$. By Lemma 7 such a check is not necessary in our case, because that property is satisfied whenever $P \otimes Q$ is defined.

Still, caution should be taken in case P is not completely specified, and Q is not Moore in $\mathcal{Y}$. Incomplete specification of P typically comes from one of two sources, which should be approached differently:

1. *Unfinished design:* P could be incompletely specified because some design decisions have not been made yet. In this case, we must make sure that any later design decision preserves the property that (modified) P is Moore in $\mathcal{V}$.
2. *Abstraction:* P could be incompletely specified because it is an abstract model (obtained, for example, by hiding some inputs) of a physical circuit (say P'). It might be the case that the abstraction P is Moore in $\mathcal{V}$ even though the circuit P' is not. Since Q will eventually be composed with P' rather than a model P, we should either verify that P' is Moore in $\mathcal{V}$, or require that Q is Moore in $\mathcal{Y}$.

Theorem 9. *If P is a FSM over $\mathcal{V} \mapsto \mathcal{Y}$ and $\mathcal{M} \subseteq (\mathcal{X} \times \mathcal{Y})^*$ is an arbitrary language, then a FSM Q over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ is a $\mathcal{M}$−controller of P if and only if $P \otimes Q$ is defined and:*

$$(V,Y) \in \mathcal{L}(P) \Longrightarrow (X,Y) \in \mathcal{M} \quad \textit{for all } (X,Y,V) \in \mathcal{L}(Q) \ . \tag{3}$$

Proof. Assume $P \otimes Q$ is defined and (3) holds. It follows from (1) that $\mathcal{L}(P \otimes Q) \downarrow_{\mathcal{X}\times\mathcal{Y}} \subseteq \mathcal{M}$. Conversely, assume there exists $(X,Y,V) \in \mathcal{L}(Q)$ such that $(V,Y) \in \mathcal{L}(P)$ and $(X,Y) \notin \mathcal{M}$. By (1), $(X,Y) \in \mathcal{L}(P \otimes Q) \downarrow_{\mathcal{X}\times\mathcal{Y}}$, and hence Q is not a $\mathcal{M}$−controller of P. □

4.1 Constructing the Most General Controller

If the language $\mathcal{M}$ in Theorem 9 is the language of a given DFA M, then it is not hard to construct a DFA with the language:

$$\{(X,Y,V) \in (\mathcal{X} \times \mathcal{Y} \times \mathcal{V})^* \mid (V,Y) \in \mathcal{L}(P) \Longrightarrow (X,Y) \in \mathcal{L}(M)\} \ . \tag{4}$$

If p and m are the number of states of P and M respectively, then a standard procedure for constructing such an automaton (e.g. [8]) yields a DFA with $(p+1)(m+1)$ states. However, since $P \otimes Q$ is a FSM and thus has a prefix-closed language, it is enough to consider the maximal prefix-closed subset of (4). Here, we present a procedure that yields a DFA (denoted $P \Rightarrow M$) with such a language, and $pm+1$ states. Let $P = (S, s_0, \delta_P, S)$ and $M = (Q, q_0, \delta_M, F)$ be given and assume without loss of generality[4] that F contains q_0. The set of states of $P \Rightarrow M$ is $R = (S \times F) \cup \{@\}$ where @ is a distinguished state not in $S \times Q$. The initial state is (s_0, q_0), all the states are final, and the transition function $\delta : (R \times \mathcal{X} \times \mathcal{Y} \times \mathcal{V}) \mapsto R$ is defined by:

$$\delta(r,x,y,v) = \begin{cases} (s',q') & \text{if } r=(s,q) \in S \times F,\ s' = \delta_P(s,v,y), \text{ and} \\ & \quad q' = \delta_M(q,x,y) \in F\ , \\ @ & \text{if } r = @, \text{ or } r=(s,q) \in S \times F, \text{ and} \\ & \quad \delta_P(s,v,y) \text{ is not defined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It follows from Theorem 9 that the problem of finding a $\mathcal{L}(M)$-controller of some FSM P can be reduced to one of the following two problems:

- Find a CSFSM Q (if it exists) such that $\mathcal{L}(Q) \subseteq \mathcal{L}(P \Rightarrow M)$.
- Find a CSFSM Q (if it exists) such that $\mathcal{L}(Q) \subseteq \mathcal{L}(P \Rightarrow M)$, and Q is Moore in $\mathcal{Y}$.

The first problem arises when P is Moore in $\mathcal{V}$. Otherwise we need to construct a FSM Q which is Moore in $\mathcal{Y}$. We devote the rest of this section to these two problems. In both cases we are able to define procedures which eliminate states and transitions of $P \Rightarrow M$, in such a way that if P is not $\mathcal{L}(M)$–controllable, then the procedures eliminate all the states; otherwise they yield a FSM that is a $\mathcal{L}(M)$–controller of P, and whose language includes the languages of all other $\mathcal{L}(M)$–controllers of P.

4.2 Control of Moore FSM's

In this section we address the problem of finding (or proving the absence of) a CSFSM whose language is contained in the language of a given DFA. This problem can be seen as special case of the problem, where one looks for a CSFSM contained (in terms of languages) in a given ω-automaton. The solution to this problem was first suggested by Buchi and Landweber [3]. Unfortunately, that approach is based on Muller automata, which makes it unsuitable for DFA's, because the interpretation of a DFA as a Muller automaton requires exponentially many Muller acceptance conditions. A better solution for our purposes was suggested by Rabin [14], who constructs a tree automaton that accepts exactly

[4] If $q_0 \notin F$, then the empty string is not in $\mathcal{L}(M)$. Since $\mathcal{L}(P \otimes Q)$ always contains the empty string, we conclude that P (or any other FSM for that matter) is not $\mathcal{L}(M)$–controllable.

those trees which represents infinite unfoldings of CSFSM's, which are solutions to the problem. Checking for language emptiness of such a tree automaton solves the problem. Pnueli and Rosner [13] proposed an algorithm for checking language emptiness of Rabin tree automata that runs in $O((nm)^m)$, where n is the number of states and m is the number of acceptance conditions of the original ω-automaton. The algorithm is constructive, i.e. if the language is not empty, it returns one CSFSM accepted by the tree automaton. Any DFA whose states are all final (in particular $P \rightarrow M$) can be treated as ω-automata with a single Rabin acceptance condition. Thus, the Pnueli-Rosner algorithm runs in linear time in our case.

Here, we present an alternative algorithm which does not require tree automata. We do so because our approach extends easily to the similar problem for Moore CSFSM, as shown in the next section.

Lemma 10. *Let $A = (S, s_0, \delta, F)$ be a DFA over $\mathcal{X} \times \mathcal{Y}$, and let $\hat{\psi}$ be a solution to the fix-point equation:*

$$\psi \;=\; \{s \in F \mid \forall x \in \mathcal{X}\; \exists y \in \mathcal{Y} \text{ such that } \delta(s, x, y) \in \psi\} \qquad (5)$$

that contains the initial state s_0. Then the restriction of A to reachable states in $\hat{\psi}$ is a FSM defined over $\mathcal{X} \mapsto \mathcal{Y}$.

Proof. Let S' denote a subset of $\hat{\psi}$ reachable from s_0. All states of $\hat{S}$ are reachable (by assumption) and final (by (5)). Thus, condition 1 in Definition 1 holds. Condition 2 follows immediately from (5). □

It is easy to see that the set of solutions of (5) is finite (because F is finite), non-empty (because empty set is always a solution) and closed under union, and thus there always a unique maximal solution which contains all other solutions.

Theorem 11. *Let $A = (S, s_0, \delta, F)$ be a DFA over $\mathcal{X} \times \mathcal{Y}$ and let $\hat{S}$ be the set containing all the states in the maximal solution of (5) which are reachable from s_0. Then, for any FSM B over $\mathcal{X} \mapsto \mathcal{Y}$: if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$, then also $\mathcal{L}(B) \subseteq \mathcal{L}(A|_{\hat{S}})$.*

Proof. Let a FSM B be such that $\mathcal{L}(B) \subseteq \mathcal{L}(A)$, and let $S' \subseteq S$ be defined by:

$$S' \;=\; \{s \mid s = \delta(s_0, X, Y) \text{ for some } (X, Y) \in \mathcal{L}(B)\}\ .$$

Obviously, the states in S' are all reachable and final, and $\mathcal{L}(B) \subseteq \mathcal{L}(A|_{S'})$. Also, S' is a solution to (5), because (by definition) for every $s \in S'$ there must exist $(X, Y) \in \mathcal{L}(B)$ such that $s = \delta(s_0, X, Y)$, and by Proposition 2, for every such (X, Y) and every $x \in \mathcal{X}$, there exists $y \in \mathcal{Y}$ such that $\delta(\delta(s_0, X, Y), x, y) \in S'$. It follows that $S' \subseteq \hat{S}$, and thus $\mathcal{L}(A|_{S'}) \subseteq \mathcal{L}(A|_{\hat{S}})$. □

It is interesting to note that $\mathcal{L}(A|_{\hat{S}})$ not only contains all the CSFSM-definable languages contained in $\mathcal{L}(A)$, but also contains nothing else, as stated in the following proposition:

Proposition 12. *If A and $\hat{S}$ are as in Theorem 11, then for any $(X,Y) \in \mathcal{L}(A|_{\hat{S}})$ there exists a CSFSM B such that $(X,Y) \in \mathcal{L}(B) \subseteq \mathcal{L}(A|_{\hat{S}})$.*

Proof. Let $T \subseteq S' \times \mathcal{X} \times \mathcal{Y} \times S'$ be the set containing all the transitions in the accepting run of some $(X,Y) \in \mathcal{L}(A|_{\hat{S}})$, and assume for a moment that T does not contain any pairs of transitions with same states and $\mathcal{X}$ value, but with different $\mathcal{Y}$ values. Then, we can eliminate from $A|_{\hat{S}}$ as much transitions not in T as necessary, to create a CSFSM in which (X,Y) has the same run as in $A|_{\hat{S}}$.

The conditions that any two transitions in T with different $\mathcal{Y}$ values must also differ in states or $\mathcal{X}$ values can always be satisfied by unfolding $A|_{\hat{S}}$ finitely many times, so that equivalent (but different) copies of some states are created. □

It is well known that the maximal solution of (5) can be computed by setting $\psi_0 = F$, and then iteratively computing:

$$\psi_{k+1} = \{s \in \psi_k \mid \forall x \in \mathcal{X}\ \exists y \in \mathcal{Y} \text{ such that } \delta(s,x,y) \in \psi_k\} ,$$

until $\psi_{k+1} = \psi_k$. The number of iterations is obviously bounded by the number of states in F. By Theorem 11, if the solution so obtained does not contain s_0, then A does not contain the language of any FSM. Otherwise, $\mathcal{L}(A|_{\hat{S}})$ contains the languages of all the solutions. Choosing one that is the best (according to a given criterion, e.g. minimum state) is a separate, well studied problem which is provably hard, but for which efficient (in practice) algorithms exist (e.g. [11, 16]).

The language $\mathcal{L}(A|_{\hat{S}})$ corresponds to the notion of *"supremal controllable sublanguage"* of [15]. This is a perfect point to illustrate relative simplicity of our approach: Wohnam and Ramadge has devoted a whole paper [18] to fix-point characterization of S' that corresponds to (5).

4.3 Control of Arbitrary FSM's

In this section we address the problem of finding (or proving the absence of) a Moore CSFSM whose language is contained in the language of a given DFA. A similar problem was considered by Golaszewski and Kurshan [7] in the framework of ω-languages (they use the term *"lockup-free"* to describe the Moore property). We improve on this approach by explicitly constructing a FSM that is a solution, and that contains (in term of languages) all other solutions.

Lemma 13. *Let $A = (S, s_0, \delta, F)$ be a DFA over $\mathcal{X} \times \mathcal{Y} \times \mathcal{V}$, and let $\hat{\psi}$ be a solution to the fix-point equation:*

$$\psi = \{s \in F \mid \forall x \in \mathcal{X}\ \exists v \in \mathcal{V}\ \forall y \in \mathcal{Y} \text{ such that } \delta(s,x,y,v) \in \psi\} \qquad (6)$$

that contains the initial state s_0. Then the restriction of A to reachable subset of $\hat{\psi}$ is a FSM defined over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$.

Proof. Let S' denote a subset of $\hat{\psi}$ reachable from s_0. All states of S' are reachable (by assumption) and final (by (6)). Thus, condition 1 in Definition 1 holds. Condition 2 is implied by (6). □

Again, the set of solutions of (6) is finite, non-empty and closed under union, so there exists a unique maximal solution. Note that $A|_{S'}$ is not necessarily Moore in $\mathcal{Y}$. However, it may be made so by eliminating some transitions, as described by the following lemma:

Lemma 14. *Let $A = (S, s_0, \delta, F)$ be a DFA over $\mathcal{X} \times \mathcal{Y} \times \mathcal{V}$, let $\hat{S}$ be the set containing all the states in the maximal solution of (6) which are reachable from s_0, and let $\hat{A}$ be the automaton obtained from $A|_{\hat{S}}$ by modifying its transition function as follows:*

$$\delta_{\hat{A}}(s,x,y,v) = \begin{cases} \delta_{A|_{\hat{S}}}(s,x,y,v) & \text{if } \delta_{A|_{\hat{S}}}(s,x,y'v) \text{ is defined for all } y' \in Y \\ \text{undefined} & \text{otherwise.} \end{cases} , \tag{7}$$

If $s_0 \in \hat{S}$, then $\hat{A}$ is a FSM over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$, and it is Moore in $\mathcal{Y}$.

Proof. All states of $\hat{A}$ are reachable (by assumption) and final (by (6)). By (6), condition 2 in Definition 1 holds even after some transitions are eliminated by (7). Thus, $\hat{A}$ is a FSM over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$. Definition 4 is satisfied by (7). □

Similarly to the case of arbitrary FSM's, the following two results claim that $\mathcal{L}(\hat{A})$ is exactly equal to the union of languages of all CSFSM's B satisfying $\mathcal{L}(B) \subseteq \mathcal{L}(A)$:

Theorem 15. *Let A and $\hat{A}$ be as in Lemma 14. Then, for any FSM B over $(\mathcal{X} \times \mathcal{Y}) \mapsto \mathcal{V}$ which is Moore in $\mathcal{Y}$: if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$, then also $\mathcal{L}(B) \subseteq \mathcal{L}(\hat{A})$.*

Proof. Similar to Theorem 11. □

Proposition 16. *If $\hat{A}$ is as in Lemma 14, then for every $(X,Y,V) \in \mathcal{L}(\hat{A})$ there exists a CSFSM B which is Moore in $\mathcal{Y}$, such that $(X,Y,V) \in \mathcal{L}(B) \subseteq \mathcal{L}(\hat{A})$.*

Proof. Similar to Proposition 12. □

Similarly to (5), the maximal solution to (6) can be computed iteratively in at most $|F|$ iterations, and if that solution contains s_0, then various FSM optimization techniques can be applied to find a CSFSM that is a satisfactory solution to the problem.

4.4 The Cat and Mouse Problem

To illustrate the proposed procedure we use a version of the "cat and mouse problem", originally introduced in [15]. A cat and a mouse are placed in a maze with 5 rooms. The movement between rooms is restricted, and restrictions are different for the cat and the mouse, as summarized in Figure 2. The controller can further restrict the movement by placing a barricade between two rooms, preventing thus both the cat and the mouse from moving between these two rooms. The control objective is to prevent the cat and the mouse from ever being in the same room.

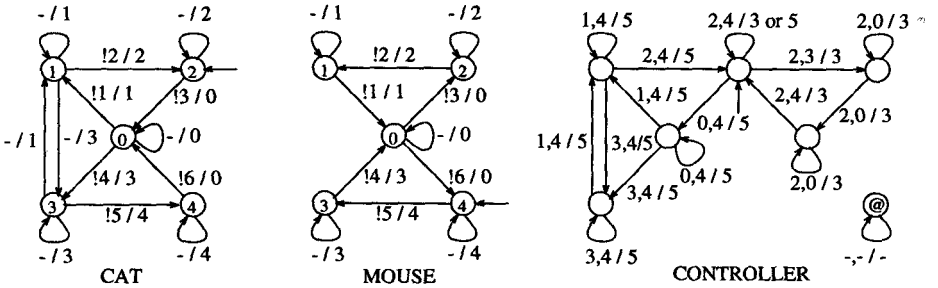

Fig. 2. The plant and the controller for the cat and mouse problem. The label x/y indicates that in a given state on the input x, the output can be set to y. The symbol $-$ denotes any value, and the expression $!n$ denotes any value except n.

In this example, the plant is the composition of the cat and mouse FSM's shown in Figure 2. Both automata have the same input that indicates the position of the barricade, and both automata indicate a choice of the next state at their output. The specification is a single-state DFA with a self-loop which is enabled only if the cat and the mouse differ in their choices of next states. This example, is a special case of the problem in Figure 1 (x variables are missing).

It is easy to check that the cat and the mouse FSM's are not Moore, therefore the procedure from the previous section has to be applied. We have implemented that procedure, and the implementation has generated the solution shown in Figure 2. To enhance readability, we have omitted transitions to the @ state. It is implicit in Figure 2 that from any state and for any input that is not specified, there exists a transition to the @ state, on which any output can be generated. The controller in Figure 2 is Moore and it contains the languages of all the Moore solutions to the cat and mouse problem. In particular, it contains two constant solutions: 3 which keeps the cat forever in room 2, and 5 which keeps the mouse forever in room 4. Computing the solution required less than a second of CPU time on a SUN SPARCstation 5.

5 Generalized Supervisory Control

In this section we discuss a generalization of our approach where we retain from the original model:

- external inputs to the system (denoted x in Figure 1, but renamed x_3 in Figure 3),
- control inputs to the plant (denoted v in Figure 1, but renamed x_4 in Figure 3), and
- measurable external outputs of the plant (denoted y in Figure 1, but renamed y_2 in Figure 3),

and add to it:

- measurable disturbances to the plant (denoted x_2 in Figure 3),
- un-measurable disturbances to the plant (denoted x_1),
- un-measurable outputs of the plant (denoted y_1),
- internal outputs of the plant (denoted y_4), which may be used by a controller, but are otherwise of no interest, and
- outputs that a controller has to generate (denoted y_3).

The desired behavior in this case is specified in terms of $x_1 - x_3$, and $y_1 - y_3$. This general formulation subsumes a series of model matching, rectification, and supervisory control problems. The definition of the compostion needs to be adjusted for this case, but the basic idea remains the same: the composition is defined only if one of the machines is Moore in inputs generated by the other machine.

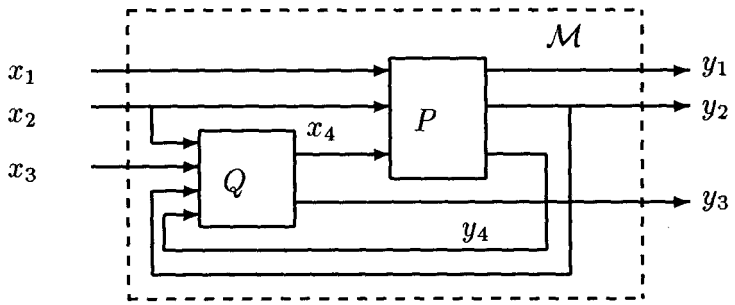

Fig. 3. A generalized supervisory control problem.

Theorem 17. *If P is a FSM over $(\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_4) \mapsto (\mathcal{Y}_1 \times \mathcal{Y}_2 \times \mathcal{Y}_4)$, and $\mathcal{M} \subseteq (\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3 \times \mathcal{Y}_1 \times \mathcal{Y}_2 \times \mathcal{Y}_3)^*$ is an arbitrary language, then a FSM Q over $(\mathcal{X}_2 \times \mathcal{X}_3 \times \mathcal{Y}_2 \times \mathcal{Y}_4) \mapsto (\mathcal{X}_4 \times \mathcal{Y}_3)$ is a $\mathcal{M}$−controller of P if and only if $P \otimes Q$ is defined and $\mathcal{L}(Q)$ is contained in:*

$$\{(X_2, X_3, Y_2, Y_4, X_4, Y_3) \mid \forall X_1 \in \mathcal{X}_1^* \; \forall Y_1 \in \mathcal{Y}_1^* : (X_1, X_2, X_4, Y_1, Y_2, Y_4) \notin \mathcal{L}(P) \text{ or } (X_1, X_2, X_3, Y_1, Y_2, Y_3) \in \mathcal{M}\} \; . \quad (8)$$

Proof. Similar to Theorem 9. □

If $\mathcal{M}$ is specified by a DFA (say M), then we construct a DFA with the language (8) as follows:

1. construct a DFA R_1 such that $\mathcal{L}(R_1) = \overline{\mathcal{L}(P)} \cup \mathcal{L}(M)$ (similar to the construction of $P \Rightarrow M$ in section 4, the size of R_1 is linear in the sizes of P and M),
2. construct a DFA R_2 such that $\mathcal{L}(R_2) = \overline{\mathcal{L}(R_1)}$ (with appropriate representation of final states, this is a constant-time operation, which does not change the size of R_1),

3. construct a DFA R_3 such that $\mathcal{L}(R_3) = \mathcal{L}(R_2) \downarrow_{x_2 \times x_3 \times y_2 \times y_4 \times x_4 \times y_3}$ (this step may incur an exponential blow-up),
4. construct a DFA R_4 such that $\mathcal{L}(R_4) = \overline{\mathcal{L}(R_3)}$ (a constant-time, constant-size operation),

It is not hard to check that the language of R_4 is exactly (8). Once we construct such a DFA, we can apply the procedures described in sections 4.2 and 4.3 to search for appropriate Q.

Acknowledgements

We are grateful to Jerry Burch, Ken McMillan and Vigyan Singhal for many useful discussions and comments.

References

1. Adnan Aziz and Robert K. Brayton. Synthesizing interacting finite state machines. Technical Report UCB/ERL M94/96, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, December 1994.
2. S. Balemi, G. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.
3. J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, April 1969.
4. J.R. Burch, D.L. Dill, E. Wolf, and G. DeMicheli. Modeling hierarcical combinational circuits. In *Digest of Technical Papers of the 1993 IEEE International Conference on CAD*, pages 612–617, November 1993.
5. M.D. DiBenedetto, A. Saldanha, and A. Sangiovanni-Vincentelli. Model matching for finite state machines. In *Proceedings of the IEEE Conference on Decision and Control*, December 1994.
6. M.D. DiBenedetto, A. Saldanha, and A. Sangiovanni-Vincentelli. Strong model matching for finite state machines. In *Proc. of Euorpean Control Conference*, September 1995.
7. C.H. Golaszewski and R.P. Kurshan. Task-driven supervisory control of discrete event systems. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of the Workshop on Computer-Aided Verification*, volume 531 of *LNCS*, pages 282–291. Springer-Verlag, June 1990.
8. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, languages and Computation.* Addison Wesley, 1979.
9. O.H. Jensen, J.T. Lang, C. Jeppesen, and K.G. Larsen. Model construction for implicit specifications in modal logic. *Lecture Notes in Computer Science*, 715, 1993.
10. B. Jonsson and K.G. Larsen. On the complexity of equation solving in process algebra. *Lecture Notes in Computer Science*, 493, 1991. In Proceedings of TAPSOFT'91.

11. T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. A fully implicit algorithm for exact state minimization. In *Proceedings of the 31th ACM/IEEE Design Automation Conference*, pages 684–690, June 1994.
12. J. Parrow. Submodule construction as equation solving in CCS. *Theoretical Computer Science*, 68, 1989.
13. Amir Pneuli and Roni Rosner. On the synthesis of a reactive module. In *Proc. Principles of Programming Languages*, 1989.
14. M.O. Rabin. *Automata on infinite trees an Church's problem*, volume 13 of *Regional Conference Series in Mathematics*. American Mathematical Society, 1972.
15. P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.
16. Yosinori Watanabe. *Logic Optimization of Interacting Components in Synchonous Digital System*. PhD thesis, University of California, Berkeley, 1994. UCB/ERL Mem. No. M94/32.
17. H. Wong-Toi and D.L. Dill. Synthesizing processes and schedulers from temporal specifications. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of the Workshop on Computer-Aided Verification*, volume 531 of *LNCS*, pages 272–281. Springer-Verlag, June 1990.
18. W. Wonham and P. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, May 1987.