On the design of a submodule based on the input/output FSM model*

J. Drissi, N. Yevtushenko,
A. Petrenko and G.V. Bochmann

publication # 1120

Avril 1998

On the design of a submodule based on the input/output FSM model*

J. Drissi[1], *N. Yevtushenko*[2], *A. Petrenko*[3] *and G. v. Bochmann*[1]

1 *Université de Montréal, CP. 6128, Succ. Centre-Ville, Montréal, H3C 3J7, Canada, Phone: (514) 343-7535, Fax: (514) 343-5834, {drissi, bochmann }@iro.umontreal.ca*

2 *Tomsk State University, 36 Lenin str., Tomsk, 634050, Russia, yevtushenko@elephot.tsu.tomsk.su*

3 *CRIM, Centre de Recherche Informatique de Montréal, 1801 Avenue McGill College, Suite 800, Montréal, H3A 2N4, Canada, Phone: (514) 840-1234, Fax: (514) 840-1244, petrenko@crim.ca*

Abstract

This paper addresses the problem of designing a submodule of a given system of communicating input/output FSMs. The problem may be formulated mathematically by the equation $C \Diamond X \cong A$, where C represents the specification of the known part of the system, called the context, A represents the specification of the whole system, X represents the specification of the submodule to be constructed, $\Diamond$ is a composition operator and $\cong$ is the trace equivalence relation. The set of solutions to the equation (if they exist) can be represented as a subset of the set of D-reductions of a proper nondeterministic FSM. The algorithm for finding this nondeterministic FSM is based on the use of a chaos machine and the construction of a machine which separates the permissible and the forbidden traces. After removing all the forbidden traces, we obtain the sought-after machine. Due to the existence of livelocks, some reductions of the obtained machine are not solutions to the equation since their composition with the context can not be modeled by an FSM. If there are no livelocks, the set of solutions to the equation coincides with the set of D-reductions of the obtained machine and then we can characterize all the solutions.

1 Introduction

In designing a complex system, the designer often uses stepwise refinement techniques where the specification is decomposed into interacting modules such that the behavior of the modules in composition is equivalent to the behavior of the overall system specification. The problem of submodule construction arises in this context; it consists of constructing the

* This work was partially supported by the NSERC strategic grant SRTGP200 "Methods for the systematic testing of distributed software systems".

specification of a submodule X when the specification of the overall system and of all submodules except X are given. Approaches to the construction of a submodule specification based on the LTS model have been presented in [Merlin 83, Qin 91, Lin 95]. Here, we consider this problem in the context of the input/output Finite State Machine model (I/O FSM). The direct application of an approach based on the LTS model is not possible since the solutions obtained are not in general I/O FSMs. We have to add constraints on the environment behavior to obtain the system's behavior in the form of an I/O Finite State Machine.

We consider the following problem. Given two deterministic I/O FSMs A and C that represent respectively the behavior of a desired system and the behavior of an existing subsystem called the context, we would like to determine if there exists an I/O FSM X which, combined with the context C, exhibits the behavior of A. We show in this paper that the set of solutions can be represented as a subset of all reductions of a specific nondeterministic I/O FSM. Any reduction of this nondeterministic I/O FSM is a candidate solution, i.e. if its composition with the context does not contain livelocks then it is a solution.

This paper is structured as follows. In Section 2, we define basic notions. Section 3 presents the architecture of the composition of the components followed by the method of determining the set of candidate solutions for a submodule. We conclude in Section 4.

2 Basic notions and definitions

An *input/output nondeterministic finite state machine* (FSM), often simply called a machine throughout this paper, is an initialized Mealy machine which can be formally defined as follows. An NFSM A is a 5-tuple (S, X, Y, h, s_0), where S is a non-empty finite set of states with s_0 as the initial state, X is a non-empty finite set of input symbols, Y is a non-empty finite set of output symbols and h is a total behavior function h: $S\times X \rightarrow \mathbb{P}(S\times Y)\backslash\{\emptyset\}$, where $\mathbb{P}(S\times Y)$ is the powerset of $S\times Y$ [Starke 72].

Given two states s and p, input x and output y, if $(p, y)\in h(s, x)$ then we say that there is a transition from s to p with the input x and output y. Such a transition is denoted $s\text{-}x/y\text{->}p$.

The FSM A is called an *observable* machine, if $|\{s'|\ (s',y)\in h(s,x)\}|\leq 1$ holds for all $(s, x)\in S\times X$ and all $y\in Y$ [Starke 72]. This means, in observable machines, a state and an I/O sequence uniquely determine the next state. The machine A becomes deterministic when $|h(s, x)|=1$ for all $(s, x)\in S\times X$. In a deterministic FSM, instead of the behavior function, we can use two functions: the next state function δ, and the output function λ.

We extend the behavior function to a function h of the FSM A on the set X^* of input sequences containing the empty sequence ε (for convenience we use the same notation h for the extended function, as well, since in our discussion, this does not imply any ambiguity),

i.e., h: $S\times X^* \rightarrow \mathbb{P}(S\times Y^*)\setminus\{\emptyset\}$. Assume $h(s, \varepsilon) = (s, \varepsilon)$ for all $s\in S$, and suppose that $h(s, \beta)$ is already specified. Then

$$h(s, \beta x)=\{(s', \gamma y) \mid \exists\ s''\in S\ [(s'', \gamma) \in h(s, \beta)\ \&(s', y)\in h(s'', x)]\}.$$

The function h^1 is the first and h^2 is the second projection of h, i.e.

$$h^1(s, \alpha) = \{ s' \mid \exists\ \gamma\in Y^*\ (s', \gamma) \in h(s, \alpha)\},$$

$$h^2(s, \alpha) = \{ \gamma \mid \exists\ s'\in S\ \ (s', \gamma) \in h(s, \alpha)\}.$$

An NFSM $B=(\tilde{S}, X, \tilde{Y}, \tilde{h}, s_0)$ is called a *submachine* of A if $\tilde{S}\subseteq S$, $\tilde{Y}\subseteq Y$ and $\tilde{h}(s, x)\subseteq h(s, x)$ for all $(s, x)\in s\times X$.

The *equivalence* relation between two states s of the FSM $A=(S, X, Y, h, s_0)$ and t of the FSM $B=(T, X, Y, H, t_0)$, written $s\cong t$, holds iff $(h^2(s, \alpha)=H^2(t, \alpha))$ for each $\alpha \in X^*$, otherwise, the states are not equivalent. The FSMs A and B are said to be equivalent if their initial states are equivalent, otherwise they are nonequivalent. The equivalence relation between FSMs is sometimes called trace equivalence. The traces of a machine are input/output (I/O) sequences accepted by its initial state. Equivalent machines exhibit identical behaviors, i.e. they execute the same traces.

A state t of the FSM $B=(T, X, Y, H, t_0)$ is said to be a *reduction* of a state s of the FSM $A=(S, X, Y, h, s_0)$, written $t\leq s$, iff $\forall\alpha \in X^*\ (\ H^2(t, \alpha) \subseteq h^2(s, \alpha))$, otherwise t is not a reduction of s, written $t\not\leq s$. The FSM B is said to be a *reduction* of A, $B\leq A$, iff the initial state t_0 of B is a reduction of the initial state s_0 of A. Otherwise, B is not a reduction of A, $B\not\leq A$. If B is deterministic and $B\leq A$, then B is called a *D-reduction* of A.

A *Labeled Transition System* (LTS) is a quadruple $I=(S, L, T, s_0)$, where S is a non-empty finite set of states with s_0 as the initial state, L is a non-empty finite set of observable actions and $T\subseteq S\times(L\cup\{\tau\})\times S$ is a transition relation where τ is a non-observable action. We denote $Tr(s_0)$ the set of traces of I, that is the set of sequences of observable actions of I.

The *parallel composition* of two LTSs $I_1=(S_1, L_1, T_1, s_{01})$ and $I_2=(S_2, L_2, T_2, s_{02})$ without internal actions, written $I_1 \parallel I_2$, is the LTS $I =(S_1\times S_2, L_1\cup L_2, T, (s_{01}, s_{02}))$, where T is defined as follow :

- $a\in L_1\cap L_2 \wedge (s_1, a, s'_1)\in T_1 \wedge (s_2, a, s'_2)\in T_2 \Rightarrow ((s_1, s_2), a, (s'_1, s'_2))\in T$
- $a\in L_1\setminus L_1\cap L_2 \wedge (s_1, a, s'_1)\in T_1 \Rightarrow \forall s_2\in S_2, ((s_1, s_2), a, (s'_1, s_2))\in T$
- $a\in L_2\setminus L_1\cap L_2 \wedge (s_2, a, s'_2)\in T_2 \Rightarrow \forall s_1\in S_1, ((s_1, s_2), a, (s_1, s'_2))\in T$

Let $\alpha=u_1...u_k \in U^*$ and $\beta=z_1...z_k \in Z^*$, we denote by $\alpha\ltimes\beta$ the sequence $u_1z_1...u_kz_k$. Let $\delta\in(X\cup Y)^*$, we denote by $\mathrm{Pr}_X(\delta)$, the projection of δ over X that is obtained by deleting each action that is not in X.

Given an FSM $A=(S, X, Y, h, s_0)$, we say that a deterministic LTS $(S', X\cup Y, T, s_{01})$ *corresponds* to A, if $Tr(s_{01})=\{\ \alpha\ltimes\beta \mid \alpha\in X^* \ \&\ \beta\in h^2(s_0, \alpha)\}$. An LTS corresponding to A can

always be obtained by unfolding each atomic transition $s\text{-}x/y\text{->}s'$ into two consecutive transitions $s\text{-}x\text{->}s''\text{-}y\text{->}s'$. We denote the corresponding LTS by $I_A = (S', X \cup Y, T, s_0)$.

3 The design of a submodule

3.1 A compound system

Many compound systems can be described as a composition of several components. We use here the composition of two communicating components, connected as shown in Figure 3.1 to discuss problems related to the design of a component of a compound system. For the sake of simplicity, we assume that the sets X, X', U, Z, Y and Y' of actions are pairwise disjoint. Actions in $X \cup X'$ are controlled by the environment and only actions in $Y \cup Y'$ can be observed. Note that this composition represents, in fact, many different compositions, assuming that each composition has at least one external input, one external output; while each component must have an input, as well as an output.

We consider the class of systems which can be represented by two FSMs that communicate asynchronously. One FSM, called a *component* machine *Comp*, represents the behavior of a certain component of the system (the submodule to be designed), while the other machine, called a *context* C, models the remaining part of the system.

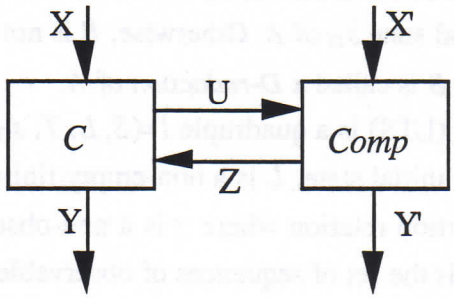

Figure 3.1: The composition of two communicating components C and *Comp*

The composite system cannot be specified as an FSM if its environment does not obeys a proper I/O *ordering constraint*. Specifically, a next external input $x \in X \cup X'$ is only submitted to the system after it has produced an external output $y \in Y \cup Y'$ in response to the previous input. Formally, such an environment can be modeled by the LTS shown in Figure 3.2.

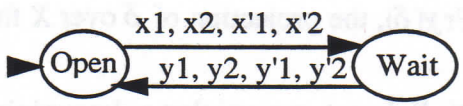

Figure 3.2: Example of an environment E

The collective behavior of the system of two communicating FSMs C and *Comp* in such an environment, can be described by means of a global LTS and an FSM denoted $C \Diamond Comp$ (if it exists). The former describes the behavior in terms of all actions within the system including internals actions, whereas the latter describes the observed behavior in terms of external inputs and outputs only.

We define in this paper, the so-called *global LTS*, which represents the joint behavior of the environment and the composite system, as the LTS $I_C \| I_{Comp} \| I_E$ where I_C is the LTS corresponding to the context C, I_{Comp} is the LTS corresponding to the component *Comp* and I_E is the LTS corresponding to the environment E. The composite system accepts an external input only when it is at a stable state, i.e. when the LTS I_E is in the state *Open*.

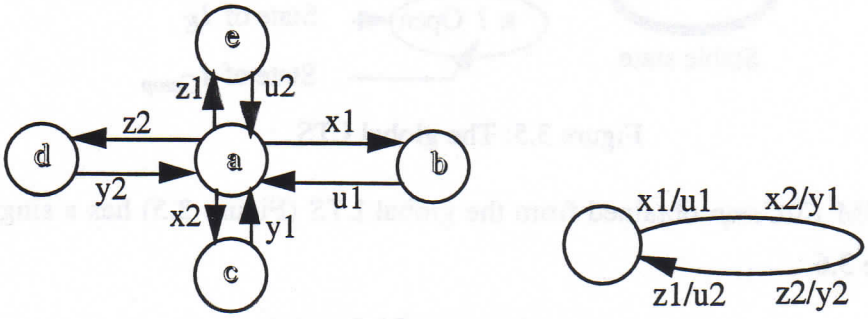

Figure 3.3: The LTS I_C and the context C

To illustrate the computation of the global LTS, we consider the following example. Consider the context C and the component *Comp* in Figures 3.3 and 3.4. We construct the global LTS represented by the graph in Figure 3.5.

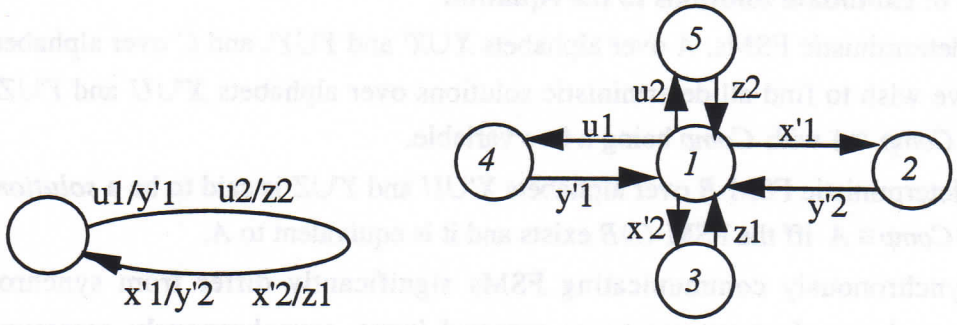

Figure 3.4 : The component *Comp* and the LTS I_{Comp}

Based on the global LTS, the FSM $C \Diamond Comp$, can be obtained provided that there are no livelocks, i.e. cycles labeled only with internal actions. All the internal actions in the global LTS are declared nonobservable actions and the obtained LTS is determinized. The FSM $C \Diamond Comp$ is then obtained by pairing inputs with the subsequent outputs.

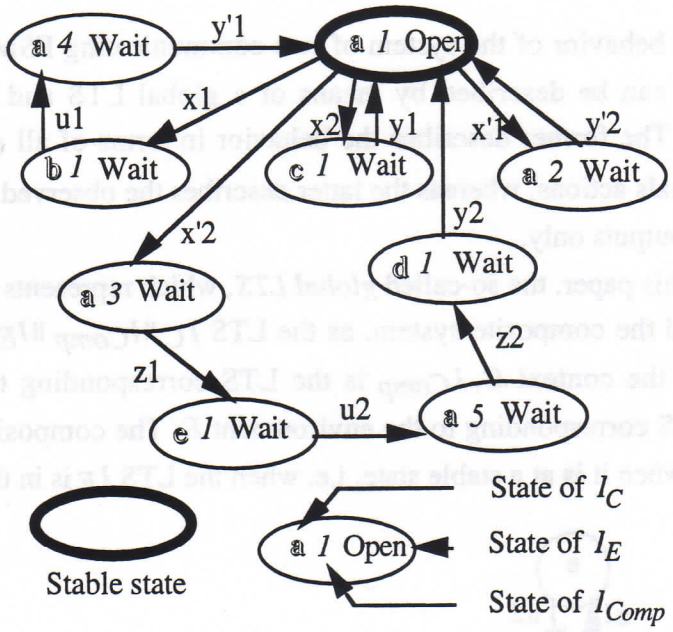

Figure 3.5: The global LTS

The FSM $C \Diamond Comp$ obtained from the global LTS (Figure 3.5) has a single state as shown in Figure 3.6.

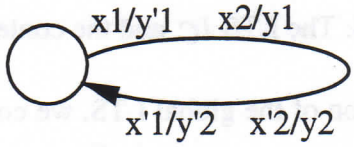

Figure 3.6: The composed FSM

3.2 The set of candidate solutions to the equation

Given two deterministic FSMs, A over alphabets $X \cup X'$ and $Y \cup Y'$, and C over alphabets $X \cup Z$ and $Y \cup U$, we wish to find all deterministic solutions over alphabets $X' \cup U$ and $Y' \cup Z$ to the equation $C \Diamond Comp \cong A$ with $Comp$ being a free variable.

A deterministic FSM B over alphabets $X' \cup U$ and $Y' \cup Z$ is said to be a *solution* to the equation $C \Diamond Comp \cong A$ iff the FSM $C \Diamond B$ exists and it is equivalent to A.

Asynchronously communicating FSMs significantly differ from synchronously interacting machines. In response to an external input, asynchronously communicating machines in a feedback composition may involve into a sequence of interactions before they produce an external output. The number of interactions varies for different global states and inputs, opposed to synchronous systems executing just a single interaction in each global state. For synchronous systems with feedback, in the case where the sets X' and Y' in Figure 3.1 are empty, the set of permissible behaviors to the equation (if not empty), where a

permissible behavior is one represented by a solution, can be represented as the set of reductions of a specific nondeterministic FSM [Watanabe 93, Aziz 95].

We use a chaos machine (Figure 3.7), defined by $Ch=(\{ch\}, X'\cup U, Y'\cup Z, H, ch)$, where $H(ch, v)=\{(ch, w) \mid w\in Y'\cup Z\}$ for all $v\in X'\cup U$. The chaos machine represents all the traces over the input alphabet $X'\cup U$ and the output alphabet $Y'\cup Z$, it contains all possible behaviors of the component machine to be designed.

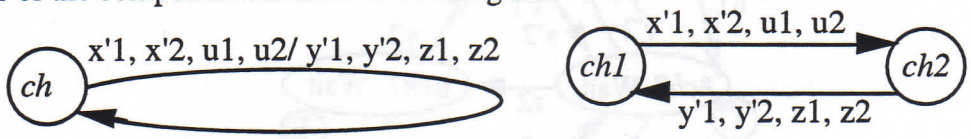

Figure 3.7: The chaos machine and the LTS I_{Ch}

A trace $\beta\bowtie\delta$ of the chaos machine, is *forbidden* w.r.t. an external sequence $\alpha\in(X\cup X')^*$ iff any FSM with the I/O sequence β/δ combined with the given context C exhibits a behavior different from that of the FSM A w.r.t. the input sequence α; otherwise the trace is *permissible* w.r.t. α. In order to classify traces into forbidden and permissible, we construct the global LTS $I_C \| I_{Ch} \| I_E$ and we complete I_A by adding, for each state with an outgoing transition labeled with $y\in Y\cup Y'$, transitions to the deadlock state labeled with all elements of $(Y\cup Y')\setminus\{y\}$, then the obtained LTSs are composed and the resulting LTS is denoted $I_{A,C}$. We hide in $I_{A,C}$ the external inputs and outputs of the context and we derive the LTS $P_{A,C}$ over alphabet $X'\cup U\cup Z\cup Y'$ that characterizes all the forbidden traces. Any forbidden trace $\beta\bowtie\delta$ w.r.t. at least one external input sequence α has a prefix in the LTS $P_{A,C}$ that takes the LTS to the deadlock state. The next step is to transform the LTS $P_{A,C}$ into an FSM, denoted $[[A,C]]$, where all the I/O sequences corresponding to forbidden traces take $[[A,C]]$ from the initial state to the special *FAIL* state. The last step is to derive from $[[A,C]]$ a machine (if it exists), denoted by $[[A,C]]_f$, that contains only permissible traces. Any solution of the equation $C\Diamond Comp\cong A$ is a reduction of $[[A,C]]_f$ but $[[A,C]]_f$ may have reductions that are not solutions because their compositions with C can not be modeled by an FSM due to livelocks. Any reduction of $[[A,C]]_f$ is an FSM over input alphabet $X'\cup U$ and output alphabet $Y'\cup Z$ without forbidden traces; we call such an FSM a *candidate solution*.

If the set of D-reductions of an FSM G coincides with the set of solutions to the equation $C\Diamond Comp\cong A$ then G is called the *largest solution*.

3.2.1 Construction of $[[A,C]]$

We present a method for constructing $[[A,C]]$ and illustrate it using our example.

Step 1. Construct the global LTS $I_C \| I_{Ch} \| I_E$ of the context and chaos machines in the environment E (Figure 3.8).

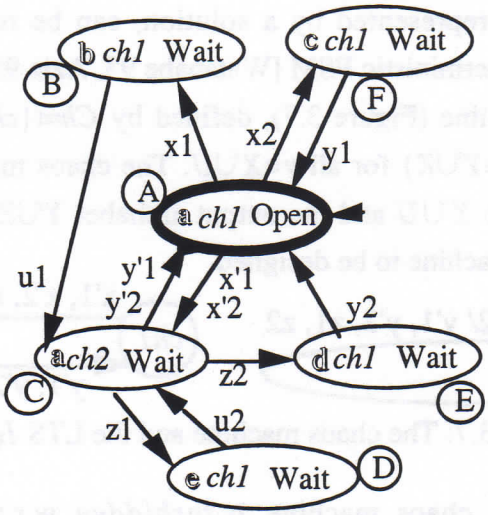

Figure 3.8 : The global LTS $I_C \| I_{Ch} \| I_E$.

Step 2. For each state of the LTS I_A with an outgoing transition labeled with $y \in Y \cup Y'$, add transitions from this state to the deadlock state (a state without outgoing transitions) labeled with each element of $(Y \cup Y') \setminus \{y\}$, and denote the augmented LTS by $\hat{I}_A$ (Figure 3.9). Construct the composition of the global machine $I_C \| I_{Ch} \| I_E$ and $\hat{I}_A$ (Figure 3.10), denoted by $I_{A,C}$, to compare traces of the global machine with those of $\hat{I}_A$. The idea behind augmenting I_A is to represent in $I_{A,C}$ all the forbidden traces of the component to be designed.

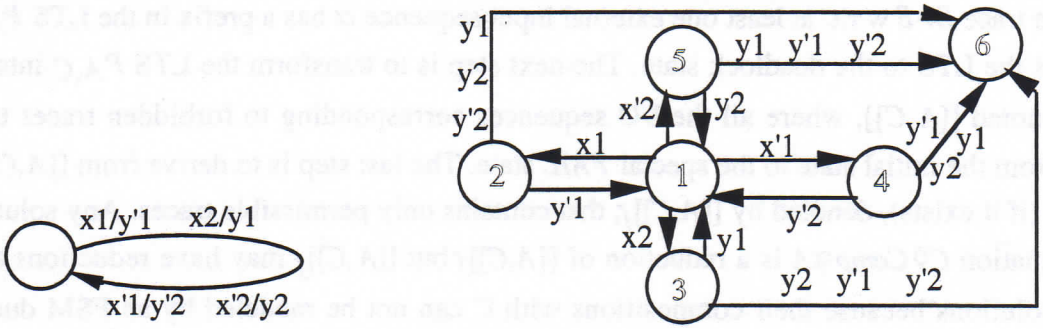

Figure 3.9 : The machine A and the LTS $\hat{I}_A$.

Proposition 3.1. Given a trace $\alpha = \alpha_1 y$ over alphabet $X \cup X' \cup U \cup Z \cup Y \cup Y'$, α takes $I_{A,C}$ to the deadlock state iff α is a trace of the global LTS $I_C \| I_{Ch} \| I_E$ and $\mathrm{Pr}_{Y \cup Y'}(\alpha_1)y'$, with $y \neq y'$, is the output sequence produced by the FSM A in response to $\mathrm{Pr}_{X \cup X'}(\alpha)$.

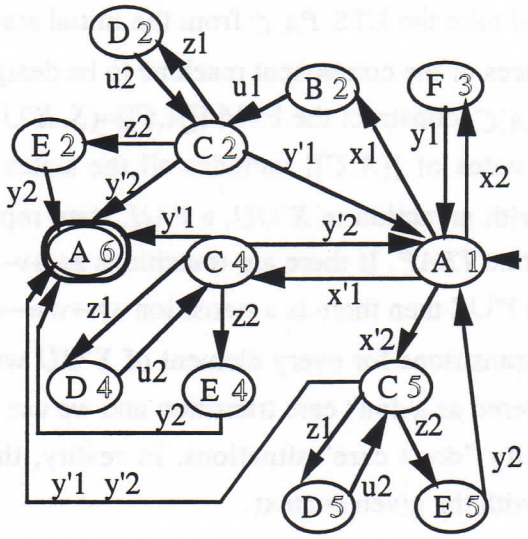

Figure 3.10 : The composition $I_{A,C}$

In other words, a trace of the LTS $I_{A,C}$ taking it from the initial state to the deadlock state represent a trace of the global LTS whose projection over the alphabet of A is not a trace of A. Here we notice that Proposition 3.1 holds only for deterministic FSMs A and C since we assume that each machine produces exactly one output sequence to any input sequence. Our next step is to construct the LTS $P_{A,C}$ which is the $(X' \cup U \cup Z \cup Y')$-projection of $I_{A,C}$ and characterizes the traces that take it from the initial state to the deadlock state.

Step 3. Derive the LTS that represents the projection of traces of $I_{A,C}$ over the alphabet $X' \cup U \cup Y' \cup Z$, i.e. declare all the actions $x \in X$ and $y \in Y$ as nonobservable actions and determinize the obtained LTS. If a subset of states of the determinized LTS comprises a deadlock state of $I_{A,C}$ then we declare this state a deadlock state. The obtained LTS is denoted $P_{A,C}$.

Proposition 3.2. A trace α takes the LTS $P_{A,C}$ to a deadlock state iff there exist a trace α_1 of $I_{A,C}$ such that $\alpha = \mathrm{Pr}_{X' \cup U \cup Y' \cup Z}(\alpha_1)$, and the trace α_1 takes the LTS $I_{A,C}$ to the deadlock state.

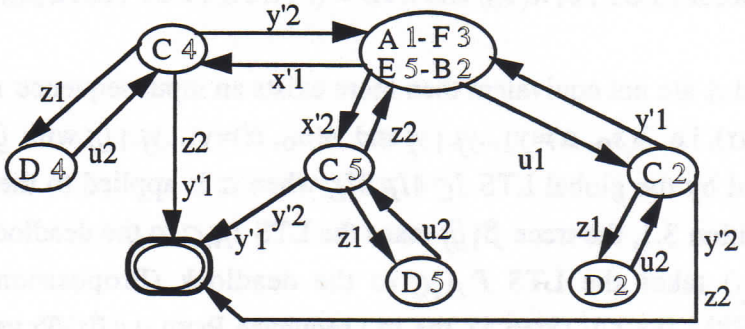

Figure 3.11 : The LTS $P_{A,C}$

In fact, the traces that take the LTS $P_{A,C}$ from the initial state to the deadlock state represent all the forbidden traces of the component machine to be designed.

Step 4. Based on the LTS $P_{A,C}$, construct the FSM $[[A,C]]=(S, X'\cup U, Y'\cup Z\cup\{fail\}, h, s_0)$ with $fail\notin Y'\cup Z$. The set of states of $[[A,C]]$ includes all the states of $P_{A,C}$ that have an outgoing transition labeled with an action in $X'\cup U$, a *FAIL* state representing the deadlock state of $P_{A,C}$ and a special state *TRAP*. If there are transitions $s_i\rightarrow v\rightarrow s_j\rightarrow w\rightarrow s_k$ in the LTS $P_{A,C}$ where $v\in X'\cup U$ and $w\in Y'\cup Z$ then there is a transition $s_i\rightarrow v/w\rightarrow s_k$ in the FSM $[[A,C]]$. The *FAIL* state has looping transitions for every element of $X'\cup U$ with the output *fail*. Any undefined transition is considered as a don't care transition and we use a *TRAP* state similar to [Unger 69] to formally specify "don't care" situations. In reality, these transitions are not executed in the composition with the given context.

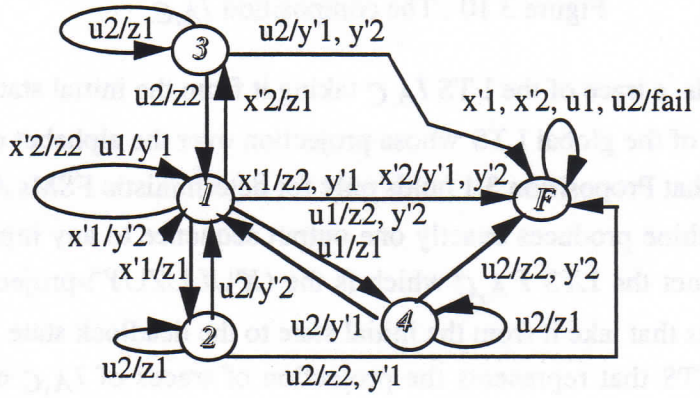

Figure 3.12 : The machine $[[A,C]]$, the transitions leading to the *TRAP* state are not shown.

Proposition 3.3. Given sequences $\beta\in(X'\cup U)^*$ and $\delta\in(Y'\cup Z)^*$, the FSM $[[A,C]]$ has the I/O sequence β/δ iff any proper prefix of the trace $\beta\bowtie\delta$ that is a trace of the LTS $P_{A,C}$ does not take $P_{A,C}$ from the initial state to the deadlock state.

Proposition 3.4. Given a deterministic FSM B over alphabets $X'\cup U$ and $Y'\cup Z$ such that the machine $C\Diamond B$ exists, the FSM $C\Diamond B$ is equivalent to A iff B is a reduction of $[[A,C]]$.

Proof.

Let $A=(S, X\cup X', Y\cup Y', \delta, \lambda, s_0)$ and $C\Diamond B=(P, X\cup X', Y\cup Y', \Delta, \Lambda, p_0)$.

First part. ($\Leftarrow$)

If the FSMs $C\Diamond B$ and A are not equivalent then there exists an input sequence $\alpha\in(X\cup X')$ such that $\lambda(s_0, \alpha)\neq\Lambda(p_0, \alpha)$, i.e. $\lambda(s_0, \alpha)=y_1...y_{j-1}y_j$ and $\Lambda(p_0, \alpha)=y_1...y_{j-1}\tilde{y}_j$ with $\tilde{y}_j\neq y_j$. Let $\beta_1\tilde{y}_j$ be the trace executed by the global LTS $I_C\|I_B\|I_E$ when α is applied to the initial state of $C\Diamond B$. Due to Proposition 3.1, the trace $\beta_1\tilde{y}_j$ takes the LTS $I_{A,C}$ to the deadlock, i.e. the trace $\mathrm{Pr}_{X'\cup U\cup Y'\cup Z}(\beta_1\tilde{y}_j)$ takes the LTS $P_{A,C}$ to the deadlock (Proposition 3.2), and by construction of the FSM $[[A,C]]$ (Step 5), the I/O sequence $\mathrm{Pr}_{X'\cup U}(\beta_1)/\mathrm{Pr}_{Y'\cup Z}(\beta_1\tilde{y}_j)$ takes the FSM $[[A,C]]$ to the *FAIL*-state.

Therefore, the FSM B cannot be a reduction of the FSM $[[A,C]]$ w.r.t. any prolongation of the input sequence $\mathrm{Pr}_{X'\cup U}(\beta_1)$ since the FSM $[[A,C]]$ at the state $FAIL$ produces the output $fail \notin Y'\cup Z$ to any input $v \in X'\cup U$.

Second part. ($\Rightarrow$)

If B is not a reduction of $[[A,C]]$ there exists an I/O sequence v/μ of B that is not an I/O sequence of the FSM $[[A,C]]$. Therefore, there exists a proper prefix β/γ of v/μ such that the trace $\beta\times\gamma$ takes the LTS $P_{A,C}$ from the initial state to the deadlock state (Proposition 3.3), i.e. there exists a trace ξ of $I_{A,C}$ such that $\beta\times\gamma=\mathrm{Pr}_{X'\cup U\cup Y'\cup Z}(\xi)$, and the trace ξ takes the LTS $I_{A,C}$ to the deadlock (Proposition 3.2). Let $\xi=\xi_1 y$, then ξ is a trace of the global LTS $I_C \| I_B \| I_E$ and $\mathrm{Pr}_{Y\cup Y'}(\xi_1)y'$ is the output sequence of the FSM A produced in response to $\mathrm{Pr}_{X\cup X'}(\xi)$, such that $y\neq y'$ (Proposition 3.1).

Thus, the FSMs $C\Diamond B$ and A are not equivalent w.r.t. the input sequence $\mathrm{Pr}_{X\cup X'}(\xi)$. □

3.2.2 Deleting the fail output

We are interested in the problem of finding the set of candidate solutions to the equation $C\Diamond X \cong A$ that can be represented as the set of D-reductions of a machine over alphabets $X'\cup U$ and $Y'\cup Z$, while the FSM $[[A,C]]$ has the output alphabet $Y'\cup Z\cup\{fail\}$. Our next step is to remove from the output set of $[[A, C]]$ the superfluous output *fail* preserving the set of deterministic reductions over alphabets $X'\cup U$ and $Y'\cup Z$ of the FSM $[[A, C]]$.

Speaking more generally, we are facing the following problem. Given an FSM $K=(S, V, W', h, s_0)$ and a subset W of its output alphabet W', we need to construct a submachine D of K over the output set W such that the sets of D-reductions over the alphabets V and W of the two FSMs K and D coincide.

Let B be a D-reduction of K. Then for each state t of B there exists a state s of K such that t is a reduction of s. By this reason, if we delete from K each state s for which there is no state b of a deterministic FSM over the output alphabet W such that $b\leq s$, and delete each transition such that its output is not in W, the resulting submachine will preserve all reductions of K over alphabets V and W. A question arises which properties should have a state s of K in order not to admit a state b of any deterministic FSM over the output alphabet W such that $b\leq s$.

Definition 3.1. Given an FSM $K=(S, V, W', h, s_0)$ and a set $W\subset W'$, a state s of K is said to be an $W(1)$-*redundant* state if there exists an input $v\in V$ such that $h^2(s, v)\cap W=\varnothing$. Suppose we have determined all the $W(k)$-redundant states for $k>0$. A state s of K is said to be $W(k+1)$-redundant if it is $W(k)$-redundant, or there exists an input $v\in V$ such that all the states in $h^1(s, v)$ are $W(k)$-redundant. The state s is said to be W-redundant if there exists an integer k such that it is $W(k)$-redundant.

Since the set S of states of K is finite there exists some $k \le |S|$ such that the sets of $W(k)$-redundant and $W(k+1)$-redundant states coincide. We denote $\hat{S}$ the set of W-redundant states of K.

Proposition 3.5. Given an FSM $K=(S, V, W', h, s_0)$, $W \subset W'$, and a state s of K, if there exists a deterministic FSM $B=(Q, V, W, \delta, \lambda, q_0)$ and a state q of B such that $q \le s$ then s is not a W-redundant state of K.

Proof

1. If there exists a state q of an FSM B such that $q \le s$, then the state s is not $W(1)$-redundant.
2. Assumption of induction. Let the statement of Proposition 3.5 holds for all integers less than k, $k \ge 1$, i.e. if a state of B is a reduction of the state s then the state s is not $W(k\text{-}1)$-redundant.
3. Suppose now that the state s is $W(k)$-redundant and there exists a state q of a deterministic FSM B such that $q \le s$. Then there exists an input $v \in V$ such that for every output $w \in W$ the state $h^1_w(s,v)$ (if it exists) is $W(j)$-redundant, $j<k$. Since q is a reduction of s then the state $q'=\delta(q, v)$ of B should be a reduction of the state $h^1_w(s,v)$, where $w=\lambda(q,v) \in W$. The latter contradicts the assumption of induction. Thus, if q is a reduction of the state s then s is not $W(k)$-redundant for any k, i.e. $s \notin \hat{S}$. □

Proposition 3.6. Given an FSM $K=(S, V, W', h, s_0)$ and a set $W \subset W'$, let $\hat{S}$ be the set of all W-redundant states of K. If $s_0 \in \hat{S}$ then there is no reduction of K over the output set W. Otherwise, the submachine $D=(S \setminus \hat{S}, V, W, \hat{h}, s_0)$ of K, such that $\hat{h}(s,v) = h(s,v) \setminus [\{(s',w') | w' \notin W\} \cup \{(s',w) | s' \in \hat{S}\}]$, for each $(s,v) \in (S \setminus \hat{S}) \times V$ has the same set of D-reductions over alphabets V and W as the FSM K.

Proof

If $s_0 \in \hat{S}$ then there is no state b of any deterministic FSM over the output alphabet W such that $b \le s_0$ (Proposition 3.5), and therefore, the set of reductions of K with the output set W is empty. If $s_0 \notin \hat{S}$ then, by definition of the set $\hat{S}$, the set $\hat{h}(s, v)$ is not empty for each $(s,v) \in (S \setminus \hat{S}) \times V$. Moreover, by construction of $\hat{h}$, $\hat{h}(s,v) \subseteq (S \setminus \hat{S}) \times W$. Thus, D is a submachine of K.

Let $B=(Q, V, W, \delta, \lambda, q_0)$ be a D-reduction of K and $w_1 \ldots w_k$ be the output sequence of B to the input sequence $v_1 \ldots v_k$ applied in the initial state. The state $\delta(q_0, v_1 \ldots v_j)$ is a reduction of the state $h^1_{w1 \ldots wj}(s_0, v_1 \ldots v_j)$ for each $j=1,\ldots,k$, i.e. the state $h^1_{w1 \ldots wj}(s_0, v_1 \ldots v_j) \in S \setminus \hat{S}$ (Proposition 3.5). Thus, $(\delta(q_0, v_1 \ldots v_j), w_j) \in \hat{h}(s_0, v_1 \ldots v_j)$, and B is a reduction of D. □

Proposition 3.6 implies the following algorithm for reducing the FSM $[[A, C]]$ with the output alphabet $Y' \cup Z \cup \{fail\}$ to an FSM with the output alphabet $Y' \cup Z$ that has the same set of D-reductions over alphabets $X' \cup U$ and $Y' \cup Z$ as $[[A, C]]$.

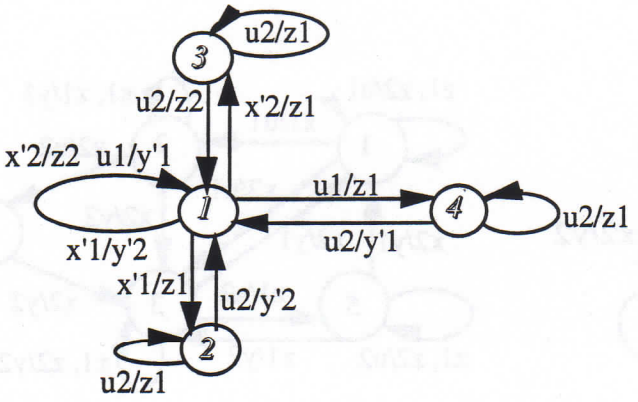

Figure 3.13 : The machine $[[A, C]]_f$, the transitions leading to the *TRAP* state are not shown.

Algorithm

Input: The FSM $[[A,C]] = (S, X'\cup U, Y'\cup Z\cup\{fail\}, h, s_0)$.

Output: A submachine $[[A, C]]_f$ of $[[A, C]]$ over the output alphabet $Y'\cup Z$ with the same set of D-reductions over alphabets $X'\cup U$ and $Y'\cup Z$ as $[[A,C]]$ if such D-reductions exist.

Step 1. Determine the set $\hat{S}$ of $Y'\cup Z$-redundant states of $[[A, C]]$. If $s_0\in\hat{S}$ then the FSM $[[A, C]]$ has no D-reductions over the output alphabet $Y'\cup Z$, and the procedure terminates; otherwise Step 2.

Step 2. For each $(s, v)\in(S\setminus\hat{S})\times X'\cup U$, $\hat{h}(s, v)$ is obtained from $h(s, v)$ by deleting every pair (s', w) such that $s'\in\hat{S}$ or $w\notin Y'\cup Z$. The obtained machine $[[A, C]]_f = (S\setminus\hat{S}, X'\cup U, Y'\cup Z, \hat{h}, s_0)$ has the same set of D-reductions over alphabets $X'\cup U$ and $Y'\cup Z$ as the FSM $[[A, C]]$.

The validity of the above algorithm is stated in the following theorem.

Theorem 3.1. Given deterministic FSMs A and C, if the machine $[[A, C]]_f$ exists then the set of candidate solutions to the equation $C\Diamond X\cong A$ coincides with the set of D-reductions of $[[A, C]]_f$.

In the case when the machine $[[A, C]]_f$ does not exist, there is no solution to the equation $C\Diamond X\cong A$. When the $[[A, C]]_f$ exists, two cases are possible. In the first case, the LTS $I_C \| I_{[[A, C]]_f} \| I_E$ does not contain livelocks, i.e. there are no cycles labeled only with internal actions in $U\cup Z$. In this case, every D-reduction of $[[A, C]]_f$ is a solution to the equation $C\Diamond X\cong A$, and $[[A, C]]_f$ is the largest solution to the equation. In the second case, the LTS $I_C \| I_{[[A, C]]_f} \| I_E$ has livelocks. Here we have two possibilities :

1- every D-reduction F of $[[A, C]]_f$ is not a solution since $C\Diamond F$ does not exist due to livelocks in $I_C \| I_F \| I_E$ (see Example 1, Figure 3.14).

2- the set of solutions to the equation $C\Diamond X\cong A$ is a subset of the set of D-reductions of $[[A, C]]_f$ (see Example 2, Figure 3.15 and Figure 3.16).

Example 1

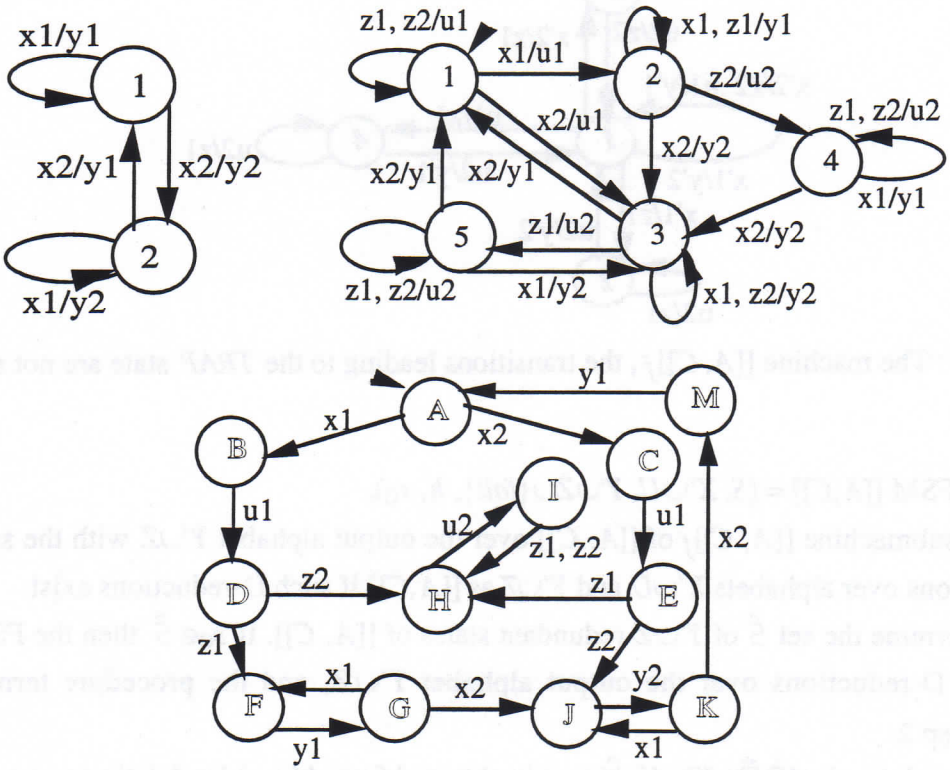

Figure 3.14 : The machines A, C and the LTS $I_C \| I_{Ch} \| I_E$.

The $[[A, C]]_f$ machine obtained is the chaos machine. The machine $C \Diamond F$ does not exist for any D-reduction F of $[[A, C]]_f$ since after any possible output in response to u_1 in the initial state of F, the compound system enters a livelock for an appropriate external input x.

Example 2

We use the same machine A as in Example 1 and the context C shown in Figure 3.16.

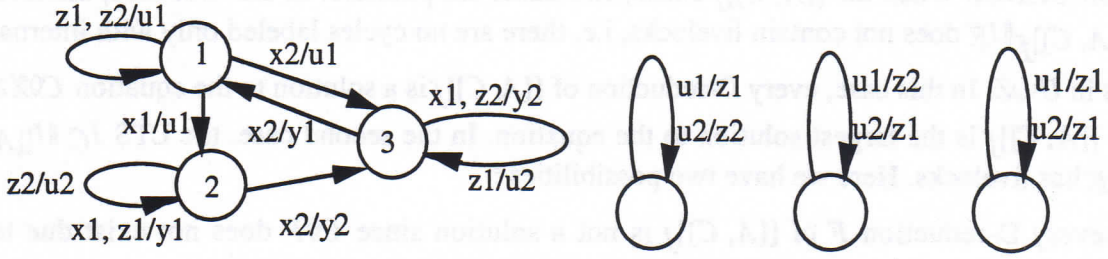

Figure 3.15 : The machines C, F_1, F_2 and F_3.

The $[[A, C]]_f$ machine obtained is again the chaos machine, however its D-reductions F_1 and F_2 are solutions, while the D-reduction F_3 of $[[A, C]]_f$ is not a solution. In this example, the largest solution does not exist since for any I/O sequence of $[[A, C]]_f$ we can

find a D-reduction F which is a solution and includes this I/O sequence. Consider the FSM B_n in Figure 3.16:

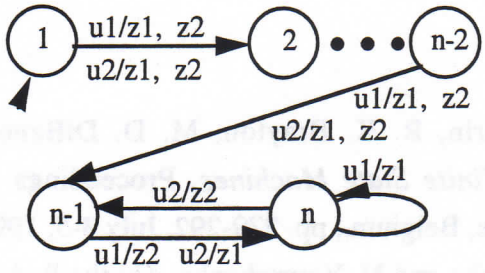

Figure 3.16 : The machine B_n.

The FSM B_n is a reduction of $[[A, C]]_f$ and the LTS $I_C \| I_{Bn} \| I_E$ has no livelocks. Thus, every D-reduction of B_n, $n>1$, is a solution to the equation $C \Diamond X \cong A$. Consider now any I/O sequence over alphabets U and Z of length m. It is an I/O sequence of the FSM B_{m+2} and if we choose a D-submachine F of B_{m+2} having this I/O sequence then F is a solution to the equation $C \Diamond X \cong A$. Therefore, if a largest solution G exists, we have $G \leq [[A, C]]_f$ and $[[A, C]]_f \leq G$, but if we remove any trace from $[[A, C]]_f$ we loose some solutions. This implies that there is no largest solution in this case.

4 Conclusion

We have presented in this paper an approach to solve the problem of submodule construction in the realm of the Finite State Machine model. This problem may be formulated mathematically by the equation $C \Diamond X \cong A$, where C represents the specification of the known part of the system, A represents the specification of the whole system, X represents the specification of the submodule to be constructed, $\Diamond$ is a composition operator and $\cong$ is the trace equivalence relation. The set of solutions to the equation (if they exist) can be represented as a subset of the set of D-reductions of a proper nondeterministic FSM. The algorithm for finding this nondeterministic FSM is based on the use of a chaos machine and the construction of a machine which separates the permissible and the forbidden traces. After removing all the forbidden traces, we obtain the sought-after machine.

Due to the existence of livelocks, some reductions of the obtained machine are not solutions to the equation since their composition with the context can not be modeled by an FSM.

If there are no livelocks, the set of solutions to the equation coincides with the set of D-reductions of the obtained machine and then we can characterize all the solutions. In this case, we can be interested in finding a particular solution for the implementation that optimizes a given criterion. Different criteria can be used for this optimization. The first possible criterion

might be the number of states, in this case, we have to construct a D-reduction of the largest solution with the minimal number of states [Drissi 98]. Others criteria could be the design of a component "easy" to test or having a minimal number of outputs or a minimal number of internal interactions.

References

[Aziz 95] A. Aziz, F. Balarin, R. K. Brayton, M. D. DiBenedetto and A. Saldanha, *Supervisory Control of Finite State Machines*, Proceedings of the 7th International Conference, CAV'95, Liège, Belgium., pp. 279-292, July 3-5, 1995.

[Drissi 98] J. Drissi, A. Petrenko and N. Yevtushenko, *On the Reduction of Nondeterministic Finite State Machine*, to be submitted soon.

[Gill 62] A. Gill, *Introduction to the theory of Finite-State Machines*, Mc Graw-Hill Book Company, Inc, 1962.

[Lin 95] B. Lin, G. de Jong and T. Kolks, *Hierarchical Optimization of Asynchronous Circuits*, Proceedings of the 32nd Design Automation Conference, pp 712-717, 1995.

[Merlin 83] P. Merlin and G. v. Bochmann, *On the Construction of Submodule Specifications and Communication Protocols*, ACM Trans. on Programming Languages and Systems, Vol. 5, No. 1, pp. 1-25, Jan. 1983.

[Petrenko 93] A. Petrenko, N. Yevtushenko, A. Lebedev and A. Das, *Nondeterministic state machines in protocol conformance testing*, Proceedings of the IFIP Sixth International Workshop on Protocol Test Systems, Pau, France, pp. 363-378, September 1993.

[Petrenko 94] A. Petrenko, N. Yevtushenko and G. v. Bochmann, *Experiments on Nondeterministic Systems for the Reduction Relation*, IWTCS'96.

[Petrenko 96] A. Petrenko and G. v. Bochmann, *On fault coverage of tests for finite state specifications*, in Computer Networks and ISDN Systems, special issue on Protocol Testing, 1996.

[Qin 91] H. Qin and P. Lewis, *Factorisation of Finite State Machines under Strong and Observational Equivalences*, Journal of Formal Aspects of Computing, Vol. 3, pp 284-307, July-Sept. 1991.

[Starke 72] P. H. Starke, *Abstract automata*, American Elsevier Publishing Company, Inc-New York, 1972.

[Unger 69] S. H. Unger, Asynchronous Sequential Switching Circuits. New York, Wiley-Interscience, 1969.

[Watanabe 93] Watanabe, Y, and Brayton, R K 'The maximal set of permissible behaviors for fsm networks' *Proc. of the IEEE/ACM International Conference on Computer-Aided Design* , pp 316-320, 1993.

[Wood 87] D. Wood, *Theory of Computation*, John Wiley & Sons, Inc, 1987.