Solving asynchronous equations

Alexandre Petrenko [1] *and Nina Yevtushenko* [2]
1 *CRIM, Centre de Recherche Informatique de Montréal, 550, Sherbrooke St., Suite 100 Montréal, H3A 1B9, Canada, petrenko@crim.ca*

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

Key words: I/O FSMs, nondeterminism, equation solving, verification, submodule construction, controller design, machine factorization, livelocks

Abstract: The paper addresses the problem of solving an equation formulated in terms of the input/output finite state machine model. The relation used in equations is either FSM equivalence or reduction, i.e. trace inclusion. The composition operator corresponds to asynchronous communications between nondeterministic input/output machines. These types of FSM equations are called asynchronous. A procedure for determining the largest potential solution to a given asynchronous equation is proposed. To verify whether or not the largest potential solution satisfies the equation, the absence of livelocks needs to be established. A solution to the livelock problem is also suggested.

1. INTRODUCTION

Designing complex systems usually involves an important step of dividing the whole system into a number of separate components which interact in some well-defined way. In this context, a question arises how to design a component that combined with a known part of the system, called the context, satisfies a given overall specification. The problem is also known as the problem of submodule construction [Mer83], component redesign [Kim72], [Rho91], [Wat93], controller design [Azi95], equation solving [Par89], [Lar90], [Qin91], [Che96] or machine factorization [Qin91]. This problem has many important applications, ranging from hardware optimization to protocol design. In the case of compound systems, one may encounter it at the level of validation [Hee95] and test derivation [Petr93], [Petr96].

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35394-4_29

S. Budkowski et al. (eds.), *Formal Description Techniques and Protocol Specification, Testing and Verification*
© IFIP International Federation for Information Processing 1998

Equation solving has been extensively studied within the process algebra, however, not much work has yet been done to solve equations formulated in terms of input/output finite state machines, I/O FSMs. In the context of hardware optimization [Rho91], one usually assumes strictly synchronous communications between components, as several input signals should be processed simultaneously, see [Yev91], [Wat93], [Azi95]. In many applications, including distributed systems, e.g. protocols, we are obliged to consider more loose communications when components interact asynchronously via message buffers or queues. While the problem of composing asynchronously communicating I/O FSM has been addressed in a number of works in various contexts (mainly within the reachability analysis) [Lee93], [Pet93], [Pet94], we are not aware of any result directly related to the problem of decomposing a nondeterministic I/O machine into asynchronously communicating machines, i.e. equation solving.

This paper addresses the problem of equation solving for a system of I/O possibly nondeterministic FSMs that communicate asynchronously via bounded input queues where actions are stored, assuming that the system has a single message in transit. Under this assumption, any output of one component is immediately consumed by another one as its input, so one may say that communications are performed by rendezvous. However, we cannot directly use the results obtained in process algebra to solve an equation in terms of the FSM model. The problem is that the set of actions of an FSM is divided into two sets, inputs and outputs, and any transition is an atomic transition labeled by an I/O pair. An I/O FSM can, in fact, be transformed into a labeled transition system, but a solution (in trace semantics) to an equation can not always be translated back into an I/O FSM. Another approach has to be used to ensure that the obtained solution to the equation remains in the domain of I/O FSMs. In this paper, we propose a procedure for determining the largest potential solution to the equation, in the sense that any other potential solution is trace included in the obtained machine, i.e. it is a reduction of the largest potential solution. A potential solution becomes a solution if there are no livelocks when it is combined with the given part of the system. When the largest solution to the equation is determined an optimal implementation of the component of interest can be obtained as an appropriate reduction of the largest solution. For example, an optimal implementation should have minimal number of states [Dam94], [Wat94] or transitions. In the case where potential solutions exist and the largest solution does not, it remains an open question whether a solution to the equation exists or not. We demonstrate in this paper that the largest solution may sometimes not exist. We also consider an equation where composed machines are allowed to communicate internally a certain number of times. The idea is to avoid livelocks solving the equation.

The paper is organized as follows. In Section 2, we give necessary basic notions and define a composition operator for I/O nondeterministic finite state machines through a parallel composition operator for labeled transition systems. A method for FSM decomposition in terms of equation solving is presented in Section 3. In Section 4, we propose some enhancements to our method which allow us, solving the equation, to avoid livelocks. Section 5 presents possible future work.

2. PRELIMINARIES

2.1 Finite state machines and labeled transition systems

A finite state machine (FSM), often simply called a machine throughout this paper, is a completely specified, initialized, (possibly nondeterministic) observable Mealy machine which can be formally defined as follows. A *finite state machine* is a quintuple (S, X, Y, h, s_0), where S is a set of states with s_0 as the initial state; X - a finite nonempty set of input symbols; Y - a finite nonempty set of output symbols; and h - a behavior function $h: S\times X \rightarrow \wp(S\times Y)\setminus\varnothing$, where $\wp(S\times Y)$ is the powerset of $S\times Y$ and $|\{s' | (s',y)\in h(s,x)\}|\leq 1$ holds for all $(s,x)\in S\times X$ and all $y\in Y$, i.e. the machines considered in this paper are observable [Star72]. The machine becomes *deterministic* when $|h(s,x)|=1$ for all $(s,x)\in S\times X$. In a deterministic FSM, instead of the behavior function which is required to represent a nondeterministic behavior, we use two functions: the next state function, and the output function. An FSM $B = (S', X, Y', h', s_0)$ is said to be a *submachine* of an FSM $A = (S, X, Y, h, s_0)$ if $S'\subseteq S$, $Y'\subseteq Y$ and $h'(s,x) \subseteq h(s,x)$ for all $(s,x)\in S'\times X$. We extend the behavior function to a function on the set X^* of all input sequences containing an empty sequence ε, i.e., $h: S\times X^*\rightarrow \wp(S\times Y^*)\setminus\varnothing$. For convenience, we use the same notation h for the extended function, as well. 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, often called the next state function, is the first projection of h, while h^2, often called the output function of A, is the second projection of h, i.e., $h^1(s,\alpha) = \{ s' \mid \exists\ \beta \in Y^*\ [(s',\beta) \in h(s,\alpha)] \}$, $h^2(s,\alpha) = \{\beta \mid \exists s'\in S\ [(s',\beta) \in h(s,\alpha)] \}$ for all $\alpha\in X^*$.

Given alphabets X and Y such that $X\cap Y = \varnothing$ and sequences $\alpha = x_1...x_k \in X^*$ and $\beta = y_1...y_k \in Y^*$, the sequence $\alpha\natural\beta = x_1y_1...x_ky_k$ is called a *trace* over alphabets X and Y. Given state $s\in S$ of the FSM $A = (S, X, Y, h, s_0)$, the trace $\alpha\natural\beta$ is called a trace of the FSM A at the state s if $\beta\in h^2(s,\alpha)$ or simply a trace of the FSM A if s coincides with the initial state s_0. Given an input alphabet X and output alphabet Y, there exists a special nondeterministic

FSM such that any trace over these alphabets is a trace of this machine. We call such a machine a *chaos* machine over alphabets X and Y. In particular, a single-state chaos machine is an FSM $(\{p\}, X, Y, H, p)$, where $H(p,x) = \{p\}\times Y$ for all $x\in X$.

Given two states s of the FSM A and r of the FSM $B= (T, X, Y, H, t_0)$, state r is said to be a *reduction* of s, written $r\leq s$, if for any input sequence the condition $H^2(r,\alpha) \subseteq h^2(s,\alpha)$ holds; otherwise, r is not a reduction of s. States s and r are *equivalent* states, written $s\equiv r$, iff $s\leq r$ and $r\leq s$. On the class of deterministic machines, the above relations coincide.

Given two machines, A and B, over the same input alphabet, B is a reduction of A, written $B\leq A$, if the initial state of B is a reduction of the initial state of A. Similarly, the equivalence relation between machines is defined. $B\equiv A$, iff $B\leq A$ and $A\leq B$.

A *labeled transition system* (LTS) is a quadruple (S, L, T, s_0), where S is a finite set of states with s_0 as the initial state, L is a finite nonempty set of observable actions and $T \subseteq S\times(L\cup\{\tau\})\times S$ is a transition relation, where $\tau\notin L$ is a nonobservable action. For state $s\in S$, we denote $in(s)$ and $out(s)$ subsets of actions labeling incoming and outgoing transitions at the state s, respectively. As usually, the set of traces of the LTS I is denoted by $Tr(I)$. The LTS I is said to be *deterministic* if for each state $s\in S$, it holds that $\tau\notin out(s)$ and for each pair (s,a), $s\in S$, $a\in out(s)$, there exists at most one state s' such that $(s,a,s')\in T$.

If the set L of actions of the LTS I is partitioned into sets X and Y, $L = X\cup Y$, $X\cap Y = \varnothing$, then for a sequence σ over the alphabet L, the *X-projection* of σ is a sequence obtained by deleting all the actions $y\in Y$ from the sequence σ, and the *X-projection* of the LTS I [Che96] is an LTS that is obtained by hiding all the actions from Y in I and determinizing the resulting LTS by means of a subset construction [Hop79].

Given an FSM $A = (S, X, Y, h, s_0)$, a minimal deterministic LTS $I = (Q, X\cup Y, T, q_0)$ is said *to correspond* to A, if $Tr(I) = \{\alpha\#\beta \mid \alpha\in X^* \ \&\ \beta\in h^2(s_0,\alpha)\}$. We denote I_A an LTS corresponding to A. An LTS corresponding to an FSM can be obtained through the following steps:

- unfold every atomic transition $s\text{-}x/y\text{->}s'$ into two consecutive transitions $s\text{-}x\text{->}s''\text{-}y\text{->}s'$;
- determinize the obtained LTS by a subset construction;
- minimize the resulting LTS in trace semantics.

2.2 Composition of LTSs

Given two deterministic LTSs $C = (Q, L_1, H, q_0)$ and $D= (T, L_2, G, t_0)$, the *composition* of C and D is the LTS $(Q\times T, L_1\cup L_2, F, q_0t_0)$, denoted $C\|D$,

such that for all states $(q,t) \in Q \times T$ and $a \in L_1 \cup L_2$, a triple $(qt, a, q't')$ is a transition of the LTS $C \| D$ if one of the following conditions holds

- $a \in L_1 \cap L_2$, $(q, a, q') \in H$ and $(t, a, t') \in G$;
- $a \in L_1 \setminus L_1 \cap L_2$, $t' = t$ and $(q, a, q') \in H$;
- $a \in L_2 \setminus L_1 \cap L_2$, $q' = q$ and $(t, a, t') \in G$.

We usually consider the initially connected part of the composition and trim off states that are not reachable from the initial state and their outgoing transitions.

2.3 Composition of FSMs

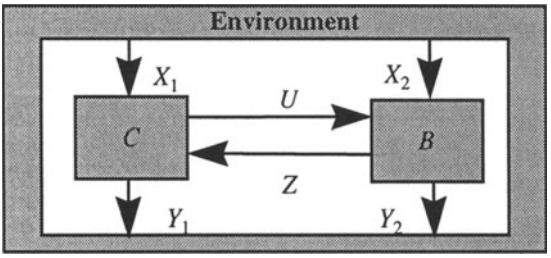


Figure 1. A closed system

We consider a system of two components which are connected as shown in Figure 1 and communicate via bounded input queues, where actions are stored, assuming that the system has a single message in transit. Let the FSM C be defined over alphabets I_1, O_1 and the FSM B over alphabets I_2, O_2 such that $I_1 \cap I_2 = \varnothing$ and $O_1 \cap O_2 = \varnothing$. The following relationships hold between the alphabets: $X_1 = I_1 \setminus O_2$, $Y_1 = O_1 \setminus I_2$, $X_2 = I_2 \setminus O_1$, $Y_2 = O_2 \setminus I_1$, $Z = I_1 \cap O_2$ and $U = I_2 \cap O_1$. We call symbols of the alphabet $X_1 \cup X_2 = X$ *external* inputs, symbols of the alphabet $Y_1 \cup Y_2 = Y$ - *external* outputs, while symbols of the alphabet $Z \cup U$ - *internal actions*. We further assume that at least one of the sets X_1, X_2 and at least one of the sets Y_1, Y_2 are not empty. Otherwise, i.e. when the system has no inputs or no outputs, it can not be represented by an FSM. As the system has a single message in transit, at any moment either the components exchange messages or one of them communicates with its environment which submits a next external input to the system only after the system has produced an external output in response to the previous input. In other words, we consider a closed system, where the environment is described by the LTS L_E with two states (a total chaos system) shown in Figure 2. The behavior of the closed system can be described by the composition $L_C \| L_B \| L_E$ where L_C and L_B are LTSs corresponding to FSMs C and B.

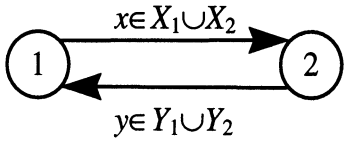


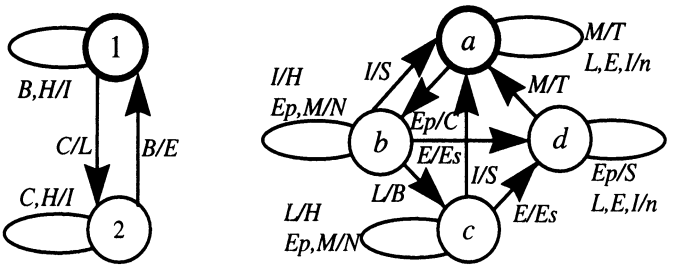
Figure 2. The environment LTS

If the system $L_C \| L_B \| L_E$ has no livelocks then it can be transformed into an FSM over the external inputs X and external outputs Y. The system of the FSMs C and B is said to have *a livelock w.r.t. an external input sequence* $\alpha \in X^*$ if there exists a trace β of the LTS $L_C \| L_B \| L_E$ such that α is the X-projection of β and the LTS has a path labeled with α containing a cycle labeled only with internal actions in $Z \cup U$. A livelock is said to be an *output deadlock* if no state s with $out(s) \cap Y \neq \varnothing$ is reachable from the cycle. The system in an output deadlock will never produce an external output even in the case of a non-catastrophic interpretation of livelocks when it is assumed that a system may get out of a livelock. We assume here a catastrophic interpretation of livelocks (a divergent LTS) and leave the composition operator for FSMs undefined. Formally, we define the composition operator $\Diamond$ for two FSMs as follows.

Given the LTS $L_C \| L_B \| L_E$ without livelocks, we determine $(X \cup Y)$-projection of the LTS $L_C \| L_B \| L_E$ and transform the projection into an FSM, denoted $C \Diamond B$, by pairing each input with a subsequent output and replacing the pair of corresponding transitions by a single transition. We are usually interested in the initially connected part of $C \Diamond B$, called a *composite* machine of C and B. We keep the same notation $C \Diamond B$ for a composite machine.

Compared with LTSs, the definition of the composition operator on FSMs is slightly more involved, and opposed to the composition of LTSs, the operator is not defined in case of livelocks. Another difference is that the environment is required to alternate between inputs and outputs, enabling the extraction of the composite FSM $C \Diamond B$ from the LTS which describes the closed system.

Our definition of the composition operator is, of course, applicable to two FSMs communicating without feedback. If, for example, $U = \varnothing$ or $Z = \varnothing$ we have a well-known *loop-free* composition of FSMs [Cec86]. The case when both sets are empty corresponds to a *parallel* (independent) composition of two FSMs. If the sets U, X_1 and Y_2 or Z, X_2 and Y_1 are empty then the system is a *sequential* composition of two FSMs studied, e.g. in [Kim72], [Petr94]. All these compositions are constrained versions of the one considered in this paper.

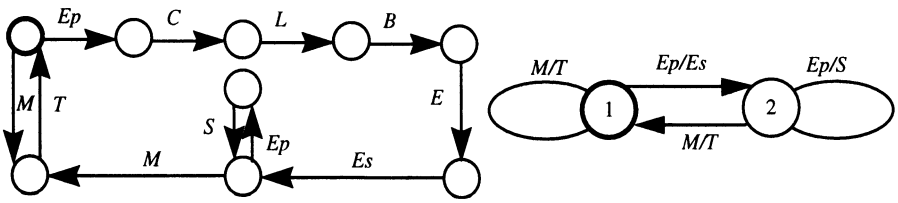


(a) (b)

Figure 3. An espresso machine, the FSM *B* (a), and waiter, the FSM *C* (b)

Example. Consider a coffee shop which has a machine that produces a cup of espresso, it is the FSM *C* with the inputs **C**oin, **B**utton, and **H**it, and the outputs **L**amp, **E**spresso, **I**dle, shown in Figure 3 (a) and a waiter, the FSM *B* that accepts two inputs, **E**spresso-**p**lease and **M**oney, from the environment (customer) and three inputs, **L**amp, **E**spresso and **I**dle, from the coffee machine, the outputs are **E**spresso-served, **T**hanks, **S**orry, **N**o action (to the environment); **C**oin, **B**utton, **H**it (to the espresso machine) or **n**o action at all, shown in Figure 3 (b). Note that the waiter behaves nondeterministically, when the machine, having accepted a coin, idles she/he either apologizes to the customer or hits the machine. The customer submits an input **E**spresso-**p**lease or **M**oney after one of the outputs has been received from the coffee shop in response to the previous input. Such behavior is modeled by the LTS L_E shown in Figure 2.

The detailed behavior of each machine becomes clear from the LTS $L_C||L_B||L_E$, Figure 4 (a). The obtained LTS has no livelock, so a (composite) FSM over the external alphabets can be extracted. The composite FSM $C \Diamond B$ is shown in Figure 4 (b).



(a) (b)

Figure 4. The LTS $L_C||L_B||L_E$ (a) and the composite FSM $C \Diamond B$ (b)

Note that our definition of the composition operator on FSMs essentially differs from that used in the area of hardware, see for example, [Wat93], [Lin95], [Azi95]. Synchronous circuits are usually constructed in such way that several input signals to the same component are synchronized. It means

that in the synchronous case, for example, the FSM B in Figure 1 should have a compound input alphabet $X_2 \times U$ modeling such synchronization. In the asynchronous case, the input alphabet $X_2 \cup U$ corresponds to interleaved, i.e. asynchronous, communication considered in our setting. The two definitions of the operator come to the same only when each component has just one input. Moreover, to exclude oscillations, hardware components must participate in just a single interaction in response to external input. Opposed to the synchronous case, asynchronously communicating FSMs for different external input sequences may involve into a various number of internal interactions before one of them produces an external output. As a result, no external output might be produced when the system falls into livelock. Thus, the problem of livelocks becomes crucial for solving equations in the asynchronous case.

3. FSM DECOMPOSITION

3.1 Asynchronous FSM equations

The decomposition problem can be formulated by means of an equation in terms of FSMs, similar to that considered within process algebra. Given two FSMs, A over alphabets X and Y (the overall specification) and C over alphabets $X_1 \cup Z$ and $Y_1 \cup U$ (the context), we are required to determine an FSM B such that the composition operator $\Diamond$ is defined for the FSMs C and B and a relation, reduction or equivalence, holds for the FSMs $C \Diamond B$ and A. For nondeterministic FSMs, we have the following three (asynchronous) equations

$$C \Diamond W \leq A \ (1); \quad A \leq C \Diamond W \ (2); \quad C \Diamond W \cong A \ (3),$$

where W is the unknown FSM to be found (if possible). Intuitively, the equation, for example, (2) corresponds to the case where the system to be designed in given alphabets is allowed to do more, but no less than a given specification.

An FSM B over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ is said to be a *solution to the equation* (1), (2) or (3) if

- the composition operator is defined for the FSMs C and B and
- $C \Diamond B \leq A$, $A \leq C \Diamond B$, or $C \Diamond B \cong A$, respectively.

A solution B is called the *largest* solution to an equation if any solution to the equation is a reduction of B. Once the largest solution is found, one can then reduce it to a solution that is optimal according to a chosen criterion, for example, a machine with a minimal number of states and transitions [Dam94], [Wat94]. Similarly, one may define the smallest solution to an equation.

Generally speaking, one may treat each equation separately. On the other hand, given the largest solution to (1) and the smallest solution to (2), the product of these solutions is obviously a solution to (3). Hereinafter, we consider the equation $C \lozenge W \leq A$ only, as it seems most interesting for practice. Solving this equation, we solve, in fact, the FSM equation $C \lozenge W \cong A$ in the deterministic setting.

3.2 The largest potential solution to the equation $C \lozenge W \leq A$

Any potential solution to the equation $C \lozenge W \leq A$ is a reduction of the single-state chaos machine over the alphabets $X_2 \cup U$ and $Y_2 \cup Z$, denoted *Ch*. This observation suggests that one may solve the equation by deleting certain traces from the chaos machine. To this end, we first state the following obvious property of the largest solution.

Proposition. Let an FSM $M \leq Ch$ be the largest solution to the equation $C \lozenge W \leq A$. Then any FSM $B \leq Ch$ such that at least one of the three following conditions is satisfied

a) $C \lozenge B \leq A$ does not hold,
b) the LTS $L_C \| L_B \| L_E$ has an output deadlock,
c) the LTS $L_C \| L_B \| L_E$ has a livelock,

is not a reduction of P.

The statement indicates the cases when a certain trace of the chaos machine should be deleted to reduce it to the largest solution or at least to some solution may the largest solution not exist. We will proceed in two steps. We first define and construct a so-called largest potential solution by weakening the three conditions to the first two and then we check whether the composition operator is defined for the context and obtained machine (the condition c). The last step can be performed by constructing a composite machine, as explained in Section 3.1. This motivates the following definition.

An FSM $P \leq Ch$ is said to be the *largest potential solution* to the equation $C \lozenge W \leq A$ if any FSM $B \leq Ch$ such that

a) $C \lozenge B \leq A$ does not hold, or
b) the LTS $L_C \| L_B \| L_E$ has an output deadlock

is not a reduction of P.

By definition, the largest potential solution P composed with the context C has no non-conforming external behavior, but may yet have livelock and the composite machine might not be defined. In other words, the largest solution (if it exists) is a reduction of the largest potential solution, and the latter becomes the largest solution to the equation when the composition operator is defined. Then it remains to determine how the largest potential solution can be constructed.

We first present a scheme that generalizes existing procedures for equation solving in terms of the LTSs and the FSM (synchronous case) in trace semantics.

Step 1. Based on a chosen definition of the composition operator, derive a global machine describing all the sequences of actions that may occur in the composition.

Step 2. Construct the acceptor whose designated dead-state* accepts all the traces of the global machine which result in an external behavior not conforming to A (according to a given conformance relation).

Step 3. Construct the projection of the acceptor into the alphabets of solutions to the equation. Replace every state of the projection, comprising the dead-state, by the dead-state.

Step 4. Construct the complement of the obtained projection (and in case of an FSM equation, convert the obtained complement into an FSM).

These steps use various procedures depending on the model, conformance relation, and composition operator. Adapted to our setting, the scheme can be implemented in the following algorithm.

Algorithm 3.1. Construction of the largest potential solution to the equation $C \Diamond W \leq A$.

Input. The FSM A and context C.

Output. The largest potential solution $[[A,C]]$ (if it exists).

Step 1. Transform FSMs C and Ch into corresponding LTSs L_C and L_{Ch}. Construct the LTS $L_C \| L_{Ch} \| L_E$ where L_E is the environment LTS.

Step 2.

a) Transform the FSM A into the corresponding LTS L_A. Construct the LTS $L_{\hat{A}}$ by adding transitions to a designated dead-state labeled with all $y \in (Y \cup Y_1) \backslash out(s)$ from each state s of the LTS L_A such that $out(s) \cap (Y \cup Y_1) \neq \varnothing$.

b) Compose the LTSs $L_C \| L_{Ch} \| L_E$ and $L_{\hat{A}}$ and replace all global states of the composed LTS, containing a component dead-state, as well as all states from which no state s such that $out(s) \cap Y \neq \varnothing$ is reachable, by the (global) dead-state. Let $L_{A,C}$ be the obtained LTS.

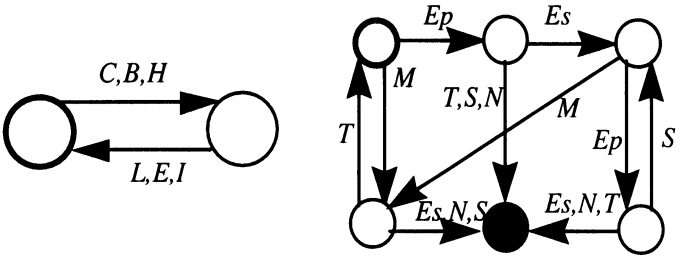
Step 3. Construct the $(X_2 \cup U \cup Y_2 \cup Z)$-projection of the LTS $L_{A,C}$ by a subset construction. Replace every state of the $(X_2 \cup U \cup Y_2 \cup Z)$-projection, which comprises a component dead-state, by a dead-state. Let $\hat{L}$ be the resulting LTS.

Step 4. Convert the LTS $\hat{L}$ into an FSM $[A,C]$. Complete the FSM $[A,C]$ by adding a *don't care* state and its incoming transitions from each state with an undefined transition for an input in $X_2 \cup U$ and label them with a corresponding input and each output in $Y_2 \cup Z$. Delete the dead-state with its

* The dead state in an LTS has no transitions, but in an FSM, it has self-loops labeled with each I/O pair over the given alphabets.

incoming transitions and consecutively all states whose transition becomes undefined for some input until the initial state is deleted or no more state can be deleted. In the first case, the equation has no solution and in the second case, the largest potential solution $[[A,C]]$ is returned.

We immediately notice that since the LTS $L_{\hat{A}}$ is trace included into the LTS L_E, one can directly construct $L_C||L_{Ch}||L_{\hat{A}}$ instead of $L_C||L_{Ch}||L_E||L_{\hat{A}}$. We first illustrate the algorithm using our example considered above and then prove its validity.



(a) (b)

Figure 5. The LTSs L_C (a) and $L_{\hat{A}}$ (b)

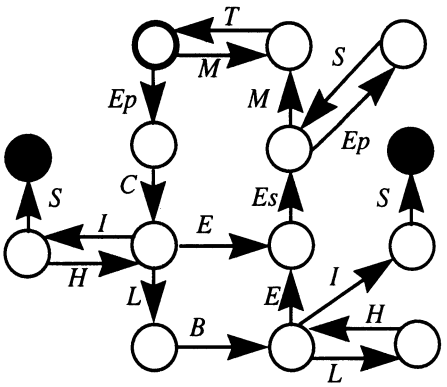
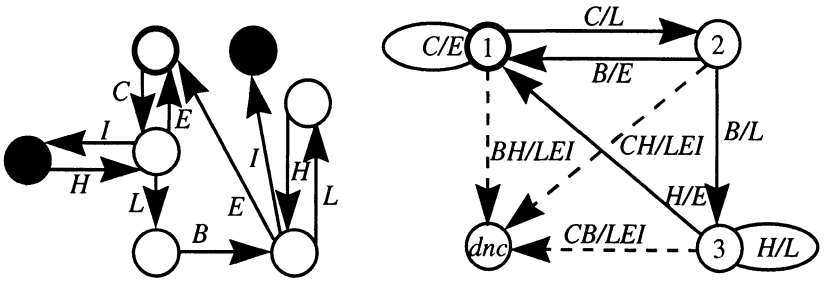


Figure 6. The LTS $L_{A,C}$ obtained at Step 2

Example. We apply the above algorithm to construct a coffee machine which can be used in the coffee shop, the FSM A (Figure 4b) with the waiter, the FSM C (Figure 3b). Figure 5 shows the LTS corresponding to the chaos machine over the alphabets of the coffee machine and the LTS $L_{\hat{A}}$, where a black hole represents a dead-state. Figure 6 shows the LTS $L_{A,C}$ obtained at Step 2, where black holes represent a global dead-state. Figure 7 shows the last intermediate result, the LTS $\widehat{L}$ obtained at Step 3 (a) and the

FSM [[*A*,*C*]] (b), where dashed lines represent don't care transitions to a don't care state. The largest potential solution [[*A*,*C*]] is a nondeterministic machine. Based on this machine, one can construct a few (deterministic) coffee machines, including the simplest machine which dispenses a cup of espresso once a coin is inserted.



(a) (b)

Figure 7. The LTS $\widehat{L}$ obtained at Step 3 (a) and the FSM [[*A*,*C*]] (b)

Theorem 1. Given two FSMs, A over alphabets $X_1 \cup X_2$ and $Y_1 \cup Y_2$ and C over alphabets $X_1 \cup Z$ and $Y_1 \cup U$, if Algorithm 3.1 does not return an FSM the equation $C \Diamond W \leq A$ has no solution, and if the composition operator is defined for the FSMs C and [[*A*,*C*]] derived by Algorithm 3.1, then the FSM [[*A*,*C*]] is the largest solution to $C \Diamond W \leq A$. Moreover, if $C \Diamond$[[*A*,*C*]] is not equivalent to A then the equation $C \Diamond W \cong A$ has no solution.

Proof. According to Algorithm 3.1, a trace $\beta\alpha\gamma$ over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ is not a trace of the resulting FSM [[*A*,*C*]] if and only if one of the following conditions holds.

(1) There exists a trace of the LTS $L_{A,C}$ (Step 2b) with the $(X_2 \cup U \cup Y_2 \cup Z)$-projection $\beta\alpha\gamma$ resulting in the external behavior different from that of A.

(2) There exists a trace of the LTS $L_{A,C}$ (Step 2b) with the $(X_2 \cup U \cup Y_2 \cup Z)$-projection $\beta\alpha\gamma$ resulting in an output deadlock of the composed system (Step 2).

(3) Trace $\beta\alpha\gamma$ cannot be a trace of a potential solution (Step 4), because there exists $\nu \in X_2 \cup U$ such that any trace $\beta\nu\alpha\gamma\delta$ over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ takes the LTS $\widehat{L}$ obtained at Step 3 to the dead-state.

If Algorithm 3.1 does not return an FSM then there exists a sequence $\beta \in U^*$ such that for each trace $\beta\alpha\gamma$ over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ one of the above conditions 1, 2, or 3 holds while each FSM B over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ has a trace $\beta\alpha\gamma$ for an appropriate $\gamma \in U^*$. If B has trace $\beta\alpha\gamma$ such that condition 3 holds then there exists its prolongation $\beta\nu$ such that for each trace $\beta\nu\alpha\gamma\delta$ over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ the condition 1 or 2 holds. In other words, without loss of generality, we assume that B has a trace $\beta\alpha\gamma$

for which the condition 1 or 2 holds. In this case, there exists an external input sequence α such that the composition of the context C and FSM B exhibits the behavior different from that of the FSM A or has an output deadlock when α is submitted, i.e. B cannot be a solution to the equation $C \Diamond W \leq A$ and therefore cannot be a solution to the equation $C \Diamond W \cong A$.

Suppose now that Algorithm 3.1 returns an FSM $[[A,C]]$. Trace $\beta\alpha\gamma$ is a trace of $[[A,C]]$ if and only if none of the above conditions 1, 2 and 3 holds, i.e. only a reduction of $[[A,C]]$ may be a solution to the equations $C \Diamond W \leq A$ and $C \Diamond W \cong A$. Thus, if the composition operator is defined for the FSMs C and $[[A,C]]$ derived by Algorithm 3.1, but $C \Diamond [[A,C]] \cong A$ does not hold then the equation $C \Diamond W \cong A$ has no solution. On the other hand, any reduction of a solution to the equation $C \Diamond W \leq A$ is also a solution, i.e. if the composition operator is defined for the FSMs C and $[[A,C]]$ derived by Algorithm 3.1 then $[[A,C]]$ is the largest solution to the equation $C \Diamond W \leq A$.

❑

If the machine $[[A,C]]$ is not a solution to the equation $C \Diamond [[A,C]] \cong A$ then the question as to whether there exists a solution to both equations remains open. This is the case in our example, where the composition operator is not defined for the FSMs C and $[[A,C]]$ due to livelock. The problem is that the waiter may repeatedly hit the coffee machine which continues to light the lamp, so the customer may no longer see the waiter (state 3 in Figure 7b).

Concluding this section, we notice that the largest solution to the equation $C \Diamond W \leq A$ may not exist even when the equation has a largest potential solution. Consider the FSMs A and C in Figure 8 (a) and (b). A solution to the equation is to be found in the alphabets $U = \{u_1, u_2\}$ and $Z = \{z_1, z_2\}$, as $X_2 = Y_2 = \varnothing$.

The LTS $L_{A,C}$, obtained at the second step of Algorithm 3.1, has no dead-state (Figure 9). Therefore, the largest potential solution $[[A,C]]$ is a single-state chaos machine over the alphabets $\{u_1, u_2\}$ and $\{z_1, z_2\}$, however, it is not a solution to the equation $C \Diamond W \leq A$. At the same time, each FSM trace over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ is a trace of some solution to the equation. The FSM $[[A,C]]$ is not a solution to the equation and cannot be reduced without losing some solutions. In fact, by direct inspection of the LTS in Figure 9, one can verify that an FSM B is a solution if and only if the set of traces of B includes the regular set

$L = [u_2(z_1u_1)^*z_2 \vee (u_1z_2)^*u_1z_1]^*$

Apparently, there exists a number of such FSMs. Thus, for each trace $\beta\alpha\gamma \in L$ there exists a solution to the equation $C \Diamond W \leq A$ with this trace. Consider now trace $\beta\alpha\gamma$ over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ such that $\beta\alpha\gamma \notin L$. Then, by definition of L, the maximal prefix of $\beta\alpha\gamma$ that is in the set L is the empty sequence or is terminated with an appropriate $z \in Z$, i.e. the regular set

$L \cup \{\beta \square \gamma\}$ also is in the set of traces of an appropriate FSM that is a solution to the equation.

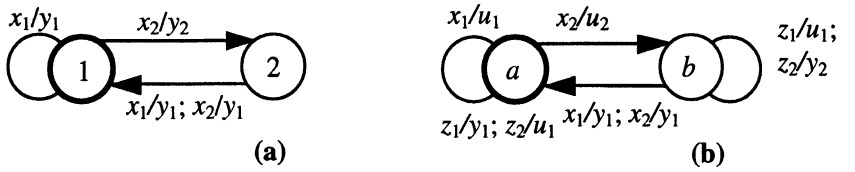


Figure 8. The FSMs A (a) and C (b)

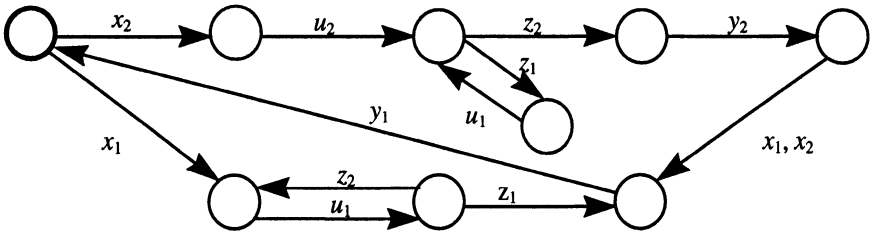


Figure 9. The LTS $L_{A,C}$

4. AVOIDING LIVELOCKS

As discussed in the previous section, the largest potential solution is not always a solution to the equation due to livelocks. Livelock corresponds to a situation when the component and its context in response to external input involve into unbounded internal interactions. To avoid livelocks we may thus bound the maximal number of internal actions executed by the system in response to any external input. One may also impose such a restriction attempting at finding a solution such that the response time of the overall system does not exceed a certain limit.

We have to find a solution to the equation $C \Diamond W \leq A$ such that in the resulting composition, the number of internal actions successively executed between any external input and a resulting output never exceeds a given threshold l, $l \geq 1$. We first introduce the l-restricted composition operator $\Diamond_l$. Given two FSMs C and B, the FSM $C \Diamond_l B$ coincides with the FSM $C \Diamond B$ if the LTS $L_C \| L_B \| L_E$ has no path labeled with more than l consecutive internal actions. Otherwise, i.e. when there exists a path with more than l consecutive internal actions, the operator $\Diamond_l$ is not defined for the FSMs C and B. Next, we demonstrate how the results of Section 3 can be further extended to solve the equation $C \Diamond_l W \leq A$.

Given two FSMs, A over alphabets $X=X_1 \cup X_2$ and $Y=Y_1 \cup Y_2$ and C over alphabets $X_1 \cup Z$ and $Y_1 \cup U$, an FSM B over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ is said to be *a solution to the equation* $C \Diamond_l W \leq A$ if the l-restricted composition operator is defined for the FSMs C and B and $C \Diamond_l B \leq A$.

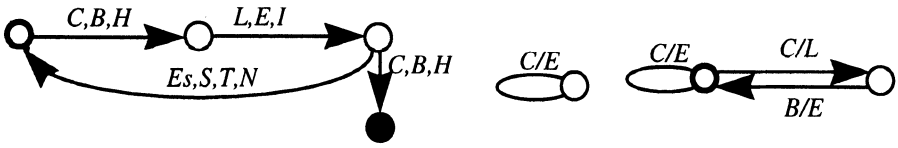
The idea behind our approach to solving such an equation is to replace a total chaos system L_{Ch} over alphabets $X_2 \cup U$ and $Y_2 \cup Z$ by a so-called *l-restricted* chaos system $L_{l\text{-}Ch}$ that can be obtained from the LTS L_{Ch} by redirecting internal traces of length exceeding l to a designated dead-state. Since the appearance of external output terminates any internal trace we add at appropriate states of the LTS $L_{l\text{-}Ch}$ corresponding transitions resetting the machine back to the initial state. Figure 10 (a) gives an example of an l-restricted chaos system $L_{l\text{-}Ch}$ for the coffee shop, where l=2.

Replacing the LTS L_{Ch} by the LTS $L_{l\text{-}Ch}$, we can use Algorithm 3.1 to find a solution. A small adjustment is required, though. In Step 4, if at state s of the LTS $\widehat{L}$ there is a transition labeled with $a \in U$ to dead-state then in the corresponding FSM, denoted $[A,C,l]$, there is a transition from s to dead-state labeled with a/b for each $b \in Y_2 \cup Z$. Such transitions may exist since the l-restricted chaos system has transitions to dead-state labeled with internal inputs. We denote $[[A,C,l]]$ the resulting machine.

Theorem 2. Given two FSMs, A and C, and an integer $l \geq 1$, if Algorithm 3.1 does not return an FSM then the equation $C \Diamond_l B \leq A$ has no solution. If it results in an FSM $[[A,C,l]]$ then $[[A,C,l]]$ is the largest solution to the equation $C \Diamond_l W \leq A$.

The proof of the theorem is similar to that of Theorem 1. As the FSMs C and $[[A,C,l]]$ may execute at most l internal actions before producing an external output, the composition operator for the FSMs C and $[[A,C,l]]$ is always defined; thus, the FSM $[[A,C,l]]$ is the largest solution to the equation $C \Diamond_l W \leq A$.

Example. For the coffee shop, we solve the equation $C \Diamond_l W \leq A$ for l=2 and 4. Figure 10 shows the FSM $[[A,C,2]]$ (b) and the FSM $[[A,C,4]]$ (c). Note that for l=3, no solution can be found. The submodule has only internal inputs and outputs, so the number of internal interactions is always even.



(a) (b) (c)

Figure 10. The chaos system $L_{2\text{-}Ch}$ (a), the FSMs $[[A,C,2]]$ (b) and $[[A,C,4]]$ (c)

5. CONCLUSION

We considered the problem of equation solving for asynchronously communicating nondeterministic finite state machines. It was demonstrated that opposed to the case of labeled transition systems, equation solving is complicated by potential livelocks arising when two machines interact via feedback. Due to livelocks, a given equation may have no solution. By the same reason, asynchronous equations differ also from synchronous ones considered in the context of sequential circuit optimization. An algorithm for finding the largest potential solution to the asynchronous equation for trace inclusion, i.e. reduction relation, was presented. The largest potential solution is the largest solution when the resulting composition has no livelock. We also demonstrated that the largest solution may sometimes not exist. To resolve the problem of livelocks we proposed to use a special chaos system that prevents the composition from falling into livelock by imposing a predefined limit on the possible number of interactions inside the system. This approach could also be useful when one needs to decompose a given system and to keep its response time within a certain bound. The results of this paper could be used in a number of applications, such as protocol design, controller design, system validation, and test derivation for embedded testing.

Future work in this direction is to generalize the presented approach to incompletely specified nondeterministic finite state machines. It would also be interesting to study the relation between the complexity of a solution (e.g. in terms of the number of transitions) and the number of internal interactions. Whether the simplest submodule also needs a minimal number of internal interactions in the resulting composition (the system with a minimal response time) remains an open question.

ACKNOWLEDGMENTS

The first author acknowledges the support of the NSERC grant OGP0194381. The authors would like to thank G. v. Bochmann for stimulating discussions on the problem of submodule construction.

REFERENCES

[Azi95] 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, Belgium, pp. 279-292, 1995.

[Cec86] F. Cecseg, *Products of Automata*, Monographs in Theoretical Computer Science. Springer Verlag, 1986.

[Che96] S. C. Cheung and J. Kramer. *Context constraints for compositional reachability analysis*, ACM Trans. on Software Engineering and Methodology, Vol. 5, N 4, 1996, pp. 334-377.

[Dam94] M. Damiani, *Nondeterministic finite state machines and sequential don't cares*, Proceedings of the Euro Design and Test Conference, 1994.

[Hee95] L. Heerink and E. Brinksma, Validation in context, *Proceedings of the 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing, and Verification,* Chapman & Hall (1995), pp 221-236.

[Hop79] J. E. Hopcroft and J. D. Ullman (1979) *Introduction to automata theory, languages, and computation*, Addison-Wesley, New York.

[Kim72] J. Kim and M. Newborn. *The simplification of sequential machines with input restrictions*, IEEE Trans. on Computers, C-20, pp. 1440-1443, 1972.

[Lar90] K. Larsen and L. Xinxin, *Compositionality through an operational semantics of contexts*, Lecture Notes in Computer Science, Vol 443, 1990.

[Lin95] B. Lin, G. de Jong, and T. Kolks, *Hierarchical optimization of asynchronous circuits*, Proceedings of the 32nd Design Automation Conference, 1995, pp. 712-717.

[Lee93] D. Lee, K. Sabnani, D. Krispot, S. Paul, and M. Uyar, *Conformance testing of protocols specified as communicating fsms*, INFOCOM'93.

[Mer83] 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, Jan. 1983, pp. 1-25.

[Qin91] H. Qin and P. Lewis, *Factorisation of Finite State Machines under Strong and Observational Equivalences*, Journal of Formal Aspects of Computing, Vol. 3, July-Sept. 1991, pp. 284-307.

[Par89] J. Parrow. *Submodule construction as equation solving in ccs,* Theoretical Computer Science, 68, pp. 175-202, 1989.

[Pet93] 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, pp. 363-378, 1993.

[Pet94] A. Petrenko, N. Yevtushenko, and R. Dssouli, *Testing strategies for communicating FSMs,* Proceedings of the 7th IWTCS, (1994) pp. 193-208.

[Pet96] A. Petrenko, N. Yevtushenko, G. v. Bochmann, and R. Dssouli. *Testing in context: framework and test derivation*, Computer Communications, Vol. 19, pp. 1236-1249, 1996.

[Rho91] J.-K. Rho, G. Hachtel, and F. Somentzi. *Don't care sequences and the optimization of interacting finite state machines*, Proceedings of the IEEE Conference on Computer-Aided Design, Santa Clara, 1991, pp. 414-421.

[Sta72] P. H. Starke, *Abstract automata*, American Elsevier Publishing Company, Inc. New York, 1972.

[Wat93] Y. Watanabe, and R. K. Brayton. *The maximal set of permissible behaviors for fsm networks*, Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 316-320, 1993.

[Wat94] Y. Watanabe and R. K. Brayton. *State minimization of pseudo nondeterministic FSMs*, Proceedings of Euro Design and Test Conference, pp. 184-191, 1994.

[Yev91] N. Yevtushenko and A. Matrosova. *On one approach to automata networks checking sequences construction*, Automatic Control and Computer Sciences, Allerton Press, NY, Vol. 25, N 2, pp. 3-7, 1991.