Solution of Parallel Language Equations for Logic Synthesis

Nina Yevtushenko¶ Tiziano Villa§ Robert K. Brayton† Alex Petrenko‡

Alberto L. Sangiovanni-Vincentelli†

¶Dept. of EECS, Tomsk State University, Tomsk, 634050, Russia

§PARADES, Via di S.Pantaleo, 66, 00186 Roma, Italy

†Dept. of EECS, Univ. of California, Berkeley, CA 94720

‡CRIM, 550 Sherbrooke West, Montreal, H3A 1B9, Can

Abstract

The problem of designing a component that combined with a known part of a system, conforms to a given overall specification arises in several applications ranging from logic synthesis to the design of discrete controllers. We cast the problem as solving abstract equations over languages. Language equations can be defined with respect to several language composition operators such as synchronous composition, $\bullet$, and parallel composition, $\diamond$; conformity can be checked by language containment. In this paper we address parallel language equations.

Parallel composition arises in the context of modeling delay-insensitive processes and their environments. The parallel composition operator models an exchange protocol by which an input is followed by an output after a finite exchange of internal signals. It abstracts a system with two components with a single message in transit, such that at each instance either the components exchange messages or one of them communicates with its environment, which submits the next external input to the system only after the system has produced an external output in response to the previous input.

We study the most general solutions of the language equation $A \diamond X \subseteq C$, and define the language operators needed to express them. Then we specialize such equations to languages associated with important classes of automata used for modeling systems, e.g., regular languages and FSM languages. In particular, for $A \diamond X \subseteq C$, we give algorithms for computing: the largest FSM language solution, the largest complete solution, and the largest solution whose composition with A yields a complete FSM language. We solve also FSM equations under bounded parallel composition.

In this paper, we give concrete algorithms for computing such solutions, and state and prove their correctness.

1 Introduction

An important step in the design of complex systems is the decomposition of the system into separate components which interact in some well-defined way. A typical problem is how to design a component that combined with a known part of a system, called the context, conforms to a given overall specification. This arises in several applications ranging from logic synthesis to the design of discrete controllers. There are three key issues to consider: how to model the system, its components and specification, how to model the composition of the components, and how to model the notion of a system conforming.

In general, one can define equations over languages associated with the components of a given system and check conformity by language containment. Two composition operators for abstract languages are: synchronous , $\bullet$, and parallel, $\diamond$, leading to the synchronous equation $A \bullet X \subseteq C$ and the parallel equation $A \diamond X \subseteq C$. The theory of equations over languages can be specialized further to languages associated with interesting classes of automata, such as finite automata (FAs) and finite state machines (FSMs). Then the goal is to characterize the solutions of the equations over FSMs $M_A \bullet M_X \subseteq M_C$ and $M_A \diamond M_X \subseteq M_C$, where M_A models the context, M_C the specification and M_X is an unknown FSM.

In this paper we present a theory for parallel language equations. In particular we show how to compute the largest language solution of $A \diamond X \subseteq C$, the largest FSM language, the largest complete solution, and the largest solution whose composition with the context A yields a complete FSM language. We solve also equations under bounded parallel composition.

For references on parallel equations see [8] on "asynchronous equations". In process algebra parallel equations have been solved for processes, which are described by prefix-closed regular languages, over various conformance relations [7, 9, 6]. In this paper we introduce parallel equations over arbitrary languages under language containment, and propose effective procedures for regular languages and languages of FSMs. Some procedures in this paper are based on those proposed in [8] for "asynchronous equations". Solution of synchronous equations in the context of logic synthesis are surveyed in [4], Chap. 6.

2 Equations over Languages

2.1 Languages

Definition 2.1 *An alphabet is a finite set of symbols. The set of all strings over a fixed alphabet X is denoted by $X^\star$. $X^\star$ includes the empty string ϵ. A subset $L \subseteq X^\star$ is called a* **language** *over alphabet X.*

We introduce some operations on languages. It is useful to recall the notions of substitution and homomorphism of languages [3]. A **substitution** f is a mapping of an alphabet Σ onto subsets of $\Delta^\star$ for some alphabet Δ. The substitution f is extended to strings by setting $f(\epsilon) = \epsilon$ and $f(xa) = f(x)f(a)$. An **homomorphism** h is a substitution such that $h(a)$ is a single string for each symbol a in the alphabet Σ.

1. Given a language L over alphabet $X \cup V$, consider the

homomorphism h defined as

$$h(x) = \begin{cases} x & \text{if } x \in V \\ \epsilon & \text{if } x \in X \setminus V \end{cases},$$

then

$$L_{\Downarrow V} = \{h(\alpha) \mid \alpha \in L\}$$

over alphabet V is the **restriction** of L to the alphabet V, or V-restriction of L, i.e., words in $L_{\Downarrow V}$ are obtained from those in L by deleting all the symbols in X that are not symbols in V. By definition of substitution $h(\epsilon) = \epsilon$.

2. Given a language L over alphabet X and an alphabet V disjoint from X, consider the mapping f defined as

$$f(x) = \alpha x \beta,\ \ \forall x \in X, \forall \alpha, \beta \in V^\star,$$

then

$$L_{\Uparrow V} = \{f(x) \mid x \in L\}$$

over alphabet $X \cup V$ is the **expansion** of L over alphabet V, or V-expansion of L, i.e. words in $L_{\Uparrow V}$ are obtained from those in L by inserting anywhere in them words from $V^\star$. Notice that f is not a substitution and that $f(\epsilon) = \{\alpha \mid \alpha \in V^\star\}$.

Given a language L over alphabet X, an alphabet V disjoint from X and a natural number l, consider the mapping f defined as

$$f(x) = \alpha x \beta,\ \ \forall x \in X, \forall \alpha, \beta \in V^{\leq l},$$

then the language

$$L_{\Uparrow (V,l)} = \{f(\alpha) \mid \alpha \in L\}$$

over alphabet $X \cup V$ is the **l-bounded expansion** of language L over alphabet V, or (V, l)-expansion of L, i.e. words in $L_{\Uparrow V}$ are obtained from those in L by inserting anywhere in them words from $V^{\leq l}$. Notice that f is not a substitution and that $f(\epsilon) = \{\alpha \mid \alpha \in V^{\leq l}\}$.

The following straightforward fact holds between the restriction and expansion operators.

Proposition 2.1 *Given alphabets X and Y, a language L over alphabet X and a string $\alpha \in (X \cup Y)^\star$, then $\alpha_{\Downarrow X} \in L$ iff $\alpha \in L_{\Uparrow Y}$.*

We introduce some classes of languages used later in the paper.

Definition 2.2 *Given a language L over alphabet X, the language of all prefixes of words in L is* $\mathrm{Init}(L) = \{x \in X^\star \mid \exists y \in X^\star, xy \in L\}$. *$L$ over alphabet X is* **prefix-closed** *if $\forall \alpha \in X^\star\ \forall x \in X\ [\alpha x \in L \Rightarrow \alpha \in L]$. Equivalently, L is prefix-closed iff $L = Init(L)$.*

Definition 2.3 *$L \subseteq (IO)^\star$ over alphabet $I \cup O$ is IO-***prefix-closed** *if $\forall \alpha \in (IO)^\star\ \forall io \in IO\ [\alpha\, io \in L \Rightarrow \alpha \in L]$.*

Definition 2.4 *$L \subseteq (IO)^\star$ over alphabet $I \cup O$ is IO-***progressive** *if $\forall \alpha \in (IO)^\star\ \forall i \in I\ \exists o \in O\ [\alpha \in L \Rightarrow \alpha\, io \in L]$.*

Definition 2.5 *$L \subseteq (I(U \cup V)^\star O)^\star$ over alphabet $I \cup U \cup V \cup O$ is $I^\star O$-***progressive** *if $\forall \alpha \in (I(U \cup V)^\star O)^\star\ \forall i \in I\ \forall \beta \in (U \cup V)^\star\ \exists \gamma \in (U \cup V)^\star\ \exists o \in O\ [[\alpha \in L \Rightarrow \alpha\ i \gamma o \in L] \wedge [\alpha i \beta \in L \Rightarrow \alpha i \beta\ \gamma o \in L]]$.*

Definition 2.6 *L over alphabet $I \cup O$ is $I_{\Downarrow}$-***defined** *if $L_{\Downarrow I} = I^\star$.*

An $I^\star O$-progressive language is IO-progressive, which in turn is $I_{\Downarrow}$-defined. The converse of the two implications does not hold.

Definition 2.7 *L over alphabet $X \cup U$ (X and U disjoint) is U-***convergent** *if $\forall \alpha \in X^\star$ the language $\alpha_{\Uparrow U} \cap L$ is finite, otherwise it is U-***divergent**.

Example 2.1 *$L = \{iu^\star o\}$ where $X = \{i, o\}$ and $U = \{u\}$ is U-divergent, as witnessed by the string $io \in X$ whose expansion includes the infinite set $\{iu^\star o\}$ coinciding with L.*

2.2 Composition of Languages

Consider two systems A and B with associated languages $L(A)$ and $L(B)$. The systems communicate with each other by a channel U and with the environment by channels I and O. We introduce a composition operator that describes the external behaviour of the composition of $L(A)$ and $L(B)$.

Definition 2.8 *Given alphabets I, U, O, language L_1 over $I \cup U$ and language L_2 over $U \cup O$, the* **parallel composition** *of languages L_1 and L_2 is the language $[(L_1)_{\Uparrow O} \cap (L_2)_{\Uparrow I}]_{\Downarrow I \cup O}$, denoted by $L_1 \diamond L_2$, defined over $I \cup O$.*

*Given alphabets I, U, O, language L_1 over $I \cup U$ and language L_2 over $U \cup O$, the l-***bounded parallel composition** *of languages L_1 and L_2 is the language $[(L_1)_{\Uparrow O} \cap (L_2)_{\Uparrow I} \cap (I \cup O)^\star_{\Uparrow (U,l)}]_{\Downarrow I \cup O}$, denoted by $L_1 \diamond_l L_2$, defined over $I \cup O$.*

When $l = \infty$ the definition of l-bounded parallel composition reduces to the definition of parallel composition of languages, because then $(I \cup O)^\star_{\Uparrow (U,l)}$ becomes $(I \cup O \cup U)^\star$, that is the universe over $I \cup O \cup U$, and so it can be dropped from the conjunction.

In the sequel it will be useful to consider the notion of U-convergence extended to the composition of two languages.

Definition 2.9 *Given a language A over alphabet $I \cup U$, a language B over alphabet $U \cup O$ is A-***compositionally** *U-***convergent** *if the language $L = A_{\Uparrow O} \cap B_{\Uparrow I}$ over alphabet $X = (I \cup O) \cup U$ is U-convergent, i.e., $\forall \alpha \in (I \cup O)^\star$ the language $\alpha_{\Uparrow U} \cap L$ is finite.*

When clear from the context, instead of *A-compositionally* we will write more simply *compositionally*.

2.3 Solution of Equations over Languages

2.3.1 Language Equations under Parallel Composition

Given pairwise disjoint alphabets I, U, O, languages A over $I \cup U$ and C over $I \cup O$, consider

$$A \diamond X \subseteq C. \quad (1)$$

Definition 2.10 *Given pairwise disjoint alphabets I, U, O, a language A over alphabet $I \cup U$ and a language C over alphabet $I \cup O$, language B over alphabet $U \cup O$ is called a* **solution** *of the equation $A \diamond X \subseteq C$ iff $B \neq \emptyset$ and $A \diamond B \subseteq C$. The* **largest solution** *is a solution that contains any other solution.*

Theorem 2.1 *The largest solution of the equation $A \diamond X \subseteq C$ is the language $S = \overline{A \diamond \overline{C}}$, if $S \neq \emptyset$.*

Proof. Consider a string $\alpha \in (U \cup O)^\star$, then α is in the largest solution of $A \diamond X \subseteq C$ iff $A \diamond \{\alpha\} \subseteq C$ and the following chain of equivalences follows:

$$\begin{aligned}
A \diamond \{\alpha\} \subseteq C & \quad \Leftrightarrow \\
(A_{\Uparrow O} \cap \{\alpha\}_{\Uparrow I})_{\Downarrow I \cup O} \cap \overline{C} = \emptyset & \quad \Leftrightarrow \\
A_{\Uparrow O} \cap \{\alpha\}_{\Uparrow I} \cap \overline{C}_{\Uparrow U} = \emptyset & \quad \Leftrightarrow \\
\alpha \notin (A_{\Uparrow O} \cap \overline{C}_{\Uparrow U})_{\Downarrow U \cup O} & \quad \Leftrightarrow \\
\alpha \in \overline{(A_{\Uparrow O} \cap \overline{C}_{\Uparrow U})_{\Downarrow U \cup O}} & \quad \Leftrightarrow \\
\alpha \in \overline{A \diamond \overline{C}} &
\end{aligned}$$

Therefore the largest solution of the language equation $A \diamond X \subseteq C$ is given by the language

$$S = \overline{A \diamond \overline{C}}, \tag{2}$$

if $S \neq \emptyset$. **QED**

Corollary 2.1 *A language $B \neq \emptyset$ over alphabet $U \cup O$ is a solution of $A \diamond X \subseteq C$ iff $B \subseteq \overline{A \diamond \overline{C}}$.*
If $\overline{A \diamond \overline{C}} = \emptyset$, then the language equation $A \diamond X \subseteq C$ has no solution.

Proposition 2.2 *If S is U-convergent, then S is the largest U-convergent solution of the equation, and a language $B \neq \emptyset$ is an U-convergent solution iff $B \subseteq S$.*

When S is not U-convergent the largest U-convergent solution does not exist, and any finite subset of S is an U-convergent solution. An analogous proposition and remark hold for S compositionally U-convergent solutions.

Theorem 2.2 *The largest solution of the equation*

$$A \diamond_l X \subseteq C$$

is the language

$$S = \overline{(A_{\Uparrow O} \cap \overline{C}_{\Uparrow (U,l)})_{\Downarrow U \cup O}},$$

if $S \neq \emptyset$.

2.4 Equations over Mathematical Machines

Language equations can be effectively solved when they are defined over languages that can be manipulated algorithmically. Usually such languages are presented through their corresponding automata. We are going to study equations over various classes of automata, like FAs and FSMs, specializing the equations to their associated languages. A key issue is the closure of the solution set with respect to a given type of language, e.g., when dealing with FSM language equations we require that the solutions are FSM languages too. This cannot be taken for granted, since the general solution is expressed through the operators of complementation and composition, which do not necessarily preserve some classes of languages.

3 Equations over Finite Automata

3.1 Finite Automata and Regular Expressions

Definition 3.1 *A* **non-deterministic finite automaton** *(NDFA or more simply FA) is defined as a 5-tuple $F_A = \langle S, \Sigma, \Delta, r, F\rangle$. S represents the finite state space, Σ represents the finite alphabet, and $\Delta \subseteq \Sigma \times S \times S$ is the next state relation, such that $n \in S$ is a next state of present state $p \in S$ on symbol $i \in \Sigma$ if and only if $(i, p, n) \in \Delta$. The initial or reset state is $r \in S$ and $F \subseteq S$ is the set of final or accepting states. A variant of NDFAs allows the introduction of ϵ-moves, meaning that $\Delta \subseteq (\Sigma \cup \epsilon) \times S \times S$.*

The next state relation can be extended to have as argument strings in Σ^* (i.e., $\Delta \subseteq \Sigma^* \times S \times S$) as follows: $(\rho i, s, s'') \in \Delta$ if and only if there exists $s' \in S$ such that $(\rho, s, s') \in \Delta$ and $(i, s', s'') \in \Delta$.

A string x is said to be **accepted** *by the NDFA F_A if there exists a sequence of transitions corresponding to x such that $\Delta(x, r)$ contains a state in F. The* **language** *accepted by F_A, designated $\mathcal{L}(F_A)$, is the set of strings $\{x \,|\Delta(x, r) \cap F \neq \emptyset\}$.*

If for each present state p and symbol i there is at least next state n such that $(i, p, n) \in \Delta$ the NDFA is said to be **complete**.

A NDFA is a **deterministic finite automaton** *(DFA) if for each present state p and symbol i there is exactly one next state n such that $(i, p, n) \in \Delta$. The relation Δ can be replaced by the next state function δ, defined as $\delta : \Sigma \times S \to S$, where $n \in S$ is the next state of present state $p \in S$ on symbol $i \in \Sigma$ if and only if $n = \delta(i, p)$.*

A string x is said to be **accepted** *by the DFA F_A if $\delta(x, r) \in F$. The* **language** *accepted by F_A, designated $\mathcal{L}(F_A)$, is the set of strings $\{x \,|\delta(x, r) \in F\}$.*

The languages associated to finite automata are the regular languages and they are defined by means of regular expressions [3].

Definition 3.2 *The sets denoted by regular expressions are the* **regular languages**.

Regular languages are closed under union, concatenation, complementation and intersection. They are closed also under restriction, because they are closed under substitution [3]. We will show that they are closed under expansion, by providing an algorithm that, given the finite automaton of a language, returns the finite automaton of the expanded language.

3.2 Solution over Regular Languages

Non-deterministic finite automata are equivalent to deterministic ones and regular expressions are equivalent to finite automata [3]. By applying the algorithm of subset construction, one converts a NDFA into an equivalent complete DFA. Given an NDFA $F_A = \langle S, \Sigma, \Delta, r, F\rangle$, the process of subset construction builds the DFA $F_B = \langle 2^S, \Sigma, \delta, r, F_B\rangle$, where 1) the states $\tilde{s} \in 2^S$ are the subsets of S, 2) the transition relation is $\delta(i, \tilde{s}) = \cup_{s \in \tilde{s}}\{s' \mid (i, s, s') \in \Delta\}$ and 3) a state is final, i.e., $\tilde{s} \in F_B \subseteq 2^F$, iff $\tilde{s} \cap F \neq \emptyset$. Since many of the states in 2^S are unreachable from the initial state, they can be deleted. Thus, the determinized automaton usually has fewer states than the power set. To make a NDFA complete, it is not necessary to apply the full-blown subset construction; it suffices to add a new non-accepting state s_d whose incoming transitions are

(i, s, s_d) for all i, s for which there was no transition in the original automaton. By a closure construction [3], an NDFA with ϵ-moves can be converted to an NDFA without ϵ-moves; subset construction must be applied at the end to determinize it.

The equivalence of regular expressions and finite automata is shown by matching each operation on regular expressions with a constructive procedure that yields the finite automaton of the result, given the finite automata of the operands. For the most common operations (union, concatenation, complementation, intersection) see [3]. Here we sketch the procedures for restriction and expansion:

restriction ($\Downarrow$) Given FA F_A that accepts language L over $X \cup V$, the FA F'_A that accepts language $L_{\Downarrow V}$ over V is obtained by the following procedure:

1. $\forall x \in X \setminus V$, change every edge (x, s, s') into the edge (ϵ, s, s'), i.e., replace the symbols $x \in X \setminus V$ by ϵ.
2. Apply the closure construction to obtain an equivalent finite automaton without ϵ-moves and then apply subset construction to determinize it (even if the original FA F_A was deterministic, the process of adding ϵ-moves and then removing them by closure construction usually returns a NDFA).

expansion ($\Uparrow$) Given FA F_A that accepts language L over X, the FA F'_A that accepts language $L_{\Uparrow V}$ over $X \cup V$ ($X \cap V = \emptyset$) is obtained by adding for each state s, $\forall v \in V$, the edge (self-loop) (v, s, s).

l-expansion ($\Uparrow_l$) Given FA F_A that accepts language L over X, the FA F'_A that accepts language $L_{\Uparrow(V,l)}$, l integer, over $X \cup V$ ($X \cap V = \emptyset$) is obtained as follows:

1. The set of states S' of F'_A is given by
$$S' = S \cup \bigcup_{s \in S} \{s^1, \ldots, s^l\}.$$
2. The next state relation Δ' of F'_A is given by
$$\Delta' = \bigcup_{\substack{s \in S \\ v \in V}} \{(s, s^1, v)\} \cup \bigcup_{\substack{s \in S \\ 1 \leq j < l \\ v \in V}} \{(s^j, s^{j+1}, v)\} \cup \bigcup_{\substack{s, s' \in S \\ 1 \leq j \leq l \\ (s, s', x) \in \Delta}} \{(s^j, s', x)\}.$$
3. $r' = r$ and $F' = F$.

The procedure for restriction guarantees the substitution property $f(\epsilon) = \epsilon$.

Theorem 3.1 *There is an effective way to solve equations over regular languages.*

Proof. Since all the operators used to express the solution of regular language equations have constructive counterparts on automata, then there is a constuctive method for computing the solution. **QED**

The largest solution of parallel equations for prefix-closed regular languages was known already in the process algebra literature [7, 9, 6].

4 Finite State Machines

Definition 4.1 *A* **non-deterministic FSM** *(NDFSM), or simply an* **FSM** *or a* **machine***, is defined as a 5-tuple $M = \langle S, I, O, T, r \rangle$ where S represents the finite state space, I represents the finite input space, O represents the finite output space and $T \subseteq I \times S \times S \times O$ is the transition relation. On input i, the NDFSM at state p may transit to n and output o if and only if $(i, p, n, o) \in T$. State $r \in S$ represents the initial or reset state. We denote the restriction of T to $I \times S \times S$ (next state relation) by $T_n \subseteq I \times S \times S$, i.e., $(i, s, s') \in T_n \Leftrightarrow \exists o\ (i, s, s', o) \in T$; similarly, we denote the restriction of T to $I \times S \times O$ (output relation) by $T_o \subseteq I \times S \times O$, i.e., $(i, s, o) \in T_o \Leftrightarrow \exists s'\ (i, s, s', o) \in T$. If at least one transition is specified for each present state and input pair, the FSM is said to be* **complete***. If no transition is specified for some present state and input pair, the FSM is said to be* **partial***. An FSM is said to be* **trivial** *when $T = \emptyset$.*

It is convenient to think of the relations T_n and T_o as functions $T_n : I \times S \rightarrow 2^S$ and $T_o : I \times S \rightarrow 2^O$.

Definition 4.2 *An NDFSM is a* **pseudo non-deterministic FSM** *(PNDFSM) [1, 11], or* **observable FSM** *[10], if for each triple $(i, p, o) \in I \times S \times O$, there is exactly one state n such that $(i, p, n, o) \in T$.*

The qualification "non-deterministic" is because for a given input and present state, there may be more than one possible output; however, edges (i.e., transitions) carrying different outputs must go to different next states. The further qualification "pseudo" non-deterministic is because its underlying finite automaton is deterministic. In a PNDFSM the next state n is unique for a given combination of input, present state and output, so it can be given by a next state function $n = T_n(i, p, o)$. Since the output is non-deterministic in general, it is represented by a relation $T_o \subseteq I \times S \times O$.

Definition 4.3 *A* **deterministic FSM** *(DFSM) is an NDFSM where for each pair $(i, p) \in I \times S$, there is at most one next state n and one output o such that $(i, p, n, o) \in T$, i.e., there is at most one transition from p under i.*

In a DFSM the next state n and the output o can be given, respectively, by a next state function $n = T_n(i, p)$ and an output function $o = T_o(i, p)$.

The transition relation T of a NDFSM can be extended in the usual way to a relation on $I^\star \times S \times S \times O^\star$: given a present state p and an input sequence $i_1 \ldots i_k \in I^\star$, $(i_1 \ldots i_k, p, n, o_1 \ldots o_k) \in T$ iff there is a sequence $s_1 \ldots s_{k+1}$ such that $s_1 = p, \ldots, s_{k+1} = n$ and for each $j = 1, \ldots, k$ it holds that $(i_j, s_j, s_{j+1}, o_j) \in T$. A similar extension can be defined for T_p and T_n.

In this paper FSMs are assumed to be pseudo non-deterministic, unless otherwise stated. Notice that it is always possible to convert a general NDFSM into a PNDFSM by subset construction.

4.1 Languages of FSMs

We now introduce the notion of language associated with an FSM. This is achieved by looking at the automaton underlying a given FSM. For our purposes, we define an associated language over the alphabet $I \cup O$.

For a language over $I \cup O$, the automaton is obtained from the original FSM, by replacing each edge labelled by i/o with an edge labelled by i, followed by a new node (non-accepting state), followed by an edge labelled by o. All original states are made accepting.

Definition 4.4 *Given an FSM $M = \langle S, I, O, T, r\rangle$, consider the finite automaton $F(M) = \langle S \cup (S \times I), I \cup O, \Delta, r, S\rangle$, where $(i, s, (s, i)) \cup (o, (s, i), s') \in \Delta$ iff $(i, s, s', o) \in T$. The language accepted by $F(M)$ is denoted $L_r^{\cup}(M)$, and by definition is the* $\cup$**-language** *of M at state r. Similarly $L_s^{\cup}(M)$ denotes the language accepted by $F(M)$ when started at state s, and by definition is the $\cup$-language of M at state s. By construction, $L_s(M) \subseteq (IO)^\star$, where IO denotes the set $\{io \mid i \in I, o \in O\}$.*

$\epsilon \in L_r(M)$ because the initial state is accepting. An FSM M is **trivial** iff $L_r(M) = \epsilon$.

Definition 4.5 *A language L is an* **FSM language** *if there is an FSM M such that the associated automaton $F(M)$ accepts L. The language associated with a DFSM is sometimes called a* **behaviour** [1].

Remark When convenient, we say that FSM M_A has property X if its associated FSM language has property X.

Definition 4.6 *State t of FSM M_B is said to be a* **reduction** *of state s of FSM M_A (M_A and M_B are supposed to have the same input set), written $t \preceq s$, iff $L_t(M_B) \subseteq L_s(M_A)$. States t and s are* **equivalent states**, *written $t \cong s$, iff $t \preceq s$ and $s \preceq t$, i.e., when $L_t(M_B) = L_s(M_A)$. States that are not equivalent are called* **distinguishable**. *An FSM whose states are all pairwise distinguishable is a* **reduced** *FSM.*

Similarly, M_B is a **reduction** *of M_A, $M_B \preceq M_A$, iff r_{M_B}, the initial state of M_B, is a reduction of r_{M_A}, the initial state of M_A. When $M_B \preceq M_A$ and $M_A \preceq M_B$ then M_A and M_B are* **equivalent machines**, *i.e., $M_A \cong M_B$. Machines that are not equivalent are* **distinguishable**.

For complete DFSMs reduction and equivalence of states coincide. Given an FSM language, there is a family of equivalent FSMs associated to it; for simplicity we will usually talk of the FSM associated with a given FSM language. In this paper FSMs are assumed to be reduced, unless otherwise stated.

An FSM language is regular, whereas the converse is not true.

Theorem 4.1 *A regular language over alphabet $I \cup O$ is the language of a complete FSM over input alphabet I and output alphabet O iff $L \subseteq (IO)^\star$, L is IO-prefix-closed and IO-progressive. A regular language $L \subseteq (IO)^\star$ that is IO-prefix-closed, but not IO-progressive, is the language of a partial FSM.*

Given a regular language L over alphabet $I \cup O$, the following algorithm builds L^{FSM}, the largest subset of L that is the language of an FSM over input alphabet I and output alphabet O.

[1] The language associated with a NDFSM includes a set of behaviours.

Procedure 4.1 *Input: Regular language L over $I \cup O$; Output: Largest FSM language L^{FSM} over $I \cup O$.*

1. Build a deterministic automaton A accepting $L \cap (IO)^\star$.
2. Delete the initial state if it is a nonfinal state.
3. Delete all nonfinal states having incoming edges labelled with symbols from alphabet O, together with their incoming edges.
4. If the initial state has been deleted, then $L^{FSM} = \emptyset$. Otherwise, let $\hat{A}$ be the automaton produced by the procedure and L^{FSM} the language that $\hat{A}$ accepts. If there is no outgoing edge from the initial state of $\hat{A}$, then $\hat{A}$ accepts the trivial language $L^{FSM} = \{\epsilon\}$, otherwise it accepts a nontrivial FSM language L^{FSM}. Any FSM language in L must be a subset of L^{FSM}.

To obtain the largest subset of L which is the language of a complete FSM we must apply one more pruning algorithm.

Procedure 4.2 *Input: FSM Language L^{FSM} over $I \cup O$; Output: Largest IO-progressive FSM language $Prog(L^{FSM})$ over $I \cup O$.*

1. Build a deterministic automaton A accepting L^{FSM}.
2. Iteratively delete all states that are final *and* for which $\exists i \in I$ with no outgoing edge carrying the label i, together with their incoming edges, until the initial state is deleted or no more state can be deleted. Delete the initial state if $\exists i \in I$ with no outgoing edge carrying the label i.
3. If the initial state has been deleted, then $Prog(L^{FSM}) = \emptyset$. Otherwise, let $\hat{A}$ be the automaton produced by the procedure and $Prog(L^{FSM})$ the language that $\hat{A}$ accepts. Any I-progressive FSM language in L^{FSM} must be a subset of $Prog(L^{FSM})$.

Proposition 4.1 *An FSM whose language is L^{FSM} or $Prog(L^{FSM})$ can be trivially deduced from $\hat{A}$ by replacing pairs of consecutive edges labelled respectively with i and o by a unique edge labelled i/o.*

Proposition 4.2 *Given a regular language L over alphabet $I \cup O$, let M be an FSM over input alphabet I and output alphabet O. The language $L(M)$ of M is contained in L if and only if $L(M) \subseteq L^{FSM}$.*

Proof. Show that $L(M) \subseteq L \Rightarrow L(M) \subseteq L^{FSM}$. Indeed $L(M)$ is an FSM language contained in L and L^{FSM} is by construction the largest FSM language contained in L. So $L(M) \subseteq L^{FSM}$.

$L(M) \subseteq L^{FSM} \Rightarrow L(M) \subseteq L$, since by definition $L^{FSM} \subseteq L$. **QED**

4.2 Parallel Composition of FSMs

Different types of composition between pairs of FSMs may be defined, according to the protocol by which signals are exchanged. For a given composition operator and pair of FSMs we must establish whether that composition is defined, meaning that it yields a set of behaviours that can be described by another FSM. In general, the composition of FSMs is a partially specified function from pairs of FSMs to an FSM. In this paper we focus on the composition of FSMs by means of the parallel composition operator over languages introduced in Sec. 2. So the FSM yielded by the composition of FSMs M_A and M_B is the one whose language is obtained by the composition of the FSM languages associated with M_A and M_B.

Consider the pair of FSMs [2]

1. FSM M_A has input alphabet $I_1 \cup V$, output alphabet $U \cup O_1$ and transition relation T_A;

2. FSM M_B has input alphabet $I_2 \cup U$, output alphabet $V \cup O_2$ and transition relation T_B.

We define a parallel composition operator $\diamond$ that associates with a pair of FSMs M_A and M_B another FSM $M_A \diamond M_B$ such that:

1. the alphabet of the external inputs is $I_1 \cup I_2 = I$;

2. the alphabet of the external outputs is $O_1 \cup O_2 = O$.

Recall that, by definition of parallel composition of languages,

$$\alpha \in L(M_B) \diamond L(M_A) \text{ iff}$$

$$\alpha \in [L(M_B)_{\Uparrow I_1 \cup O_1} \cap L(M_A)_{\Uparrow I_2 \cup O_2}]_{\Downarrow I \cup O}.$$

Notice that the expansions $L(M_B)_{\Uparrow I_1 \cup O_1}$ and $L(M_A)_{\Uparrow I_2 \cup O_2}$ are needed to have the languages of M_B and M_A defined on the same alphabet.

Lemma 4.1 *If $L(M_A)$ and $L(M_B)$ are FSM $\cup$-languages, then $L(M_A) \diamond L(M_B) \cap\ (IO)^\star$ is an FSM $\cup$-language.*

Proof. $L(M_A) \diamond L(M_B) \cap\ (IO)^\star$ is prefix-closed, because prefix-closed $\cup$-languages are closed under $\diamond$ composition. Indeed, a state of the finite automaton corresponding to an FSM $\cup$-language is accepting iff it is the initial state or all its ingoing edges are labelled by symbols in O. The property is preserved by intersection and restriction over $I \cup O$. The intersection with $(IO)^\star$ makes sure that in the strings of the resulting FSM $\cup$-language an input is always followed by exactly one output, so that a corresponding FSM (with edges labelled by pairs (i/o)) can be reconstructed. Notice that $L(M_A) \diamond L(M_B) \cap\ (IO)^\star$ does not need to be progressive, because partial FSMs are allowed. **QED**

Therefore we can state the following definition.

Definition 4.7 *The* **parallel composition of FSMs** *M_B and M_A yields the FSM $M_A \diamond M_B$ with language*

$$L(M_A \diamond M_B) = L(M_A) \diamond L(M_B) \cap\ (IO)^\star.$$

If the language $L(M_A) \diamond L(M_B) \cap\ (IO)^\star = \{\epsilon\}$, then $M_A \diamond M_B$ is a trivial FSM.

[2] For sake of simplicity the alphabets I_1, I_2, O_1, O_2, U, V are assumed to be disjoint.

The previous definition is sound because the language $L(M_A) \diamond L(M_B) \cap\ (IO)^\star$ by Lemma 4.1 is an FSM language, which may correspond to a complete or partial FSM according to whether the language $L(M_A) \diamond L(M_B) \cap\ (IO)^\star$ is IO-progressive or not. Then by subset construction and reduction we produce a reduced observable FSM. In summary, we convert from the FSMs M_B and M_A to the automata accepting their FSM languages, operate on them and then we convert back from the resulting automaton to an FSM; then we produce a reduced PNDFSM (we assume that M_B and M_A are PNDFSMs).

4.3 Solutions of Equations over FSMs

Consider a general interconnection of two FSMs M_A and M_B, where FSM M_A has input signals I_1 and V and output signals U and O_1, and FSM M_B has input signals I_2 and U and output signals V and O_2. The network implements a specification M_C with input signals I_1, I_2 and output signals O_1, O_2. Supposing that M_A and M_C are known and M_B is unknown, we want to define an equation of the type $M_A \odot M_X \preceq M_C$, to capture the FSMs M_B that in place of M_X let the network of M_A and M_B match the specification M_C. Through Definition 4.4 we have seen a way to associate an FSM language to a given FSM, and the related composition operator $\diamond$ has been introduced in Sec. 4.2; therefore we introduce the following equation over FSMs:

$$M_A \diamond M_X \preceq M_C$$

and solve it by building first the related language equation $L(M_A) \diamond L(M_X) \subseteq L(M_C) \cup \overline{(IO)^\star}$ [3], where $L(M_A)$ and $L(M_C)$ are the FSM languages associated with FSMs M_A and M_C. Then we derive the FSM M_S associated with S. When there is no ambiguity $A \diamond X \subseteq C \cup \overline{(IO)^\star}$ denotes the language equation $L(M_A) \diamond L(M_X) \subseteq L(M_C) \cup \overline{(IO)^\star}$.

Given alphabets I_1, I_2, U, V, O_1, O_2, an FSM M_A over inputs $I_1 \cup V$ and outputs $U \cup O_1$, and an FSM M_C over inputs $I_1 \cup I_2$ and outputs $O_1 \cup O_2$, consider the FSM equation

$$M_A \diamond M_X \preceq M_C, \tag{3}$$

whose unknown is an FSM M_X over inputs $I_2 \cup U$ and outputs $V \cup O_2$. We will use the shortened notation $I = I_1 \cup I_2$ and $O = O_1 \cup O_2$.

Definition 4.8 *FSM M_B is a* **solution** *of the equation $M_A \diamond M_X \preceq M_C$, where M_A and M_C are FSMs, if and only if $M_A \diamond M_B \preceq M_C$.*

Converting to the related FSM languages, we construct the associated language equation

$$L(M_A) \diamond L(M_X) \subseteq L(M_C) \cup \overline{(IO)^\star}, \tag{4}$$

where $L(M_A)$ is over $I_1 \cup U \cup V \cup O_1$, $L(M_C)$ is over $I_1 \cup I_2 \cup O_1 \cup O_2$ and the unknown FSM language is over $I_2 \cup U \cup V \cup O_2$. The previous equation can be rewritten for simplicity as

$$A \diamond X \subseteq C \cup \overline{(IO)^\star}. \tag{5}$$

[3] $M_A \diamond M_X \preceq M_C$ iff $L(M_A \diamond M_X) \subseteq L(M_C)$ iff $L(M_A \diamond M_X) \equiv L(M_A) \diamond L(M_X) \cap (IO)^\star \subseteq (M_C)$ iff $L(M_A) \diamond L(M_X) \subseteq L(M_C) \cup \overline{(IO)^\star}$.

We want to characterize solutions that are FSM languages. We know from Theorem 2.1 that the largest solution of $A \diamond X \subseteq C \cup \overline{(IO)^\star}$ is the language $S = \overline{A \diamond (\overline{C} \cap (IO)^\star)}$, if $S \neq \emptyset$.

In general S is not an FSM language. To compute the largest FSM language contained in S, S^{FSM}, we must compute the largest prefix-closed language contained in $S \cap ((I_2 \cup U)(V \cup O_2))^\star$.

Theorem 4.2 *Let A and C be FSM languages. The largest FSM language solution of the equation $A \diamond X \subseteq C \cup \overline{(IO)^\star}$ is denoted by S^{FSM}, where $S = \overline{A \diamond (\overline{C} \cap (IO)^\star)}$. If $S = \emptyset$ then $S^{FSM} = \emptyset$; if $S \neq \emptyset$, S^{FSM} is obtained by applying Procedure 4.1 to S. If $S^{FSM} = \emptyset$ then the FSM language equation $A \diamond X \subseteq C \cup \overline{(IO)^\star}$ has no solution.*

Proof. The first step of Procedure 4.1 computes the intersection of S with $((I_2 \cup U)(V \cup O_2))^\star$ to enforce that the solution, if it exists, is an FSM language with input alphabet $I_2 \cup U$ and output alphabet $V \cup O_2$. Since A and C are regular languages, $S \cap ((I_2 \cup U)(V \cup O_2))^\star$ is regular too and, by construction, Procedure 4.1 extracts the largest FSM language contained in it. **QED**

By Proposition 4.1, it is easy to derive an FSM $M_{S^{FSM}}$ associated with S^{FSM}. This allows us to talk about FSMs that are solutions of FSM equations, meaning any reduction of the FSM $M_{S^{FSM}}$, as guaranteed by Prop. 4.2.

For logic synthesis applications, we assume that M_A and M_C are complete FSMs and we require that the solution is a complete FSM too. This is obtained by applying Procedure 4.2 to S^{FSM}, yielding $Prog(S^{FSM})$, the largest $(I_2 \cup U)(V \cup O_2)$-progressive FSM language $\subseteq ((I_2 \cup U)(V \cup O_2))^\star$.

Proposition 4.3 *FSM M_B is a solution of the equation $M_A \diamond M_X \preceq M_C$, where M_A and M_C are FSMs, if and only if M_B is a reduction of the FSM $M_{S^{FSM}}$ associated with S^{FSM}, where S^{FSM} is obtained by applying Procedure 4.1 to S, where $S = \overline{A \diamond (\overline{C} \cap (IO)^\star)}$. If $S^{FSM} = \emptyset$ then no FSM is a solution. The largest complete FSM solution $M_{Prog(S^{FSM})}$ is found, if it exists, by Procedure 4.2. A complete FSM is a solution if and only if it is a reduction of the largest complete solution $M_{Prog(S^{FSM})}$.*

Furthermore, we restrict the attention to the solutions B such that $A_{\Uparrow I_2 \cup O_2} \cap B_{\Uparrow I_1 \cup O_1} \cap (IO)^\star_{\Uparrow U \cup V}$ is an $I^\star O$-progressive FSM language $\subseteq (I(U \cup V)^\star O)^\star$ and so the composition is a complete FSM over inputs $I_1 \cup I_2$ and outputs $O_1 \cup O_2$. Since $A_{\Uparrow I_2 \cup O_2} \cap B_{\Uparrow I_1 \cup O_1} \cap (IO)^\star_{\Uparrow U \cup V}$ is IO-prefix-closed, it corresponds to a partial FSM. We have to restrict it so that it is also $I^\star O$-progressive, which corresponds to a complete FSM.

To now discuss the issue of progressiveness and livelocks (endless cycles of internal actions).

Definition 4.9 *A solution B of Eq. 5 is A-**compositionally $I^\star O$-progressive** if*

$$A_{\Uparrow I_2 \cup O_2} \cap B_{\Uparrow I_1 \cup O_1} \cap (IO)^\star_{\Uparrow U \cup V}$$

is $I^\star O$-progressive.

If B is compositionally $I^\star O$-progressive, the restriction to $I \cup O$ of the composition is an IO-progressive language, meaning that its corresponding FSM is complete.

Definition 4.10 *A solution B of Eq. 5 is A-**compositionally prefix $I^\star O$-progressive** if*

$$Init(A)_{\Uparrow I_2 \cup O_2} \cap Init(B)_{\Uparrow I_1 \cup O_1} \cap Init((IO)^\star)_{\Uparrow U \cup V}$$

is $I^\star O$-progressive.

A compositionally prefix $I^\star O$-progressive solution yields a composition that has no $(u \cup v)^\star$ cycles without exit. Such a solution is also compositionally $I^\star O$-progressive, but the converse does not hold.

Definition 4.11 *A solution B of Eq. 5 is A-**compositionally prefix $(U \cup V)$-convergent** (abbreviated as A-**compositionally $(U \cup V)$-convergent**) if*

$$Init(A)_{\Uparrow I_2 \cup O_2} \cap Init(B)_{\Uparrow I_1 \cup O_1} \cap Init((IO)^\star)_{\Uparrow U \cup V}$$

is $(U \cup V)$-convergent.

A compositionally $(U \cup V)$-convergent solution yields a composition that has no $(u \cup v)^\star$ cycles, i.e., it is livelock-free. A compositionally $I^\star O$-progressive or compositionally prefix $I^\star O$-progressive solution does not need to be compositionally $(U \cup V)$-convergent.

Theorem 4.3 *Let B be an $(I_2 \cup U)(V \cup O_2)$-progressive solution of $A \diamond X \subseteq C \cup \overline{(IO)^\star}$ and let A be $(I_1 \cup V)(U \cup O_1)$-progressive. If B is compositionally prefix $(U \cup V)$-convergent, then B is compositionally prefix $I^\star O$-progressive.*

Proof. Since the components A and B are progressive their composition $\mathrm{Init}(A)_{\Uparrow I_2 \cup O_2} \cap \mathrm{Init}(B)_{\Uparrow I_1 \cup O_1} \cap \mathrm{Init}((IO)^\star)_{\Uparrow U \cup V}$ is deadlock-free, i.e., it never stops because a component does not have a transition under a given input. If the composition is also $(U \cup V)$-convergent, there can be no livelocks, i.e., there are no cycles labeled with actions from the set $U \cup V$. Therefore an external input, after a finite path labelled with internal actions, must be followed by an external output. **QED**

If S^{FSM} is compositionally prefix $I^\star O$-progressive, then it is the largest compositionally prefix $I^\star O$-progressive solution of the equation. However, not every non-empty subset of S^{FSM} inherits this feature. If S^{FSM} is not compositionally prefix $I^\star O$-progressive, then denote the largest compositionally prefix $I^\star O$-progressive subset of S^{FSM} by $cI^\star OProgInit(S^{FSM})$. Conceptually $cI^\star OProgInit(S^{FSM})$ is obtained from S^{FSM} by deleting each string $\alpha \in (I(U \cup V)^\star O)^\star$ such that, for some $i \in I$, there is no $(u \cup v)^\star \in (U \cup V)^\star$ and no $o \in O$ for which it holds $\alpha\ i(u \cup v)^\star o \in Init(A)_{\Uparrow I_2 \cup O_2} \cap Init(S^{FSM})_{\Uparrow I_1 \cup O_1} \cap Init((IO)^\star)_{\Uparrow U \cup V}$. We designed a procedure (not reported here) to compute the largest compositionally prefix $I^\star O$-progressive solution.

Moreover we may require that the solutions are compositionally $(U \cup V)$-convergent (see Definition 2.7), to rule out all livelocks from the composition. However, when S^{FSM} is not compositionally $(U \cup V)$-convergent, then the largest $(U \cup V)$-convergent solution does not exist and each finite IO-prefix-closed subset of S^{FSM} is a compositionally $(U \cup V)$-convergent solution; thus no such sequence can be deleted from the largest solution without missing a compositionally $(U \cup V)$-convergent solution.

4.4 Bounded Parallel Composition of FSMs

Here we discuss the solutions whose composition with the context produces an external output after at most l internal actions and provide the key steps to solve FSM equations under bounded parallel composition.

Definition 4.12 *The* **l-bounded parallel composition of FSMs** *M_B, over input alphabet $I_2 \cup U$ and output alphabet $O_2 \cup V$, and of M_A, over input alphabet $I_1 \cup V$ and output alphabet $O_1 \cup U$, yields the FSM $M_A \diamond_l M_B$ with language*

$$\begin{aligned} L(M_A \diamond_l M_B) &= L(M_A) \diamond_l L(M_B) \cap (IO)^\star \\ &= [L(M_A)_{\Uparrow I_2 \cup O_2} \cap L(M_B)_{\Uparrow I_1 \cup O_1} \\ &\quad \cap (I \cup O)^\star_{\Uparrow (U \cup V, l)}]_{\Downarrow I \cup O} \cap (IO)^\star. \end{aligned}$$

When $l = \infty$, this reduces to parallel composition of FSMs.

Proposition 4.4 *FSM M_B is a solution of the equation $M_A \diamond_l M_X \preceq M_C$, where M_A and M_C are FSMs, if and only if M_B is a reduction of the FSM $M_{S^{FSM}}$ associated with S^{FSM}, where S^{FSM} is obtained by applying Procedure 4.1 to S, where $S = \overline{(A_{\Uparrow I_2 \cup O_2} \cap (\overline{C} \cap (IO)^\star)_{\Uparrow (U \cup V, l)})_{\Downarrow I_2 \cup U \cup V \cup O_2}}$. If $S^{FSM} = \emptyset$ then no FSM is a solution. S^{FSM} is the largest compositionally $(U \cup V)$-convergent solution of $M_A \diamond_l M_X \leq M_C$. The largest complete FSM solution $M_{Prog(S^{FSM})}$ is found, if it exists, by Procedure 4.2.*

Theorem 4.4 *A solution M_B of $M_A \diamond_l M_X \preceq M_C$ is also a compositionally $(U \cup V)$-convergent solution of $M_A \diamond M_X \preceq M_C$.*

If M_A and M_B are also complete, then M_B is a compositionally prefix $I^\star O$-progressive and compositionally $I^\star O$-progressive solution of $M_A \diamond M_X \preceq M_C$.

Proof. By construction, a solution M_B of $M_A \diamond_l M_X \preceq M_C$ is compositionally $(U \cup V)$-convergent. A solution M_B of $M_A \diamond_l M_X \preceq M_C$ is also a solution of $M_A \diamond M_X \preceq M_C$, because when $l = \infty$ the operator $\diamond_l$ becomes the operator $\diamond$.

By Theorem 4.3, the fact that M_B is compositionally $(U \cup V)$-convergent, together with the completeness of M_A and M_B, imply that M_B is compositionally prefix $I^\star O$-progressive and therefore compositionally $I^\star O$-progressive. **QED**

However, $M_A \diamond M_X \preceq M_C$ may be solvable even though $M_A \diamond_l M_X \preceq M_C$ has no solution. This may happen when $M_A \diamond M_X \preceq M_C$ has no compositionally $I^\star O$-progressive solution. If $M_A \diamond_l M_X \preceq M_C$ has no complete solution, it is open whether there is a compositionally $I^\star O$-progressive solution.

5 Conclusions

We addressed the problem of finding an unknown component in a network in order to satisfy a global system specification. In this paper we addressed only the parallel language equation $A \diamond X \subseteq C$ and provided solutions and algorithms for classes of automata and constrained versions of the problem. In particular we showed how to compute the largest FSM language that is a solution of the language equation $A \diamond X \subseteq C$, the largest complete solution, and the largest solution whose composition with the context A yields a complete FSM language.

We applied these techniques to a classical synthesis' problem of a converter between a given mismatched pair of protocols, using their specifications. and those of the channel and the user services. The problem was addressed in [5, 2] with supervisory control techniques. We were able to derive the largest solution, and the largest compositionally progressive solution, which were not previously reported in the literature.

6 Acknowledgments

The first author was partly supported by the Russian Fund of Basic Research (Grant 99-01-00337). Both the first and the third author gratefully acknowledge the support of a NATO travel grant (NATO Linkage Grant No. 971217). The second author gratefully acknoweledges the support of the MADESSII Project (Italian National Research Council). The fourth author gratefully acknowledges the support of NSERC (Grant OGP0194381).

References

[1] E. Cerny. Verification of I/O trace set inclusion for a class of non-deterministic finite state machines. In *The Proceedings of the International Conference on Computer Design*, pages 526–530, October 1992.

[2] H. Hallal, R. Negulescu, and A. Petrenko. Design of divergence-free protocol converters using supervisory control techniques. In *7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000*, volume 2, pages 705–708, December 2000.

[3] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.

[4] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. *Synthesis of FSMs: functional optimization*. Kluwer Academic Publishers, 1997.

[5] R. Kumar, S. Nelvagal, and S.I. Marcus. A discrete event systems approach for protocol conversion. *Discrete Event Dynamic Systems: Theory & Applications*, 7(3):295–315, June 1997.

[6] W.C. Mallon, J.T. Tijmen, and T. Werhoeff. Analysis and applications of the XDI model. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 231–242, 1999.

[7] P. Merlin and G.V. Bochmann. On the construction of submodule specifications and communication protocols. *ACM Transactions on Programming Languages and Systems*, 5(1):1–25, January 1983.

[8] A. Petrenko and N. Yevtushenko. Solving asynchronous equations. In S. Budkowski, A. Cavalli, and E. Najm, editors, *Formal Description Techniques and Protocol Specification, Testing and Verification - FORTE XI/PSTV XVIII '98*, pages 231–247. Kluwer Academic Publishers, November 1998.

[9] H. Qin and P. Lewis. Factorisation of finite state machines under strong and observational equivalences. *Formal Aspects of Computing*, 3:284–307, Jul.-Sept. 1991.

[10] P. H. Starke. *Abstract Automata*. North-Holland Pub. Co.; American Elsevier Pub. Co., 1972.

[11] Y. Watanabe and R.K. Brayton. The maximum set of permissible behaviors for FSM networks. In *IEEE International Conference on Computer-Aided Design*, pages 316–320, November 1993.