PROGRESSIVE SOLUTIONS TO FSM EQUATIONS

Khaled El-Fakih[1] and Nina Yevtushenko[2]

[1]Verimag, University Joseph Fourier (France), American University of Sharjah (UAE)
[2]*Tomsk State University (Russia)
elfakih@imag.fr , yevtushenko@elefot.tsu.ru

Abstract. The *equation solving* problem is to derive the behavior of the unknown component *X* knowing the joint behavior of the other components (or the context) *C* and the specification of the overall system *S*. The component *X* can be derived by solving the Finite State Machine (FSM) equation $C \lozenge X \sim S$, where $\lozenge$ is the parallel composition operator and ~ is the trace equivalence or the trace reduction relation. A solution *X* to an FSM equation is called *progressive* if for every external input sequence the composition $C \lozenge X$ does not fall into a livelock without an exit. In this paper, we formally define the notion of a progressive solution to a parallel FSM equation and present an algorithm that derives a largest progressive solution (if a progressive solution exists). In addition, we generalize the work to a system of FSM equations. Application examples are provided.

1. Introduction

The equation solving problem can be formulated as solving an equation $C \lozenge X \sim S$, where $\lozenge$ is a composition operator and ~ is a conformance relation. Usually the behavior of *X*, *S*, and *A*, is represented using finite state models such as finite state automata, I/O automata, Labeled Transition Systems (LTSs), and Finite Sate Machines (FSMs) [for references, see, for example, 1, 2]. The conformance relation ~ often is the trace equivalence relation, denoted $\cong$, or the trace containment (or reduction) relation, denoted $\leq$, and the composition operator is either the synchronous $\bullet$ or the parallel $\lozenge$ composition operator. The applications of the equation solving problem were considered in the context of the design of communication protocols and protocol converters, selection of test cases, and the design of controllers for discrete event systems.

If an equation has a solution, then it is known to have a largest solution that includes all other solutions [3]. However, not every solution of a largest solution is of a practical use. Usually, we are interested in so-called *progressive solutions* [3, 4] where for every external input sequence the composition $C \lozenge X$ does not fall into a livelock without an exit. If an automata

* The second author acknowledges the support of RFBR-NSC grant 06-08-89500

equation has a progressive solution then the equation is known to have a largest progressive solution [1] that contains all progressive solutions. However, not each solution of a largest progressive solution is progressive. The problem of deriving and characterizing progressive solutions has been studied for I/O automata, finite automata, and for FSMs over the synchronous composition operator.

In the first part of this paper, we first, define the notion of a progressive solution for the FSM equation $C \lozenge X \cong S$ and for the FSM inequality $C \lozenge X \leq S$ [2, 3]. Given, possibly non-deterministic, FSMs context C and specification S, we present an algorithm that provides a largest complete progressive FSM solution to the equation $C \lozenge X \cong S$ (inequality $C \lozenge X \leq S$) if the equation (the inequality) has a progressive solution. A largest progressive solution is derived by trimming a proper largest solution. We note that the results [2] obtained for deriving a progressive solution in the area of finite automata cannot be directly applied for deriving a progressive solution over FSMs, since, by definition, a progressive solution to an FSM equation can block output actions allowed by the specification. Moreover, in this paper, we propose a simpler algorithm for deriving a largest progressive solution than that for automata equations [1]. In addition, the results obtained for deriving a progressive solution of an FSM equation over the synchronous composition operator [4] cannot be directly applied for deriving a progressive solution over the parallel composition operator, since the notions of the parallel and synchronous composition operators are different; the parallel composition operator allows to produce an external output not directly after external input but, possibly after a sequence of internal actions.

In some application areas, given a finite set of $k > 1$ contexts C_i and service specifications S_i, one is interested in finding a solution (the unknown component) X that combined with C_i meets the specification S_i, for $i = 1, ..., k$. The problem of finding such a solution X is the problem of solving a *system of equations*. A largest solution to a system of FSM equations can be derived as given in [5]. In the second part of this paper, we deal with a progressive solution to a system of FSM equations. A largest progressive solution to the system of equations is derived by intersecting largest progressive solutions of every equation and then by deriving the largest submachine of this intersection that is progressive for every equation.

2. Preliminaries

Finite State Machine (FSMs): A FSM, or *machine* hereafter, is a quintuple $A = \langle S, I, O, T_A, s_0 \rangle$, where S is a finite nonempty set of states with the initial state s_0, I and O are input and output alphabets, and $T_A \subseteq S \times I \times O \times S$ is a

transition relation. In this paper, we consider only *observable* FSMs, i.e. for each triple $(s,i,o)\in S\times I\times O$ there exists at most one state $n\in S$ such that $(s,i,n,o)\in T$. An FSM A is called *complete*, if $\forall s \in S$ and $\forall i \in I$ $\exists o\in O$ and $\exists s' \in S$, such that $(s, i,o, s')\in T_A$. If A is not complete, then it is called *partial*. An FSM A is called *deterministic*, if $\forall s \in S$ and $\forall i\in I$ there exist at most one pair of output o and state s', such that $(s, i,o, s')\in T_A$. An FSM $B = (Q,I,O,T_B,q_0)$ is a sub-machine of A if $Q \subseteq S$ and $T_B \subseteq T_A$. The *largest complete submachine* of FSM A can be obtained by iterative deleting states where the behavior of the FSM is not defined at least for a single input. Each complete sub-machine of A is a submachine of the largest complete sub-machine of A (if it exists). As usual, the transition relation T_A of FSM $A = \langle S, I, O,T_A, s_0\rangle$ can be extended to sequences over the alphabet I. In this paper, we consider only initially connected FSMs, i.e., each state of an FSM is reachable from the initial state.

Given an FSM A, the set of all I/O sequences generated at state s of A is called the *language* of A *generated at state s*, or simply the set of I/O sequences at s, written $L_s(A)$. The language, generated by the FSM A at the initial state is called the language of the FSM A and is denoted by $L(A)$, for short. The FSM $\langle\{t_0\}, I, O, T, t_0\rangle$, denoted $MAX(I,O)$, where $T = \{t_0\}\times I\times O\times\{t_0\}$, is called *maximum* over the input alphabet I and the output alphabet O. The maximum machine $MAX(I,O)$ accepts the language $(IO)^*$. An FSM $B = \langle Q, I, O, T_B, q_0\rangle$ is a *reduction* of FSM $A = \langle S, I, O, T_A , s_0\rangle$, written $A \leq B$, if $L_B \subseteq L_A$. If $L_B = L_A$ then FSMs A and B are *equivalent*. For complete deterministic FSMs the reduction and the equivalence relations coincide.

The common behavior of two FSMs can be described by the intersection of these machines. The *intersection* $A \cap B$ of FSMs $A=\langle S, I, O, T_A, s_0\rangle$ and $B=\langle Q, I, O, T_B, q_0\rangle$ is the largest connected sub-machine of the FSM $\langle S \times Q, I, O, T_{A\cap B}, s_0q_0\rangle$. Formally, $T_{A\cap B} = \{(sq, i, o, s'q') \mid (s, i, o, s') \in T_A \wedge (q, i, o, q') \in T_B\}$. The language of $A \cap B$ is the intersection $L(A) \cap L(B)$. The intersection of two observable FSMs is an observable FSM; however, the intersection of complete FSMs can be partial. FSM languages are regular, and thus, the underlying model for an FSM is a finite automaton. When solving a parallel equation an FSM is represented by an automaton by unfolding each transition of the FSM [2 - 4].

Automata and FSMs: A *finite automaton*, is a quintuple $S = \langle S, V, \delta_S, s_0, F_S\rangle$, where S is a finite nonempty set of states with the initial state s_0 and a subset F_S of *final* (or *accepting*) states, V is an alphabet of actions, and $\delta_S \subseteq S \times V \times S$ is a transition relation. An automaton $\langle S', V, \delta'_S, s'_0, F'_S\rangle$ is a *submachine* of the automaton S if $S' \subseteq S$, $\delta'_S \subseteq \delta_S$, and $F'_S \subseteq F_S$. The automaton S is *deterministic*, if $\forall s \in S$ and $\forall v\in V$, $\exists$ at most one state s', such that $(s, v, s') \in \delta_S$. The language $L(S)$ *generated* or

accepted by S is known to be regular. Given a sequence $\alpha \in V^*$ and an alphabet W, a *W-restriction* of α, written $\alpha_{\downarrow W}$, is obtained by deleting from α all symbols that belong to the set $V \setminus W$. Given a sequence $\alpha \in V^*$ and an alphabet W, a *W-expansion* of α, written $\alpha_{\uparrow W}$, is a set that contains each sequence over alphabet $(V \cup W)$ with the V–projection α.

Well-known results state that regular languages are closed under the union, intersection, complementation, restriction and expansion and the constructions for deriving corresponding automata could be found, for example, in [1, 3, 6]. Let $P = \langle P, V, \delta_P, p_0, F_P \rangle$ be an automaton which accepts the language L. *Restriction* ($\downarrow$): Given a non-empty subset U of V, the automaton $P_{\downarrow U}$ that accepts the language $L_{\downarrow U}$ over U is obtained by replacing each edge (s,a,s') in P by the edge (s,ε,s').[1] *Expansion* ($\uparrow$): Given alphabet U, the automaton $P_{\uparrow U}$ that accepts the language $L_{\uparrow U}$ over $U \cup V$ is obtained by adding (s,a,s) $\forall a \in U \setminus V$ for each state s of P.

We note that not each automaton has an FSM language. However, it is known that given an automaton B over alphabet $I \cup O$, $I \cap O = \varnothing$, there exists a largest subset of the language of the automaton B that is the language of an FSM, denoted B^{FSM}, which can be constructed by intersecting B with $(IO)^*$ and deleting all non-accepting states from the resulting automaton, which have an incoming transition labeled with an output action o. The language of an FSM C over input I and output O is a subset of the language of an automaton B if and only if C is a reduction of B^{FSM} [3].

Parallel Composition of FSMs: Consider a system of two complete communicating FSMs $A = \langle A, I_1 \cup V, O_1 \cup U, T_A, s_0 \rangle$ and $B = \langle T, I_2 \cup U, O_2 \cup V, T_B, t_0 \rangle$ [1 - 3]. As usual, for the sake of simplicity, we assume that alphabets I_1, V, O_1, U, I_2, O_2 are pair-wise disjoint. The alphabet $Ext_{in} = I_1 \cup I_2$ represents the external inputs of the composition, while the alphabet $Ext_{out} \subseteq O_1 \cup O_2$ represents the external outputs of the composition; $Ext = Ext_{in} \cup Ext_{out}$, $Int = U \cup V$. The two FSMs communicate under a single message in transit, i.e., the next external input is submitted to the system only after it produced an external output to the previous input. The collective behavior of the two communicating FSMs can be described by an FSM. The *parallel composition of FSMs* A and B, denoted $C \lozenge_{Ext} B$ or simply $C \lozenge B$, can be obtained as follows [2, 3]: First, for FSMS A and B, the corresponding automata $Aut(A)$ and $Aut(B)$ are derived. Then, the intersection $(Aut(A)_{\uparrow I_2 \cup O_2} \cap Aut(B)_{\uparrow I_1 \cup O_1})_{\downarrow Ext} \cap Aut(MAX(I,O))$ is converted into an FSM. It is known that the parallel composition of two complete FSMs can be partial, since the communicating FSMs can fall into an infinite dialogue (live-

[1] Apply the closure procedure to obtain an equivalent deterministic automaton without ε-moves [6].

lock) without producing an external output. In this case, the projection of $(Aut(A)_{\uparrow I_2 \cup O_2} \cap Aut(B)_{\uparrow I_1 \cup O_1})_{\downarrow Ext}$ onto I does not coincide with I^*. Formally, the composition falls into live-lock if $Aut(A)_{\uparrow I_2 \cup O_2} \cap Aut(B)_{\uparrow I_1 \cup O_1} \cap Aut(MAX(I,O)_{\uparrow Int}$ has a state where the generated language is empty.

FSM Equations: Let $C = (C,\ I_1 \cup V,\ O_1 \cup U,\ T_C,\ c_0)$ and $S = (S,\ Ext_{in},\ Ext_{out},\ T_S,\ s_0)$ be two complete FSMs. An expression "$C \lozenge_{Ext} X \cong S$" ("$C \lozenge_{Ext} X \leq S$") is called an *FSM equation* (an *FSM inequality*) w.r.t. the unknown X that represents an FSM over the input alphabet $I_2 \cup U$, $I_2 = Ext_{in} \backslash I_1$, and the output alphabet $O_2 \cup V$, $O_2 = Ext_{out} \backslash O_1$. The FSM C is usually called the *context*, and the FSM S is usually called the *specification*. As usual, an FSM equation can have no solution while an FSM inequality is always solvable, as the trivial FSM with the language that contains only the empty sequence always is a solution to an FSM inequality. If an FSM inequality and a solvable FSM equation have a complete solution then they are known to have a largest complete solution [2, 3]. A largest complete solution M to the equation $C \lozenge_{Ext} X \cong S$ can be obtained as the largest complete submachine of the FSM over input alphabet $I_2 \cup U$ and output alphabet $O_2 \cup V$ which corresponds to the automaton $\Lambda(C,S,MAX) = \overline{(Aut(C)_{\uparrow I_2 \cup O_2} \cap \overline{Aut(S)}_{\uparrow U \cup V})_{\downarrow I_2 \cup O_2 \cup U \cup V}} \cap Aut(MAX(I_2 \cup U, O_2 \cup V))$, if such a complete submachine exists. We note that in this paper, we do not merge equivalent states (for the reasons shown later) of the automaton $\Lambda(C,S,MAX)$ when applying the closure procedure for deriving an equivalent deterministic automaton without ε-moves after the restriction operator. If such a machine M does not exist the equation and the inequality have no complete solutions. If the machine M exists then M is a largest complete solution to the inequality $C \lozenge_{Ext} X \leq S$. Moreover, each reduction of M also is a solution to the inequality. If the composition $C \lozenge_{Ext} M$ is equivalent to S [2] then M is a largest complete solution to the equation. If the composition $C \lozenge_{Ext} M$ is not equivalent to S, then the equation has no complete solution. However, not each complete reduction of M is a solution to the equation.

As an example of a largest complete solution of an FSM equation, consider the specification FSM S_1 with transitions $(1,x,o_3,1)(1,i,o_1,1)$ $(1,x,o_2,1)$ and the context C shown in Fig. 1. The context C is defined over external inputs $I_1 = \{i\}$, external outputs $O_1 = \{o_1, o_2, o_3\}$, internal inputs $V = \{v_1, v_2, v_3\}$ and internal outputs $U = \{u_1, u_2\}$. Specification S_1 is defined over external inputs $Ext_{in} = \{i, x\}$ and external outputs $Ext_{out} = \{o_1, o_2, o_3\}$. A solution to an FSM equation $C \lozenge_{Ext} X \cong S_1$ is defined over the external input alphabet $I_2 = \{x\}$, the internal input alphabet $U = \{u_1, u_2\}$ and the internal output alphabet $V = \{v_1, v_2, v_3\}$. A largest complete solution to the equation is shown in Fig. 2.

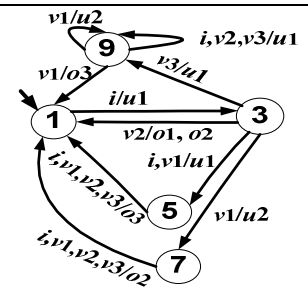

Fig. 1. Context FSM *C*

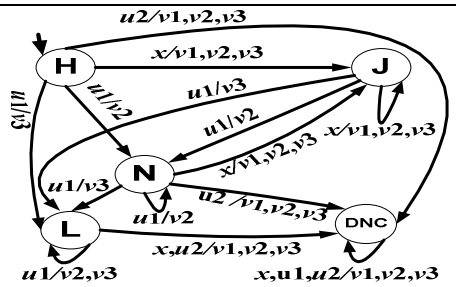

Fig. 2. A largest complete solution to $C \lozenge_{Ext} X \cong S_1$

3. Progressive Solutions to FSM Equations

Consider an FSM equation $C \lozenge X \cong S$, where C and S are FSMs over input alphabets $I_1 \cup V$ and $I_1 \cup I_2$ and over output alphabets $O_1 \cup U$ and $O_1 \cup O_2$ correspondingly, while X is the unknown FSM over input alphabet $I_2 \cup U$ and output alphabet $O_2 \cup V$. A solution *Prog* to an FSM equation $C \lozenge X \cong S$ (or inequality) is *progressive* if the system $C \lozenge Prog$ cannot fall into a live-lock under any external input sequence, i.e., for each external input action of an input sequence the composition eventually produces an external output. Formally, a solution *Prog* to an FSM equation $C \lozenge X \cong S$ (or to inequality $C \lozenge X \leq S$) is progressive if *Prog* is a complete FSM and the intersection $Aut(C)\uparrow_{(I2 \cup O2)} \cap Aut(Prog)\uparrow_{(I1 \cup O1)} \cap Aut(MAX(I,O))\uparrow_{U \cup V}$ has no states where the empty language is generated. The definition of a progressive solution requires that the above intersection has no cycles over internal actions without an exit from the cycle with an external output. If we consider a deterministic context FSM then each complete deterministic solution to the equation is progressive.

As an example of a non-progressive solution, consider a largest complete FSM solution *Largest*, shown in Fig. 2. At the initial states 1 of context *C* (Fig. 1) and H of *Largest*, if the external input *i* is applied to the context, FSM *C* produces the internal output u_1. In response to the input u_1, the FSM *Largest* may produce the output v_3, and then the system $C \lozenge Largest$ falls into a livelock. This is due to the fact that states (9,L,B) and (10,M,B) of the intersection shown in Fig. 3 are non-progressive. The *O*-restriction of the language generated at these states is empty.

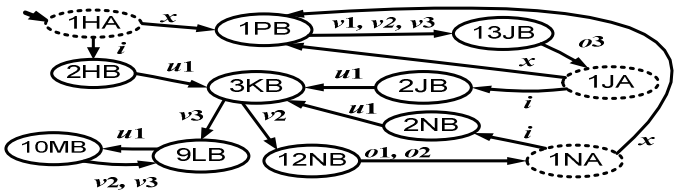

Fig. 3. The intersection $Aut(C)\uparrow_{(I2\cup O2)} \cap Aut(Largest)\uparrow_{(I1\cup O1)} \cap Aut(MAX(I,O))\uparrow_{U\cup V}$

In the following, we identify a property of a solution *Sol* to an FSM equation such that a largest progressive reduction of *Sol* can be derived as an appropriate submachine of *Sol*. In particular, the largest complete submachine of an FSM corresponding to the automaton $\Lambda(C,S,MAX)$ (without merging equivalent states) possesses this property.

A solution $P = (P,\ I_2\cup U,\ O_2\cup V,\ T_P,\ p_0)$ to an FSM inequality $C \lozenge_{Ext} X \leq S$ is called *perfect in the context C* (or simply *perfect*) if for each state (c,p,t_0) of the intersection $Aut(C)\uparrow_{(I2\cup O2)} \cap Aut(P)\uparrow_{(I1\cup O1)} \cap Aut(MAX(I,O))\uparrow_{U\cup V}$ that has an incoming transition labeled with an action $a \in O_2\cup V$, the $(I_2\cup O_2\cup U\cup V)$-projection of the language accepted at state (c,p,t_0) equals to the set of I/O sequences which take the FSM *P* from the initial state to state *p*.

Given a solution *F* to the inequality, an equivalent perfect solution can be obtained by splitting states of *F* [4]. However, if we do not merge equivalent states when deriving the automaton $\Lambda(C,S,MAX)$ then the largest complete submachine *M* of an FSM corresponding to the automaton (if it exists) is perfect w.r.t. the context *C*. However, if the obtained FSM *M* is not reduced then the reduced form of *M* does not generally possess the property.

Theorem 1: Given an FSM equation $C \lozenge X \cong S$ (inequality $C \lozenge X \leq S$), the largest complete submachine *M* of an FSM corresponding to the automaton $\Lambda(C,S,MAX)$ (without merging equivalent states) is a perfect solution (w.r.t. the context *C*) to the inequality $C \lozenge_{Ext} X \leq S$.

Theorem 2: Let *P* be a perfect solution (w.r.t. the context *C*) to the inequality $C \lozenge X \leq S$. 1. Every complete submachine P_{sub} of *P* is perfect. 2. A complete intersection of *P* and some FSM is perfect.

Algorithm 1. Deriving a largest complete progressive solution to an FSM equation (inequality)

Input: Observable FSMs *C* and *S*.

Output: A largest progressive solution to $C \lozenge X \cong S$ (if it exists).

Step-1. Derive the largest complete submachine *M* of the FSM corresponding to the automaton $\Lambda(C,S,MAX)$. If *M* does not exist or *M* is not a solution to the equation, then the equation has no complete solution. End Algorithm 1. Otherwise, construct the intersection $Aut(C)\uparrow_{I2\cup O2} \cap Aut(M)\uparrow_{I1\cup O1} \cap Aut(MAX(I,O))\uparrow_{Int}$ and Go-to Step-2.

Step-2. If there is no triplet in the intersection where the language generated at the triplet is empty and there is no accepting triplet (c,m,t) where at least one transition under an external input is undefined, then Go-to Step-4. Otherwise; Go-to Step-3.

Step-3. Iteratively delete from the intersection every triplet (c,m,t) where the external restriction of language generated at the triplet is empty and each accepting triplet (c,m,t) where at least one transition under an external input is undefined.

- If the initial state of the intersection is deleted then there is no progressive solution. End Algorithm 1. Otherwise,
- For each deleted triplet (c,m,t), delete state m from the FSM M and iteratively delete from M states where at least one input is undefined.
  If the initial state of M is deleted then there is no progressive solution. End Algorithm 1. When a state m is deleted from the FSM M each triplet (c,m,t) is deleted from the intersection. If the initial state of the intersection is deleted then there is no progressive complete solution. End Algorithm 1. Otherwise, Go-to Step-2.

Step-4. If $C \lozenge M \cong S$ then M is a largest progressive solution. End Algorithm 1. If $C \lozenge M \ncong S$ then there is no progressive solution. End Algorithm 1.

Theorem 3: If the equation $C \lozenge X \cong S$ has a progressive solution then Algorithm 1 returns a largest progressive solution.

As an example, consider a largest solution M shown in Fig. 2. In order to derive a largest progressive solution to the equation, at Step-2, derive the intersection in Fig. 3. Let $Aut(F)$ denote the obtained intersection. State (10,M,B) of $Aut(F)$ is non-progressive, i.e., delete this state from $Aut(F)$ and correspondingly delete state M from $Aut(M)$. Furthermore, state (9,L,B) is also non-progressive, i.e., delete this state from the automaton and correspondingly delete state L from $Aut(M)$. The remaining states of the obtained automaton are all progressive. End Algorithm 1. The FSM corresponding to the resulting automaton $Aut(M)$ is a submachine of the FSM in Fig. 2 without state L, and this FSM is a largest complete progressive solution to the equation.

4. A System of FSM Equations

Given an integer $k>1$, complete context FSMs $C_i = (C_i, I_1 \cup V, O_1 \cup U, T_{Ci}, c_{i0})$, specifications $S_i = (S_i, Ext_{in}, Ext_{out}, T_{Si}, s_{i0})$, $k > 1$, and a system of equations $C_i \lozenge_{Ext} X \cong S_i$, $i = 1, ..., k$. An FSM X over the input alphabet $Ext_{in} \backslash I_1 \cup U$ and over the output alphabet $Ext_{out} \backslash O_1 \cup V$ is a solution to the system if it is a solution to each equation, i.e., $C_i \lozenge_{Ext} X \cong S_i$, $i = 1, ..., k$.

If a system of FSM equations has a complete solution then the system has a largest complete solution. A complete solution to the system $C_i \lozenge_{Ext} X \cong S_i$, i = 1, ..., k, is called a *largest complete solution* if it includes all complete solutions as reductions. A largest complete solution M to a system of equations $C_i \lozenge_{Ext} X \cong S_i$ can be obtained similar to that in [5]. Given a system of equations $C_i \lozenge_{Ext} X \cong S_i$, i = 1, ..., k, a solution B to the system is called *progressive* if it is a progressive solution to every equation of the system. In general, the intersection of two largest progressive solutions to two equations not necessary is a progressive solution to the system of two equations.

Algorithm 2. Deriving a largest progressive solution to a system of FSM equations

Input: Observable FSMs C_i and S_i, i = 1, ..., k.

Output: A largest progressive solution over the input alphabet $I_2 \cup U$ and output alphabet $O_2 \cup V$ to the system $C_i \lozenge_{Ext} X \cong S_i$, i = 1, ...,k (if a progressive solution exists).

Step-1: For i = 1, …, k, call Algorithm 1 and obtain a largest progressive solution M_i to the equation $C_i \lozenge_{Ext} X \cong S_i$. If for some i = 1, ..., k, there is no progressive solution to the equation $C_i \lozenge_{Ext} X \cong S_i$, then there is no progressive solution to the system of equations, End Algorithm 2. Else; Go-to Step-2.

Step-2: Derive the largest complete submachine F of the intersection $\cap M_i$. If the intersection has no complete submachine, then there is no progressive solution to the system of equations, End Algorithm 2. Else, Go-to Step-3.1.

Step-3.1) If F is a progressive solution to each equation, then F is a largest progressive solution to the system of equations. End Algorithm 2. Else, Go-to Step-3.2.

Step-3.2) For every $j \in$ 1, ..., k such that F is not a progressive solution to an equation $C_j \lozenge_{Ext} X \cong S_j$, assign $M = F$, construct the intersection $Aut(C_j)\uparrow_{I2\cup O2} \cap Aut(M)\uparrow_{I1\cup O1} \cap Aut(MAX(I,O))\uparrow_{Int}$ and call Steps 2 and 3 of Algorithm 1 in order to derive a largest complete submachine F_j of F that is a progressive solution to the equation. If at least for one equation there is no such submachine then the system of equations has no progressive solution; END Algorithm 2. Else, assign $M_j := F_j$ and Go-to Step-2.

Theorem 4: If a system of equations $C_i \lozenge_{Ext} X \cong S_i$, i = 1, ..., k, has a progressive solution then Algorithm 2 returns a largest progressive solution.

As an example, consider the specification S_1 [with $(1,x,o_3,1)(1,i,o_1,1)$ $(1,x,o_2,1)$] and the context C_1 which is that of Fig. 1 where the output label o_3 of the transition $(9,v_1,o_3,1)$ is changed to o_1. Moreover, consider the specification S_2 with a single state 1 and transitions $(1,x,o_3,1)(1,i,o_1,1)$ $(1,x,o_3,1)$. Consider also, the context C_2 shown in Fig. 4 and the system of two equations $C_1 \lozenge_{Ext} X \cong S_1$ and $C_2 \lozenge_{Ext} X \cong S_2$. For each of these equations, at Step-1 of Algorithm 2, apply Algorithm 1 and obtain the largest complete progressive solutions LP_1 and LP_2. The intersection of these solutions is

shown in Fig. 5. The corresponding FSM F is not a progressive solution to the equation $C_1 \lozenge_{Ext} X \cong S_1$, since states (9,LS,B), (10,MW,B), and (9,LT,B) of the intersection in Fig. 6 are non-progressive. Correspondingly, in Step-3.2, in order to derive a largest complete submachine of F that is a progressive solution to the equation $C_1 \lozenge_{Ext} X \cong S_1$, derive the intersection in Fig. 6 and apply Steps 2 and 3 of Algorithm 1. States (9,LS,B), (10,MW,B) and (9,LT,B) of the intersection are non-progressive, thus, delete states LS, MW, and LT from *Aut*(*F*). The obtained automaton *Aut*(*F*) is that of Fig. 5 without deleted states LS, MW, and LT and its corresponding FSM is a complete progressive solution to the system of two equations.

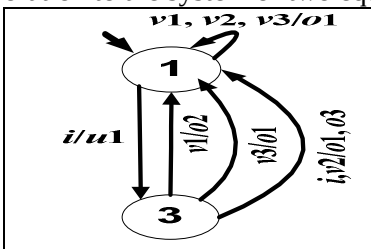

Fig. 4. Context C_2

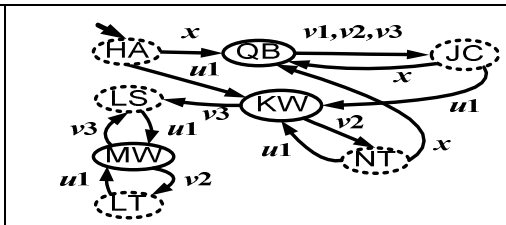

Fig. 5. The intersection *Aut*(*F*)

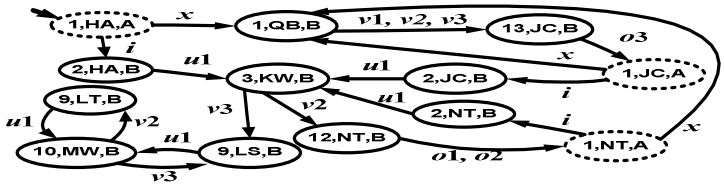

Fig. 6. The intersection $Aut(C_1)\uparrow_{(I2\cup O2)} \cap Aut(F)\uparrow_{(I1\cup O1)} \cap Aut(MAX(I,O))\uparrow_{Int=U\cup V}$

References

1. El-Fakih, K., Yevtushenko, N., Buffalov, S., Bochmann, G.: Progressive Solutions to a Parallel Automata Equation. Theoretical Computer Science. 362, 17--32 (2006)
2. Petrenko, A., Yevtushenko, N.: Solving Asynchronous Equations. In: Formal Description Techniques and Protocol Specification, Testing, and Verification, pp. 231--247 (1998)
3. Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A.,Sangiovanni-Vincentelli, A.: Solution of Parallel Language Equations for Logic Synthesis. In: *ICCAD*, pp. 103--110 (2001)
4. Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., Sangiovanni-Vincentelli, A: Compositionally Progressive Solutions of Synchronous FSM Equations. Discrete Event Dynamic Systems. 18(1): 51--89 (2008)
5. Yevtushenko, N., Zharikova, S., Vetrova, M.: Multi Component Digital Circuit Optimization by Solving FSM Equations. In: Euromicro Symposium on Digital System Design, IEEE Computer society, pp. 62--68 (2003)
6. Hopcroft J., Ullman, J.: Introduction to automata theory, Languages, and Computation. Addison-Wesley (1979)