Solving Parallel Multi Component Automata Equations

Natalia Shabaldina, Nina Yevtushenko

Abstract — **The problem of designing the unknown component of a system of interacting automata that combined with the known part of the system meets the specification, is well known. However, most publications are devoted to solving the problem for the proper composition of two automata. In this paper, we consider a parallel multi component automata equation and propose two methods for deriving a largest solution to this equation (if the equation is solvable). In particular, we show that the union of alphabets over all known components and of that of the specification is the largest alphabet of actions over which a solution for a solvable equation should exist, and show how a solution over an appropriate alphabet can be derived from such a largest solution.**

Index Terms — **Automata, Equations, Discrete event systems**

I. Introduction

Many problems over discrete event systems can be reduced to solving a language inequality $A \mathbin{@} X \subseteq S$ or to solving a language equation $A \mathbin{@} X = S$ where X is a free variable and @ is the composition operator. For different applications, appropriate equations are formulated and their solutions were investigated by various researchers. Most papers in process algebra [see, for example, 1-5] are devoted to solving equations over parallel composition which allows arbitrary delay between communication events. In this paper, we consider a parallel multi component automata equation and propose two methods for deriving a largest solution to this equation (if the equation is solvable). In particular, we show that the union of alphabets over all known components and of that of the specification is the largest alphabet of actions over which a solution for a solvable equation should exist, and show how a solution over an appropriate alphabet can be derived from such a largest solution.

Manuscript received March 17, 2008. F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: author@ boulder.nist.gov).

Natalia Shabaldina is with the Tomsk State University, 36 Lenin Str., Tomsk, 634050, Russia (e-mail: snv@kitidis.tsu.ru)

Nina Yevtushenko is with the Tomsk State University, 36 Lenin Str., Tomsk, 634050, Russia (e-mail: yevtushenko@elefot.tsu.ru).

Acknowledgments. The authors gratefully acknowledge the partial support of the RFBR-NSC Grant 06-08-89500.

II. Parallel equations over automata

A. Parallel composition operator

Let A be an alphabet, and a language L is defined over the alphabet A. Given a non-empty subset A_1 of the alphabet A and a sequence α over alphabet A, the A_1-*restriction* of α, denoted $\alpha_{\Downarrow A_1}$, is a sequence obtained from α by erasing symbols in $A \backslash A_1$. The language $L_{\Downarrow A_1} = \{\beta_{\Downarrow A_1} : \beta \in L\}$ is the *restriction* of language L onto the subset A_1. For each word $\beta \in L$ that does not have symbols of alphabet A_1 the restriction of β is the empty word ε. Let now language L be defined over alphabet $A_1 \subseteq A$. The language $L_{\Uparrow A} = \{\beta : \beta_{\Downarrow A_1} \in L\}$ is the *expansion* of language L over the alphabet A.

In this paper, we consider equation solving over finite automata. A *finite automaton* (or simply an *automaton* throughout this paper) is a quintuple $\mathcal{S} = \langle S, A, \delta_S, s_0, F_S \rangle$, 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, A is an alphabet of actions, and $\delta_S \subseteq A \times S \times S$ is a transition relation. We say that there is a transition from a state s to a state s' labeled with an action a, if and only if the triple (a,s,s') is in the transition relation δ_S. The automaton $\mathcal{S}$ is called *deterministic*, if for each state $s \in S$ and any action $a \in A$ there exists at most one state s', such that $(a,s,s') \in \delta_S$. If $\mathcal{S}$ is not deterministic, then it is called *nondeterministic*. As usual, the transition relation δ_S of the automaton $\mathcal{S}$ can be extended to sequences over the alphabet A. Given a state s of the automaton $\mathcal{S}$, the set $L_s(\mathcal{S}) = \{\alpha \in A^* \mid \exists s' \in F_S \, ((s, \alpha, s') \in \delta_S)\}$ is called the language, *generated at the state* s. The language, generated by the automaton $\mathcal{S}$ at the initial state, is called the *language generated* or *accepted by the automaton* $\mathcal{S}$ and is denoted by $L(\mathcal{S})$, for short. Automata $\mathcal{S}$ and $\mathcal{T}$ are called *equivalent* ($\mathcal{T} \cong \mathcal{S}$) if $L(\mathcal{T}) = L(\mathcal{S})$. Automaton $\mathcal{T}$ is a *reduction* of automaton $\mathcal{S}$ ($\mathcal{T} \leq \mathcal{S}$) if $L(\mathcal{T}) \subseteq L(\mathcal{S})$. In the same way, the equivalence and the reduction relations are defined between two states of an automaton. Well-known results state that for each non-deterministic automaton, there exists an equivalent deterministic automaton [6]. It also is well-known how the union, intersection, complementation, restriction and expansion over deterministic automata [4-6] can be derived.

Given k automata $F_1, F_2, \ldots, F_k$, let the automaton F_j accept the language L_j, $j = 1, 2, \ldots, k$, over alphabet A_j, $A = A_1 \cup A_2 \cup \ldots \cup A_k$ and E is a non-empty subset of A. The *parallel* composition $\Diamond_E(F_1, F_2, \ldots, F_k)$ is the automaton $(F_{1\Uparrow_A} \cap F_{2\Uparrow_A} \cap \ldots \cap F_{k\Uparrow_A})_{\Downarrow_E}$.[1] The automaton $\Diamond_E(F_1, F_2, \ldots, F_k)$ has the empty language if one component automaton has the empty language.

B. Parallel language equations

In this section, we extend the notion of an automata equation to k automata, $k > 2$, and determine a largest alphabet over which the equation should be solvable.

Extending the formula for a largest solution to a multi component automata equation

Given k automata $F_1, F_2, \ldots, F_{k-1}, F$, let the automaton F_j accept the language L_j, $j = 1, 2, \ldots, k - 1$, over alphabet A_j, while the automaton F accepting the language L over alphabet E, $A = A_1 \cup A_2 \cup \ldots \cup A_{k-1} \cup E$ and R is a non-empty subset of A. Consider an *automata inequality* $\Diamond_E(F_1, F_2, \ldots, X) \leq F$ and an *automata equation* $\Diamond_E(F_1, F_2, \ldots, X) \cong F$ with a free variable X that is an automaton over alphabet R. An automaton F_R over the alphabet R is a *solution* to the inequality $\Diamond_E(F_1, F_2, \ldots, F_R) \leq F$ if $\Diamond_E(F_1, F_2, \ldots, F_R) \leq F$. An automaton F_R is a *solution* to the equation $\Diamond_E(F_1, F_2, \ldots, F_R) \cong F$ if $\Diamond_E(F_1, F_2, \ldots, F_R) \cong F$. A solution M_R over the alphabet R is a *largest* solution to the inequality $\Diamond_E(F_1, F_2, \ldots, X) \leq F$ (to the equation $\Diamond_E(F_1, F_2, \ldots, X) \cong F$) if each solution over alphabet R is a reduction of M_R.

Similar to a parallel equation over two automata, a multi component parallel automata inequality as well as a solvable parallel automata equation has always a largest solution[2].

Theorem 1. 1. Given k automata $F_1, F_2, \ldots, F_{k-1}, F$, let the automaton F_j accept the language L_j, $j = 1, 2, \ldots, k-1$, over alphabet A_j, while the automaton F accepting the language L over alphabet E, $A = A_1 \cup A_2 \cup \ldots \cup A_{k-1} \cup E$ and R is a non-empty subset of A. A largest solution to the automata inequality $\Diamond_E(F_1, F_2, \ldots, X) \leq F$ is the automaton $\overline{\langle\rangle_R(F_1,\ldots,F_{k-1},\overline{F})}$. 2. Given a language equation $\Diamond_E(F_1, F_2, \ldots, X) \cong F$, the equation is solvable if and only if $\Diamond_E(F_1, F_2, \ldots, \overline{\langle\rangle_R(F_1,\ldots,F_{k-1},\overline{F})}) \cong F$. If the equation is solvable then the automaton $\overline{\langle\rangle_R(F_1,\ldots,F_{k-1},\overline{F})}$ is a largest solution to the equation.

Determining a largest alphabet over which the equation is solvable

The formula of the previous section has the exponential complexity. However, the complexity of checking if there exists an alphabet s.t. the equation is solvable over this alphabet can be reduced. In this section, we propose an algorithm for deriving a largest solution over alphabet A without using the complementation and restriction operators and state that if the equation has no solution over alphabet A then the equation is not solvable over any subset R of the alphabet A.

A largest solution to the inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$ over alphabet A can be derived using the following procedure. Derive the automaton M_A by adding the designated accepting Don't Care state (*DNC*) to the set of states of $F_{1\Uparrow_A} \cap \ldots \cap F_{k-1\Uparrow_A} \cap F_{\Uparrow_A}$. Given a state $f_1 \ldots f_{k-1} f$ of the automaton $F_{1\Uparrow_A} \cap \ldots \cap F_{k-1\Uparrow_A} \cap F_{\Uparrow_A}$ and an action $a \in A$, the automaton M_A has a transition $(f_1 \ldots f_{k-1} f, a, \text{DNC})$ if there exists a component automaton F_j that has no transition under a from state f_j. There is a loop at the DNC state for each action $a \in A$.

Theorem 2. 1. Given an automata inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$, the automaton M_A is a largest solution to the inequality. 2. Given an automata equation $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \cong F$, if the automata $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, M_A)$ and F are equivalent then M_A is a largest solution to the equation. If the automata $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, M_A)$ and F are not equivalent then the equation is not solvable over any alphabet $R \subseteq A$.

The following proposition gives a guide how a solution can be derived over a subset $R \subset A$ (if such a solution exists).

Proposition 3. Given an automata inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$, a largest solution M_A to the inequality over alphabet A and a proper subset $R \subset A$, an automaton M_R over alphabet R is a solution to the inequality over alphabet R if and only if the expansion of $M_{R\Uparrow_A}$ is a reduction of M_A.

Corollary. Given an automata inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$, a largest solution M_A to the inequality over alphabet A and a proper subset $R \subset A$, a largest solution to the inequality over alphabet R is a largest automaton M_R (w.r.t. the language) over alphabet R s.t. the expansion of $M_{R\Uparrow_A}$ is a reduction of M_A.

Based on the above statements a largest solution M_R over alphabet R to the automata inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$ can be derived in the following steps.

Step 1. Derive a largest solution M_A to the inequality $\Diamond_E(F_1, F_2, \ldots, F_{k-1}, X) \leq F$ over alphabet A. If a largest solution M_A is trivial (the language is empty) then a solution over each alphabet is trivial.

Step 2. Derive the largest complete submachine N of M_A over alphabet $A \backslash R$. If there is no complete submachine then the inequality has only a trivial solution over alphabet R. Otherwise, let N be a set of states of N. Delete each transition in M_A to every state that is not in the set N. Denote P_A the obtained automaton.

[1] This definition can be viewed as an extension of the parallel composition of two automata [4]

[2] With respect to the reduction relation.

Step 3. Derive the set K_1 of all states reachable in M_A from the initial state under actions of the alphabet $A\backslash R$. Assign $K = \{K_1\}$; $i = 1$; M_R be a trivial automaton with the initial state K_1 over the alphabet R.

Step 4. For each action $a \in R$ do:

if the transition under a is defined at each state of the set K_i then derive the set D_i of all states where a takes M_A from states of the set K_i and the set L_i of all states reachable in M_A from the states of the set D_i under actions of the alphabet $A\backslash R$. Add a transition (K_i, a, L_i) to the automaton M_R and if $L_i \notin K$ then add L_i to the set K.

Step 5. Increment i by 1. If $i < |K|$ then Step 4. Else END; the automaton M_R is a largest solution to the inequality over alphabet R.

Theorem 4. 1. Given an automata inequality $\Diamond_E(F_1, F_2, \ldots, F_{k\text{-}1}, X) \le F$, a largest solution M_A to the inequality over alphabet A and a proper subset $R \subset A$, the automaton M_R obtained in Step 5 of the above algorithm is a largest solution to the inequality over alphabet R. 2. If $\Diamond_E(F_1, F_2, \ldots, F_{k\text{-}1}, M_R)$ is equivalent to F then M_R is a largest solution over alphabet R to the equation $\Diamond_E(F_1, F_2, \ldots, F_{k\text{-}1}, M_R) \cong F$; otherwise the equation has no solution over alphabet R.

We also remind that not each subset of the largest solution to a parallel language equation inherits the property to be a solution and the complete characterization of the set of solutions to a parallel language equation is still an open issue.

III. Example

In this section, we briefly consider a simple example. We solve the equation $\Diamond_E(F_1, X) \cong F$ for automata F_1 and F in Figures 1 and 2 where all states are accepting states. A largest solution over $A = \{x, o, u, v\}$ is shown in Figure 3.

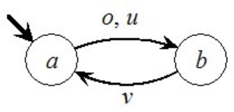

Fig. 1. Automaton F

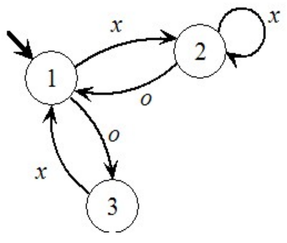

Fig. 2. Automaton F_1

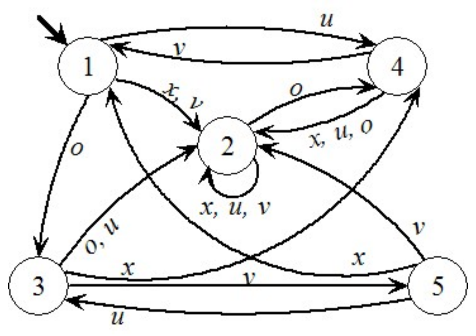

Fig. 3. Largest solution over $A = \{x, o, u, v\}$ (after merging equivalent states)

We now derive a largest solution over $R=\{x,u,v\}=A\backslash\{o\}$. A transition at state 5 under o is undefined; thus, we delete state 5. The initial state of the obtained automaton is $K_1 = \{1, 2, 3, 4\}$. There is no transition from this state under v, as the transition under v is undefined at state 3. However, there is a transition (K_1, x, K_2), where $K_2 = \{2, 4\}$. The resulting automaton is shown in Figure 4.

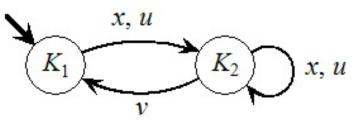

Fig. 4. A largest solution over alphabet $R = \{x, u, v\}$

IV. Conclusion

In this paper we have studied the problem of solving multi component language equations over a parallel composition operator. In particular, the largest alphabet over which a solution exists, is characterized. A number of restricted solutions which are considered for a parallel equation over two automata [5] can be defined in the same way for multi component automata inequalities and equations. The complexity of the proposed method seems to be lower than that of the known method since we do not use the automata complements in our method.

References

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

[2] H. Qin, P. Lewis, "Factorisation of Finite State Machines under strong and observational equivalences", *Formal Aspects of Computing*, 3:284–307, Jul.–Sept. 1991.

[3] A. Petrenko, N. Yevtushenko, "Solving asynchronous equations", *Formal description techniques/Protocol specification, testing and verification*. Kluwer Academic Publishers, 1998, pp. 125–140.

[4] K. El-Fakih, N. Yevtushenko, S. Buffalov, G. v. Bochmann, "Progressive solutions to a parallel automata equation", *Theoretical Computer Science*, 362, 2006, pp. 17-32 (The preliminary version was published in *Lectures Notes in Computer Science*, 2003, vol. 2767, pp. 367 – 382).

[5] N.Yevtushenko, T.Villa. R.K.Brayton, A.Petrenko, A.Sangiovanni-Vincentelli, "Solving a parallel language equation", *In Proceedings of the ICCAD'01*, USA, 2001. pp. 103-110.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, Languages, and Computation*, Addison-Wesley, 1979.