

## ANALYSIS OF THE BEHAVIOR OF CONCURRENT SYSTEMS IN DIFFERENT ENVIRONMENTS

A. PETRENKO, *Professor, Dr.hab.ing. (Computer Sciences)*  
*Centre de Recherche Informatique de Montreal*  
*550 Sherbrooke West, Suite 100*  
*Montreal H3A 1B9 (Canada)*

A. ULRICH, *Dr. Ing.*  
*Siemens AG, Corporate Technology*  
*ZT SE 1, 81730 Munich (Germany)*

V. P. CHAPENKO, *Senior Investigator, Dr.Sc. (Computer Sciences)*  
*Institute of Electronics and Computer Science*  
*University of Latvia*  
*Dzerbenes 14, LV-1006, Riga (Latvia)*

**A method of conflict detection in concurrent systems specified in the form of a set of interacting systems of labeled transitions is considered. Models of the behavior of concurrent systems that interact with a sequential-fast external environment and with a concurrent external environment are developed. Propositions that evaluate the conflict detection method considered in the article are presented.**

### 1. INTRODUCTION

The behavior of concurrent systems when races occur in them may be nondeterministic and differ from their required behavior. Races are the source of errors in concurrent software that are difficult to detect in the course of test execution. To detect races, test runs of the same test must be repeatedly executed with a variable rate of execution of the test events. As an alternative approach to such testing protocols, it was suggested in [1, 2] that potential races in a concurrent systems could be detected on the basis of the type of external environment in which the system functions and from a specification of this systems, given in the form of a set of interacting finite labeled transition systems. Each labeled transition system describes the behavior of a corresponding component of the concurrent system and is characterized by a set of states, a set of actions, and a set of transitions. The concept of an *action* does not distinguish between an event at the input of a component and activity at its output, moreover, an action may denote the reception or transmission of a message as well as, for example, the execution of a local operation. The term *transition* denotes a change in the state of a labeled transition system induced by an action. Two concepts are introduced in [1] for the analysis of the behavior of a concurrent system, that of an *action race* and that of an *action race condition*. By an action race condition is understood a situation in which more than one action is enabled in one of the global states of a concurrent system. If in such a situation, and assuming these actions have different orders of execution, the system may attain different stable states, this will then mean that there exists an action race in the concurrent system. Despite the fact that there exists an action race condition in some global state, an action race (in this state) will not arise if, whatever the order of execution of the actions that are enabled in this state, the system attains only a single stable state that coincides with the required state according to the specification. The following three propositions concerning the environment in which a concurrent system functions were discussed in [1]: a *sequential-slow environment*, a *sequential-fast environment*, and a concurrent environment. In the analysis of action race conditions, a distinction is made between internal actions in the system and external actions (actions between the environment and the system) [1]. Three types of race conditions are distinguished, depending

on the form of the actions that create the race conditions: race conditions between internal actions, or *i-race* conditions; race conditions between external actions, or *e-race* conditions; and mixed (internal and external) race conditions, or *m-race* conditions. In [1] a method of detection of action races in a system controlled by a sequential-slow environment was proposed. Such an environment, before generating a succeeding external action intended for the concurrent system, remains in a wait state until the system completes execution of internal actions. However, the assumption of a sequential-slow environment in which only *i-race* conditions may exist is a very strict assumption. A sequential-fast environment that may successively execute individual external actions before the moment the system attains a succeeding stable state is a more realistic environment. Here mixed race conditions may arise. A concurrent environment may simultaneously generate several external actions that initiate a transition into a system found in a stable state. *e-race* conditions may arise in such an environment, in addition to *m-race* conditions.

In the present article, models of the behavior of concurrent systems that interact with a sequential-fast environment and with a concurrent environment intended for action race detection are investigated. The article is a natural continuation of a previous study [1] and the necessary definitions and concepts from the latter article are presented in the next section. The article is based on part on [2].

## 2. NECESSARY CONCEPTS [1]

A *parallel system*  $\mathfrak{S}$  is modeled by a set of  $n$  finite labeled transition systems that interact synchronously on the basis of the rendezvous method [1]. A labeled transition system  $M$  is characterized by the quadruple

$$(S, A \cup \{\tau\}, \rightarrow, s_0), \quad (1)$$

where  $S$  is a finite set of states;  $A$ , a finite set of actions;  $\rightarrow \subseteq S \times (A \cup \{\tau\}) \times S$ , transition relation;  $s_0 \in S$ , an initial state; and  $\tau$ , an unobservable action of  $M$ .

The transition  $(s_1, a, s_2) \in \rightarrow$  is also written in the form  $s_1 - a \rightarrow s_2$ . The notation  $s = a \Rightarrow s'$  denotes the sequence

$$s - \tau \rightarrow \dots - a \rightarrow \dots - \tau \rightarrow s', \quad (2)$$

while the notation  $s = \sigma \Rightarrow s'$  expresses the sequence

$$s = a_1 \Rightarrow s_1 = a_2 \Rightarrow s_2 = \dots \Rightarrow s', \quad (3)$$

where  $\sigma = a_1 a_2 \dots$ . The sequence of actions that have been realized in transitions is called a *route*. For each state  $s'$  in a labeled transition system there exists a route  $\sigma$  such that  $s_0 = \sigma \Rightarrow s'$ . Since in the general case a labeled transition system is nondeterministic, state  $s'$ , which is attained following execution of route  $\sigma$ , may not be unique. Therefore, the notation ( $s$  after  $\sigma$ ) denotes the set

$$\{s' \mid s = \sigma \Rightarrow s'\}. \quad (4)$$

Below, we will be considering concurrent systems consisting of deterministic systems  $M$  without action  $\tau$ . A *set of routes of state  $s$*  is represented in the form

$$\{\sigma \in A^* \mid \exists s' \in S: s = \sigma \Rightarrow s'\} = \mathbf{Tr}(s), \quad (5)$$

and the set of routes of the system  $M$  as  $\mathbf{Tr}(s_0)$ . It is assumed that the set of actions of system  $\mathfrak{S}$  satisfies the condition

$$A_{\mathfrak{S}} = A_1 \cup \dots \cup A_n. \quad (6)$$

Let  $N$  be a set of indices  $\{1, \dots, n\}$  assigned to each labeled transition system in the concurrent system  $\mathfrak{S}$ . For action  $a \in A_{\mathfrak{S}}$ , the notation  $i(a)$  denotes the set of occurrences  $\{i \in N \mid a \in A_i\}$  of action  $a$ , that is, the set of indices of

those labeled transition systems which are implicated in a rendezvous relative to action  $a$ . Suppose at least one action  $a$  such that  $|i(a)| > 1$  is assigned to each component in the concurrent system  $\mathfrak{S}$  and let a subset of actions  $A_e \subseteq A_{\mathfrak{S}}$  be given. Actions that belong to  $A_e$  are called *external* actions if

$$A_e \supseteq \{a \in A_{\mathfrak{S}} \mid |i(a)| = 1\}. \quad (7)$$

The set of actions in  $\mathfrak{S}$  that do not belong to  $A_e$  forms the set of *internal* actions  $A_i = A_{\mathfrak{S}} \setminus A_e$  of the system  $\mathfrak{S}$ . We next introduce the concepts of *internal* or *external* transitions realized under the effect of internal or external actions, respectively. Let

$$\text{enabled}(s) = \{a \mid \exists s' \in S_{\mathfrak{S}}^*: (s, a, s') \in \rightarrow_{\mathfrak{S}}\} \quad (8)$$

denote the set of actions that belong to global transitions and are enabled in global state  $s$ . A stable global state, transient global state, and deadlock state are states  $s$  in which the set of enabled actions satisfy the following corresponding expressions:

$$\text{enabled}(s) \subseteq A_e; \quad \text{enabled}(s) \neq \emptyset \quad \text{and} \quad \text{enabled}(s) \not\subseteq A_e; \quad \text{enabled}(s) = \emptyset. \quad (9)$$

The composite machine

$$C_{\mathfrak{S}} = M_1 \parallel \dots \parallel M_n \quad (10)$$

of the concurrent system  $\mathfrak{S}$  represented by  $n$  labeled transition systems of the form

$$M_i = (S_i, A_i, \rightarrow_i, s_{0i}) \quad (11)$$

is called a labeled transition system

$$(S_{\mathfrak{S}}, A_{\mathfrak{S}}, \rightarrow_{\mathfrak{S}}, s_{0\mathfrak{S}}), \quad s_{0\mathfrak{S}} = (s_{01}, \dots, s_{0n}), \quad (12)$$

where  $s_{0\mathfrak{S}}$  is the initial global state,

$$S_{\mathfrak{S}} \subseteq S_1 \times \dots \times S_n; \quad A_{\mathfrak{S}} \subseteq A_1 \cup \dots \cup A_n; \quad \rightarrow_{\mathfrak{S}} \subseteq S_{\mathfrak{S}} \times A_{\mathfrak{S}} \times S_{\mathfrak{S}} \quad (13)$$

are the smallest sets that may be produced as a result of execution of the parallel composition operator in the concurrent system  $\mathfrak{S}$ .

The notation ( $s$  after-ext  $a$ ) denotes the set of all possible stable global states that may be attained by  $\mathfrak{S}$  after external action  $a$  has been executed in stable global state  $s$ , i.e.,

$$(s \text{ after-ext } a) = \{s' \mid \exists \sigma \in A_i^*: s' \in (s \text{ after } a\sigma) \wedge \text{enabled}(s') \subseteq A_e\}. \quad (14)$$

A slow-composite machine

$$Sc_{\mathfrak{S}} = M_1 \downarrow A_e \downarrow \dots \downarrow A_e \downarrow M_n \quad (15)$$

of a concurrent system  $\mathfrak{S}$  is a labeled transition system

$$(S_{Sc}, A_{Sc}, \rightarrow_{Sc}, s_{0Sc}), \quad (16)$$

where  $s_{0Sc} = s_{0\mathfrak{S}}$  is the initial stable state,

$$S_{Sc} \subseteq S_1 \times \dots \times S_n, \quad A_{Sc} \subseteq A_1 \cup \dots \cup A_n, \quad \text{и} \quad \rightarrow_{Sc} \subseteq S_{\mathfrak{S}} \times A_{\mathfrak{S}} \times S_{\mathfrak{S}} \quad (17)$$

are the smallest sets that may be produced as a result of execution of operator  $\lfloor A_e \rfloor$  which correlates with the concurrent system  $\mathfrak{S}$  its  $A_e$ -slow composition [1].

A *stable composite machine* is a labeled transition system

$$St_{\mathfrak{S}} = (S_{St}, A_{St}, \rightarrow_{St}, s_{0St}), \quad (18)$$

where  $s_{0St} = s_{0Sr}$ ;

$$S_{St} \subseteq S_{Sc}, A_{St} \subseteq A_e, \text{ and } \rightarrow_{St} \subseteq S_{St} \times A_e \times S_{St} \quad (19)$$

are the smallest sets that may be produced by application of the following rule:

$$(s, a, s') \in \rightarrow_{St} \Leftrightarrow s' \in (s \text{ after-ext } a). \quad (20)$$

A stable composite machine  $St_{\mathfrak{S}}$  may be produced from a slow composite machine by means of the equivalence transformation described in [3]. If as a result of the transformation, a nondeterministic stable composite machine is obtained, this will mean that there exists races in the concurrent system  $\mathfrak{S}$  controlled by a sequential-slow environment.

### 3. SEQUENTIAL-FAST ENVIRONMENT

Let us suppose that the concurrent system  $\mathfrak{S}$  interacts with a sequential-fast environment that may successively process external actions before  $\mathfrak{S}$  attains a succeeding stable state. In this case the sequence of external actions may produce races even if none of the individual actions of the sequence lead to races. In some transient global state of  $\mathfrak{S}$  both an external and an internal transition may be enabled. This may also lead to a race. Suppose

$$\sigma = a_1 \dots a_k \in A_e^* \quad (21)$$

is a sequence of  $k$  external actions in let  $\text{Tr}_{St_{\mathfrak{S}}}(s)$  denote the set of routes of state  $s$  in the stable composite machine  $St_{\mathfrak{S}}$ . We wish to consider the stable global state  $s$  and the sequence  $\sigma$  of the form (21) such that  $\sigma \in \text{Tr}_{St_{\mathfrak{S}}}(s)$ . We denote by  $(s \text{ after-ext } \sigma)$  the set of all possible stable global states that may be attained by composite machine  $C_{\mathfrak{S}}$  following execution by a sequential-fast environment of an external route  $\sigma$  beginning in state  $s$  of machine  $C_{\mathfrak{S}}$ :

$$\begin{aligned} (s \text{ after-ext } \sigma) &= \\ &= \{s' \mid \exists \beta_1, \dots, \beta_k \in A_i^* : s' \in (s \text{ after } a_1 \beta_1 \dots a_k \beta_k) \wedge \text{enabled}(s') \subseteq A_e\}. \end{aligned} \quad (22)$$

**Definition 1.** Suppose a stable state  $s$  and external route  $\sigma \in \text{Tr}_{St_{\mathfrak{S}}}(s)$  are given;

- route  $\sigma$  creates races in  $s$  if

$$|(s \text{ after-ext } \sigma)| > 1; \quad (23)$$

- $\sigma$  is race-free in  $s$  if

$$|(s \text{ after-ext } \sigma)| = 1. \quad (24)$$

A concurrent system  $\mathfrak{S}$  that functions in a sequential-fast environment is race-free if in each stable state  $s$  each route  $\sigma \in \text{Tr}_{St_{\mathfrak{S}}}(s)$  is race-free.

If an external route is race-free in some state, it may be executed in this stable state independently of the rate of the method. Moreover, the final stable state will always be the same, regardless of the rate with which these actions are successively executed.

**Proposition 1.** A concurrent system  $\mathfrak{S}$  is race-free within a sequential-fast environment if in each of its local states either only external actions are enabled or only a single internal action is enabled.

The existence of races in concurrent system  $\mathfrak{S}$  controlled by a sequential-fast environment may be verified by construction of a portion of its composite machine  $C_{\mathfrak{S}}$ . Suppose

$$s = (s_1, \dots, s_n) \quad (25)$$

is a stable global state of the composite machine  $C_{\mathfrak{S}}$  and let  $\sigma$  of the form (21) be an external route, moreover,  $\sigma \in \text{Tr}_{S_{\mathfrak{S}}}(s)$ . We denote by  $E(\sigma)$  a linear labeled transition system [2] with  $\kappa + 1$  states and a unique terminating route  $\sigma$ . The following proposition is valid.

**Proposition 2.** Route  $\sigma$  is race-free if and only if the system

$$E(\sigma) \parallel (M_1(s_1) \parallel \dots \parallel M_n(s_n)) \quad (26)$$

has a unique deadlock state.

The notation  $M(s)$  means that  $M$  is initialized in state  $s$ . This proposition shows that the race detection problem is just as difficult to compute as the classical reachability problem.

If in some global (transient) state both an external action as well as an internal action that has been induced by a preceding external action may be executed, this will mean that there exists an  $m$ -race condition in the concurrent system. It is intuitively clear that if in this situation successive external actions of the environment will not induce internal actions in the some components of the concurrent system,  $m$ -races will not occur in the system. Suppose  $(s \text{ act-ext } a)$  denotes the set of all those components of the system that execute internal transitions in response to external action  $a$  that may be applied in stable state  $s$ , i.e.,

$$(s \text{ act-ext } a) = \{i \in i(b) \mid b \in A_i \wedge \exists \beta \in A_i^* : a\beta b \in \text{Tr}_{S_{C_{\mathfrak{S}}}}(s)\}, \quad (27)$$

where

$$\text{Tr}_{S_{C_{\mathfrak{S}}}}(s) \quad (28)$$

is the set of routes of the slow composite machine  $S_{C_{\mathfrak{S}}}$ . Let us consider stable state  $s$  and an external route  $\sigma$  of the form (21). Suppose

$$s_1 \dots s_k s_{k+1} \quad (29)$$

is a corresponding sequence of stable states along which route  $\sigma$  proceeds, where  $s = s_1$ . The following assertion is valid.

**Proposition 3.** Route  $\sigma$  is race-free in  $s$  if for all  $i$ ,  $1 \leq i \leq k$ , action  $a_i$  is race-free in state  $s_i$  and

$$(s_i \text{ act-ext } a_i) \cap \cup_{j < i} (s_j \text{ act-ext } a_j) = \emptyset. \quad (30)$$

#### 4. CONCURRENT ENVIRONMENT

A concurrent environment can simultaneously generate several external actions designed by mutually distinct components of the system  $\mathfrak{S}$ , though only one action will be produced by a particular component. Suppose that the concurrent system  $\mathfrak{S}$  comprises  $n$  components. Then the concurrent environment may simultaneously execute at most  $n$  external actions from  $\mathfrak{S}$  as a single concurrent action. We will represent a concurrent action consisting of  $k$  elementary external actions,  $k \leq n$ , in the form of a labeled transition system containing  $2^k$  states (hypercubes); each completed route of such a system is a single possible ordering (permutation) of all  $k$  external actions.

To form a concurrent action it is necessary that two actions  $a, b \in A_c$  possess disjoint sets of occurrences:

$$i(a) \cap i(b) = \emptyset, \quad (31)$$

that is, actions must be independent. If this requirement is satisfied, it turns into the syntactically sufficient condition for concurrent transitions employed in [4]. This condition may also be employed in the context of the present article: two independent external actions  $a$  and  $b$  may be executed as a concurrent action if in the present stable state, neither  $a$  nor  $b$  can induce execution of subsequent internal actions in the concurrent system  $\mathfrak{S}$ . However, if a concurrent system permits execution of internal actions, there will be no guarantee that two independent external actions executed concurrently will lead to a unique terminal stable state. Hence, a more rigorous condition is needed to define a concurrent action.

**Definition 2.** Suppose a deterministic stable composite machine  $St_{\mathfrak{S}}$  is given. Two external actions  $a$  and  $b$  are *potentially concurrent actions* in stable state  $s \in S_{St}$  if:

1.  $a, b \in \text{enabled}(s)$ ,  $b \in \text{enabled}(s \text{ after}_{St} a)$ ,  $a \in \text{enabled}(s \text{ after}_{St} b)$
  2.  $s = ab \Rightarrow_{St} s'$ ,  $s = ba \Rightarrow_{St} s''$ ,  $s' = s''$ .
- (32)

The above two conditions are similar to the conditions in the relation of independence presented in [4, 5]. The difference is that conditions (32) are formulated in terms of a stable composite machine  $St_{\mathfrak{S}}$  and not in terms of composite machine  $C_{\mathfrak{S}}$  (the latter is usually used in problems of verification). Moreover, in order for potentially concurrent actions to become actually concurrent, they must not create races. Suppose a subset of potentially concurrent external actions  $P$  is given. We denote by  $[P]$  the set of all possible linearizations [4] of external actions that belong to  $P$ .

**Definition 3.** A stable state  $s$  and set  $P$  of external actions that are pairwise potentially concurrent are given,

$$P \subseteq \text{enabled}(s), \quad (33)$$

- the set  $P$  creates races in  $s$  if

$$|\cup_{\sigma \in [P]} (s \text{ after-ext } \sigma)| > 1, \quad (34)$$

- the set  $P$  is race-free in  $s$  if

$$|\cup_{\sigma \in [P]} (s \text{ after-ext } \sigma)| = 1. \quad (35)$$

If the set  $P$  is race-free, we will say that  $P$  is a concurrent action in state  $s$ .

**Proposition 4.** Suppose a stable state  $s$  and set  $P$  of the form (33) are given. The set  $P$  is a concurrent action in  $s$  if and only if all possible linearizations  $\sigma \in [P]$  are race-free in  $s$ .

**Proposition 5.** A stable state  $s$  and external actions

$$a, b \in \text{enabled}(s) \quad (36)$$

are given; the pair  $\langle a, b \rangle$  is a concurrent action in  $s$  if

$$(s \text{ act-ext } a) \cap (s \text{ act-ext } b) = \emptyset. \quad (37)$$

Let us consider a stable state  $s$  of the form (25) and a set  $P$ , or set of external actions that are pairwise potentially concurrent. Let  $E(P)$  denote a system of labeled transitions containing  $2^{|P|}$  states and having a set of completed routes  $[P]$ . The  $e$ -race conditions may be verified by direct construction of a portion of the composite machine, by-passing verification of each linearization. This result is supported by the following assertion.

**Proposition 6.** The set  $P$  is a concurrent action in  $s$  if and only if the system

$$E(P) \parallel (M_1(s_1)) \parallel \dots \parallel (M_n(s_n)) \quad (38)$$

contains a unique deadlock state.

The proofs of the propositions that have been presented here have been carried out, though are not presented in the text due to lack of space. Thus, we have established the essential definitions and assertions necessary for carrying out an analysis of action races in a concurrent system represented by interacting labeled transition systems.

## 5. CONCLUSION

By means of the proposed method of analysis of the behavior of concurrent systems interacting with a sequential-fast environment and with a concurrent environment it is possible to detect action races even at the design stage of the concurrent system in the course of verification and testing of the system. It should also be noted that analysis of races helps in the design of testers for concurrent systems that could execute test actions without races and with the greatest possible speed.

## REFERENCES

- [1] A. Petrenko, A. Ulrich, and V.P. Chapenko, "Action races in concurrent systems," *Avtomatika i Telemekhanika*, no. 6, pp. 46–55, 2002.
- [2] A. Petrenko and A. Ulrich, "Verification and testing of concurrent systems with action races," *Proc. IFIP 13th Intern. Conf. on Testing of Communicating Systems, TestCom'2000*, Ottawa, Canada, 2000, pp. 261–280.
- [3] G. Luo, G. von Bochmann, A. Das, and C. Wu, "Failure-equivalent transformation of transition systems to avoid internal actions," *Information Processing Letters*, vol. 44, pp. 333–343, 1992.
- [4] P. Godefroid, "Partial-order methods for the verification of concurrent systems," *Lecture Notes in Computer Science 1032*, Springer, 1996.
- [5] S. Katz and D. Peled, "Defining conditional independence using collapses," *Theoretical Computer Science*, vol. 101, pp. 337–359, 1992.

Received following revisions 10 October 2002; originally submitted 15 June 2002