Scenario-based Component Behavior Filtration*

Yan Zhang, Xiaofeng Yu, Tian Zhang, Xuandong Li, and Guoliang Zheng

State Key Laboratory of Novel Software Technology
Department of Computer Science and Technology
Nanjing University, Nanjing, P. R. China 210093
zhangyan@seg.nju.edu.cn, lxd@nju.edu.cn

Abstract. Components with undesired behavior could not be used properly by users. Therefore, the *scenario-based behavior filtration* of components is a significant problem to be solved, where the scenarios specify what behavior is undesired and what is desired. We propose an approach for filtering out the undesired behavior specified by a scenario specification from components. The main idea of our approach is that by constructing a special environment, i.e., *conditional exclusive environment*, for a component, all undesired behavior specified by one scenario specification can be filtered out and all desired behavior specified by another scenario specification can be preserved when the component works in the environment. We use interface automata to model the behavior of components and a set of action sequences to abstract the scenario specification in message sequence charts. The composition of components is modelled by the product of interface automata. We give the relevant algorithm in our approach and illustrate it by an example.

1 Introduction

Component-based software development (CBSD) is a good approach to attain reliable, flexible, extensible and evolvable systems. By the reuse of existing software components and the plug-and-play mechanisms, complex systems can be developed more rapidly and economically. In CBSD, users retrieve desired components from repositories and composite them to build a new system.

When an existing component could not meet the requirement of users exactly, we can compose several available components to perform the given task [1, 2]. Although components composition can repair inadequate behavior of sole component, it is insufficient to tackle the undesired behavior in available components. The behavior of a component that could obstruct the use of the component in some scenario may be undesired for specific users. Retrieved components with undesired behavior are frequently encountered by users, because users' requirements are various and it is difficult to find an exact match in repositories.

Usually, users give their requirements by a description of scenarios, which is called the *scenario specification*. The scenario specification can describe either the

* This paper is supported by the National Grand Fundamental Research 973 Program of China (No. 2002CB312001), the National Natural Science Foundation of China (No. 60425204, No. 60233020), and by Jiangsu Province Research Foundation (No. BK2004080).

user's desired or undesired behavior of a component when it interacts with other. The *scenario-based behavior filtration* of a component is to discard the undesired behavior and preserve desired behavior of the component in terms of the scenario specifications given by a user.

In this paper, we propose an approach to filtering the behavior for a component based on scenarios. By constructing an environment (i.e., another component) for a component, filter out all undesired behavior and preserve all desired behavior of the component when the component works in the environment. The undesired and desired behavior of the component are specified by scenario specifications. Interface automata [3] are used to model the behavior of components. Scenarios are specified by message sequence charts (MSCs) [4] and a MSC is abstracted as a set of action sequences further. The composition of components is modelled by the product of interface automata. We extend the concept of environment in the interface automata theory and introduce *conditional exclusive environment* (CXE). By constructing a CXE E for a given interface automaton R under two known sets $\mathcal{L}^+, \mathcal{L}^-$ of action sequences, make all behavior represented by some element in $\mathcal{L}^-$ to be discarded in $R \otimes E$. At the same time, all behavior represented by any element in $\mathcal{L}^+$, if it is also the behavior of R, is preserved in $R \otimes E$.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction on interface automata and message sequence charts. Section 3 introduces some relevant concepts about our proposal. Section 4 describes the approach to scenario-based behavior filtration of components in detail and shows the constructive algorithm of CXE. Finally, in section 5 we discuss the related works and conclude this paper. Additionally, an example is used to illustrate our approach throughout the paper.

2 Background

In the section, interface automata and MSCs are introduced briefly. The most of concepts about interface automata and MSCs refer to [3] and [4] respectively.

2.1 Interface Automata

Definition 1 (interface automaton, IA). *An* interface automaton $P = \langle V_P, V_P^{init}, \mathcal{A}_P^I, \mathcal{A}_P^O, \mathcal{A}_P^H, \mathcal{T}_P \rangle$ *is a 6-tuple, where*

- V_P *is a finite set of* states.
- $V_P^{init} \subseteq V_P$ *is a set of* initial states. *If* $V_P^{init} = \emptyset$ *then* P *is called* empty.
- $\mathcal{A}_P^I$, $\mathcal{A}_P^O$ *and* $\mathcal{A}_P^H$ *are mutually disjoint sets of* input, output *and* internal *actions.* $\mathcal{A}_P$ *denotes the set of all* actions*, i.e.,* $\mathcal{A}_P = \mathcal{A}_P^I \cup \mathcal{A}_P^O \cup \mathcal{A}_P^H$.
- $\mathcal{T}_P \subseteq V_P \times \mathcal{A}_P \times V_P$ *is a set of* steps. *If* $\tau = (v, a, u) \in \mathcal{T}_P$, *then write* $label(\tau) = a$, $head(\tau) = v$, $tail(\tau) = u$.

If $a \in \mathcal{A}_P^I$ (resp. $a \in \mathcal{A}_P^O$, $a \in \mathcal{A}_P^H$), then (v, a, v') is called an input (resp. output, internal) step. If there is a step $(v, a, v') \in \mathcal{T}_P$ for some $v, v' \in V_P$, $a \in \mathcal{A}_P$, then we say that action a is *enabled* at state v. For $v \in V_P$, let $\mathcal{A}_P^I(v) = \{a \in \mathcal{A}_P^I \mid \exists\, v' \in$

$V_P\,.\,(v,a,v') \in \mathcal{T}_P\}$, $\mathcal{A}_P^O(v) = \{a \in \mathcal{A}_P^O \mid \exists\, v' \in V_P\,.\,(v,a,v') \in \mathcal{T}_P\}$ and $\mathcal{A}_P^H(v) = \{a \in \mathcal{A}_P^H \mid \exists\, v' \in V_P\,.\,(v,a,v') \in \mathcal{T}_P\}$ be respectively the subset of input, output and internal actions that are enabled at the state v. Let $\mathcal{A}_P(v) = \mathcal{A}_P^I(v) \cup \mathcal{A}_P^O(v) \cup \mathcal{A}_P^H(v)$.

If IA P satisfies $\left|V_P^{init}\right| = 1$ and $\forall\,(v,a,u),(v,a,u') \in \mathcal{T}_P\,.\,u = u'$, then P is *deterministic*, otherwise P is *non-deterministic*. For simplicity, we make a convention that all interface automata referred in this paper are deterministic.

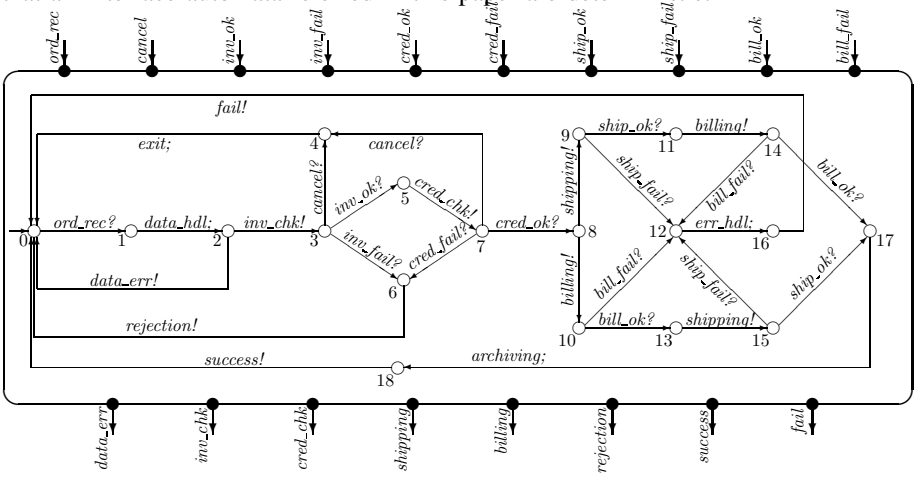


Fig. 1. Interface automaton $Seller$. The symbol "?" (resp. "!", ";") appended to the name of actions denotes that the action is an input (resp. output, internal) action. An arrow without source denotes the initial state of the interface automaton

Example 1. The IA $Seller$ (see Fig. 1) specifies the behavior of a component when it interacts with other. The component stands for a seller in a business to business system. The seller receives an order (ord_rec) from a customer and handles data in the order ($data\_hdl$), e.g., transform of data format. If there is some error in the order, it will report the error ($data\_err$) to the customer, otherwise it continues to check the inventory (inv_chk) from the supplier and the customer credit ($cred\_chk$) from the bank. Contingent on availability of inventory (inv_ok) and valid credit ($cred\_ok$), the seller will inform the shipper to ship product ($shipping$) and the bank to bill the customer for the order ($billing$). Either unavailability of inventory (inv_fail) or invalid credit ($cred\_fail$) will lead to reject the order ($rejection$). The seller can receive some information ($cancel$) from the customer to terminate ($exit$) the order. If shipping and billing finish successfully ($ship\_ok$ and $bill\_ok$), the seller will make archive ($archiving$) and give the notification ($success$) to the customer. Otherwise the negative notification ($fail$) will be given after processing the exception (err_hdl).

An *execution fragment* of IA P is a finite alternating sequence of states and actions $v_0 a_0 v_1 a_1 \cdots a_{n-1} v_n$, where $(v_i, a_i, v_{i+1}) \in \mathcal{T}_P$, for all $0 \le i < n$. Given two states $v, u \in V_P$, we say that u is *reachable from* v if there is an execution fragment with v as the first state and u as the last state. The state u is *reachable in* P if there is an initial state $v \in V_P^{init}$ such that u is reachable from v.

Let Γ_P denote the set of all execution fragments in IA P. For every $\eta \in \Gamma_P$, write the first state of η as $first(\eta)$, the last state of η as $last(\eta)$ and the set of all states of η as $V(\eta)$.

Definition 2 (interface automata product). *Two IAs P and Q are* composable *if $\mathcal{A}_P^H \cap \mathcal{A}_Q = \emptyset$, $\mathcal{A}_Q^H \cap \mathcal{A}_P = \emptyset$, $\mathcal{A}_P^I \cap \mathcal{A}_Q^I = \emptyset$ and $\mathcal{A}_P^O \cap \mathcal{A}_Q^O = \emptyset$. Let $shared(P,Q) = \mathcal{A}_P \cap \mathcal{A}_Q = (\mathcal{A}_P^I \cap \mathcal{A}_Q^O) \cup (\mathcal{A}_P^O \cap \mathcal{A}_Q^I)$ be the set of shared actions of P and Q. The* product *of P and Q, denoted by $P \otimes Q$, is the IA defined by*

$$
\begin{array}{rcl}
V_{P\otimes Q} & = & V_P \times V_Q \\
V_{P\otimes Q}^{init} & = & V_P^{init} \times V_Q^{init} \\
\mathcal{A}_{P\otimes Q}^I & = & (\mathcal{A}_P^I \cup \mathcal{A}_Q^I) \setminus shared(P,Q) \\
\mathcal{A}_{P\otimes Q}^O & = & (\mathcal{A}_P^O \cup \mathcal{A}_Q^O) \setminus shared(P,Q) \\
\mathcal{A}_{P\otimes Q}^H & = & \mathcal{A}_P^H \cup \mathcal{A}_Q^H \cup shared(P,Q) \\
\mathcal{T}_{P\otimes Q} & = & \{((v,u),a,(v',u)) \,|\, (v,a,v') \in \mathcal{T}_P \wedge a \notin shared(P,Q) \wedge u \in V_Q\} \\
 & \cup & \{((v,u),a,(v,u')) \,|\, (u,a,u') \in \mathcal{T}_Q \wedge a \notin shared(P,Q) \wedge v \in V_P\} \\
 & \cup & \{((v,u),a,(v',u')) \,|\, (v,a,v') \in \mathcal{T}_P \wedge (u,a,u') \in \mathcal{T}_Q \wedge a \in shared(P,Q)\}\,.
\end{array}
$$

At some state of $P \otimes Q$, one IA, say P (or Q), may produces an output action that is an input action of Q (or P), but isn't enabled at the current state in Q (or P). Such state is an *illegal states* of $P \otimes Q$. For two composable IAs P and Q, the set of illegal states of $P \otimes Q$ is denoted by $Illegal(P,Q) \subseteq V_P \times V_Q$,

$$
\begin{array}{rll}
Illegal(P,Q) = & \{(v, & u) \in V_P \times V_Q \,|\, \exists\, a \in shared(P,Q)\,. \\
 & & ((a \in \mathcal{A}_P^O(v) \wedge a \notin \mathcal{A}_Q^I(u)) \vee (a \in \mathcal{A}_Q^O(u) \wedge a \notin \mathcal{A}_P^I(v))\}\,.
\end{array}
$$

Definition 3 (environment). *An IA E is an* environment *for an IA R if: (1) E and R are composable, (2) E is not empty, (3) $\mathcal{A}_E^I = \mathcal{A}_R^O$, and (4) if $Illegal(R,E) \neq \emptyset$, then no state in $Illegal(R,E)$ is reachable in $R \otimes E$.*

2.2 Message Sequence Charts

MSC [4] is a trace description language for visualization of selected system runs. It concentrates on message interchange by communicating entities and their environment. Every MSC specification has an equivalent graphical and textual representation. Especially the graphical representation of MSCs gives an intuitive understanding of the described system behavior. Therefore, MSC is a widely used language for scenario specifications.

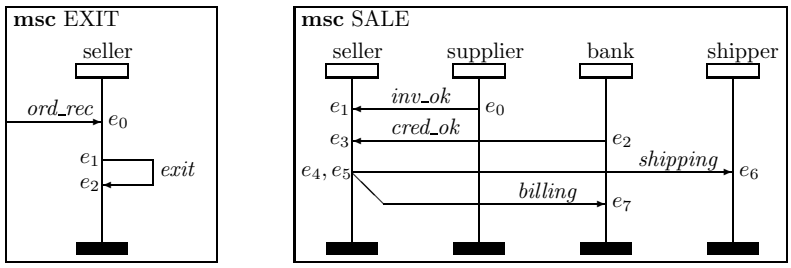
The fundamental language constructs of MSCs are component and message flow. Vertical time lines with a named heading represent components. Along these time lines, MSC events (i.e., message send or receive events) are arranged that gives an order to the events connected to this component. A message is depicted by an arrow from the send to the receive event. The fact that a message must be sent before it can be received imposes a total order on the send and receive event of a message and,

furthermore, a partial order on all events in a MSC. An example of MSCs is shown in Fig. 2 .

Definition 4 (message sequence chart, MSC). *A* message sequence chart $Ch = \langle \mathcal{C}, \mathcal{E}, \mathcal{M}, \mathcal{F}, \mathcal{O} \rangle$ *is a 5-tuple, where*

- $\mathcal{C}$ *is a finite set of* components.
- $\mathcal{E}$ *is a finite set of* events *corresponding to sending or receiving a message.*
- $\mathcal{M}$ *is a finite set of* messages. *For any* $m \in \mathcal{M}$, *let* $s(m)$ *and* $r(m)$ *to denote the events that correspond to sending and receiving message* m *respectively.*
- $\mathcal{F} : \mathcal{E} \to \mathcal{C}$ *is a* labelling function *which maps each event to a component.*
- $\mathcal{O} \subseteq \mathcal{E} \times \mathcal{E}$ *is a* partial order relation *over the set of events. For every* $(e, e') \in \mathcal{O}$, *there is* $e \neq e'$. (e, e') *represents a visual order displayed in* Ch.

Each MSC describes a set of message sequences. A *message sequence* of one MSC must be composed of all messages of the MSC and any message occurs only once in the sequence. For any two messages in the sequence, if one precedes the other then their send events and receive events should not violate the partial order relation over the set of events. Observe that messages in MSCs correspond to actions in IA. Hence, we call a message sequence of MSC as an *action sequence* derived from the MSC and write it as $\varrho = \varrho(0)\varrho(1)\cdots\varrho(n)$, where $\varrho(i)$ is a message in the message sequence for all $0 \leq i \leq n$.



(a) MSC 'EXIT' (b) MSC 'SALE'

Fig. 2. MSCs specifying scenarios about the interaction among the seller component, consumers and other components

Example 2. The MSCs 'EXIT' and 'SALE' (see Fig. 2(a) and 2(b) respectively) show two scenario specifications about the seller component (in Example 1) interacting with consumers and other components. The MSC 'EXIT' describes a scenario: the seller interrupts the process of ordering and exits after it receives an order from a customer. From the MSC 'EXIT' we can derive a set of action sequences, $\mathcal{L}_{\mathrm{E}} = \{ord\_rec\hat{}exit\}$. For legibility, we use the symbol "$\hat{}$" to separate two adjacent actions in an action sequence. The MSC 'SALE' describes a scenario: if the seller receives inv_ok and $cred\_ok$ it should produce $shipping$ to the shipper and $billing$ to the bank. From the MSC 'SALE' we can derive a set of action sequences, $\mathcal{L}_{\mathrm{S}} = \{inv\_ok\hat{}cred\_ok\hat{}shipping\hat{}billing, cred\_ok\hat{}inv\_ok\hat{}shipping\hat{}billing, inv\_ok\hat{}cred\_ok\hat{}billing\hat{}shipping, cred\_ok\hat{}inv\_ok\hat{}billing\hat{}shipping\}$.

3 Conditional Exclusive Environment

For any execution fragment $\eta = v_i a_i v_{i+1} a_{i+1} \cdots a_{j-1} v_j$ $(i < j)$ of IA P, where $v_i \in V_P^{init}$, if $v_i = v_j$ or $\mathcal{A}_P(v_j) = \emptyset$, then η is called a *run* in P. Let Σ_P denote the set of all runs in IA P. For any execution fragment $\eta = v_i a_i v_{i+1} a_{i+1} \cdots a_{j-1} v_j \in \Gamma_P$ $(i < j)$, we say that execution fragment $\eta' = v_s a_s v_{s+1} a_{s+1} \cdots a_{t-1} v_t$ $(i \leq s < t \leq j)$ *is in* η, denoted by $\eta' \sqsubseteq \eta$. Specifically, if $\eta' = v_s a_s v_{s+1}$ $(i \leq s < j)$, then we say that the step $\tau = (v_s, a_s, v_{s+1}) \in \mathcal{T}_P$ *is in* the execution fragment η, denoted by $\tau \sqsubseteq \eta$.

The *trace* of an execution fragment $\eta = v_0 a_0 v_1 a_1 \cdots a_{n-1} v_n$ is a subsequence of η, which consists of all actions in η. We write $trace(\eta) = a_0 a_1 \cdots a_{n-1}$. Given an execution fragment $\eta \in \Gamma_{P \otimes Q}$ and $trace(\eta) = a_0 a_1 \cdots a_{n-1}$, the *projection* of η on IA P, denoted by $\pi_P(trace(\eta))$, is a subsequence of $trace(\eta)$, which is obtained by deleting all actions $a_i \in \mathcal{A}_Q \setminus shared(P, Q)$, $0 \leq i \leq n-1$ from $trace(\eta)$.

Given two composable IAs P and Q, there are $\eta = v_0 a_0 v_1 a_1 \cdots a_{n-1} v_n \in \Gamma_P$ and $\alpha \in \Sigma_{P \otimes Q}$. If there exists an execution fragment $\zeta \sqsubseteq \alpha$ satisfying $\pi_P(trace(\zeta)) = trace(\eta)$ and for any $v_i a_i v_{i+1} \sqsubseteq \eta$ there is $(v_i, u_i) a_i (v_{i+1}, u_{i+1}) \sqsubseteq \zeta$, where $u_i, u_{i+1} \in V_Q$ and $0 \leq i < n$, then we say that η is *covered* by α. At the same time, (u_i, a_i, u_{i+1}) is called the *corresponding step* of (v_i, a_i, v_{i+1}) if $a_i \in shared(P, Q)$, and u_i, u_{i+1} is called the *corresponding state* of v_i, v_{i+1} respectively. If an execution fragment of IA P can be covered by a run of IA $P \otimes Q$, then it means that the behavior represented by the execution fragment of P can be preserved in $P \otimes Q$.

Given a run α of IA P and an action sequence ϱ, if ϱ is a subsequence of $trace(\alpha)$, then we say action sequence ϱ *occurs* in run α, denoted by $\varrho \propto \alpha$. The occurrence of an action sequence in a run of one IA means that some behavior of the IA contains the behavior represented by the action sequence.

Suppose that action sequence $\varrho = \varrho(0)\varrho(1) \cdots \varrho(m)$ occurs in run $\alpha \in \Sigma_P$. If there exists an execution fragment $\eta \sqsubseteq \alpha$ satisfying that ϱ is a subsequence of $trace(\eta) = a_0 a_1 \cdots a_n$ $(n \geq m)$ and $\varrho(0) = a_0$, $\varrho(m) = a_n$, then η is a *proper occurrence* of ϱ in α. Suppose that $\eta_0, \eta_1, \ldots, \eta_n \sqsubseteq \alpha$ are the proper occurrences of action sequences $\varrho_0, \varrho_1, \ldots, \varrho_n$ in α respectively. For any $\eta \sqsubseteq \alpha$, if $(V(\eta) \setminus \{first(\eta), last(\eta)\}) \cap V(\eta_i) = \emptyset$, $i = 0, 1, \ldots, n$, then η is a *proper inoccurrence* of $\varrho_0, \varrho_1, \ldots, \varrho_n$ in α.

Given a set $\mathcal{L}$ of action sequences, for any IA P, Σ_P can be partitioned as two subsets: $\phi_{\mathcal{L}}(\Sigma_P) = \{\alpha \in \Sigma_P \mid \exists\, \varrho \in \mathcal{L} \,.\, \varrho \propto \alpha\}$ and $\overline{\phi_{\mathcal{L}}(\Sigma_P)} = \Sigma_P \setminus \phi_{\mathcal{L}}(\Sigma_P)$. For every run in $\phi_{\mathcal{L}}(\Sigma_P)$, there exists at least one action sequence in $\mathcal{L}$ that occurs in it. For any run in $\overline{\phi_{\mathcal{L}}(\Sigma_P)}$, no action sequence in $\mathcal{L}$ occurs in it.

Definition 5 (conditional exclusive environment, CXE). *Given an IA R and a set $\mathcal{L}^-$ of action sequences, the* exclusive environment *of R under $\mathcal{L}^-$ is an environment E of R such that for any $\varrho \in \mathcal{L}^-$, if ϱ occurs in a run α of R, then the proper occurrence of ϱ in α must be not covered by any run of $R \otimes E$. If an exclusive environment E of R under $\mathcal{L}^-$ satisfies that for any $\varrho \in \mathcal{L}^+$, if ϱ occurs in a run α of R then α must be covered by some run of $R \otimes E$, where $\mathcal{L}^+$ is a set of action sequences and $\mathcal{L}^+ \cap \mathcal{L}^- = \emptyset$, then E is a* conditional exclusive environment *of R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$.*

Let $CXE\,(R:\mathcal{L}^-,\mathcal{L}^+)$ denote the set of conditional exclusive environments of R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$. If we consider $\mathcal{L}^+$ and $\mathcal{L}^-$ as the representation of two sets of behavior, then all behavior of R which contain any behavior in $\mathcal{L}^-$ isn't preserved in $R\otimes E$, at the same time, all behavior of R which contain any behavior in $\mathcal{L}^+$ is preserved in $R\otimes E$, where $E\in CXE\,(R:\mathcal{L}^-,\mathcal{L}^+)$. For arbitrary IA R and two sets $\mathcal{L}^-,\mathcal{L}^+$ of action sequences, it is possible that $CXE\,(R:\mathcal{L}^-,\mathcal{L}^+)=\emptyset$. It means that a CXE of R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$ may not always exist.

4 Construction of Conditional Exclusive Environment

We can use an IA, say R, to specify the behavior of a component, say COMP. An user can give his or her undesired and desired behavior about COMP by two scenario specifications in MSC, say 'SCENE$^-$' and 'SCENE$^+$' respectively. Filtering out the user's undesired behavior from COMP and preserving the desired behavior amounts to constructing a CXE for R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$, where $\mathcal{L}^-,\mathcal{L}^+$ are the sets of action sequences derived from MSCs 'SCENE$^-$', 'SCENE$^+$' respectively. If there exists $E\in CXE\,(R:\mathcal{L}^-,\mathcal{L}^+)$ and we can construct it, then all of the user's undesired behavior in R do not exist in $R\otimes E$, at the same time, all of the user's desired behavior in R are preserved in $R\otimes E$.

In this section, we will discuss how to construct a CXE $E\in CXE\,(R:\mathcal{L}^-,\mathcal{L}^+)$ for known IA R and two sets $\mathcal{L}^-,\mathcal{L}^+$ of action sequences in detail, and give the algorithm for constructing CXE.

4.1 Basic Approach to Constructing CXE

An environment of one IA, say R, can affect the runs of R only by the input actions of R. For arbitrary input step τ on arbitrary run of R, if the label of τ is a shared action of R and its environment and the environment does not provide the input action for R when R needs it, then R cannot go on along the run. For example, if the environment does not provide input action $cancel$ for IA $Seller$ (see Fig. 1) when $Seller$ stays at state 3, then $Seller$ cannot run along execution fragment "$3\;cancel\;4\;exit\;0$" back to initial state. That the environment does not provide input action $label(\tau)$ for R, when R needs it, amounts to no corresponding step of τ in the environment.

Suppose that η is a proper occurrence of some action sequence in $\mathcal{L}^-$. Only by not constructing the corresponding step in E for any input step τ of R, where $first(\eta)$ is reachable from $tail(\tau)$, the CXE E can make η not to be covered by any run of $R\otimes E$. For ensuring all runs in $\phi_{\mathcal{L}^+}(\Sigma_R)$ to be covered by runs of $R\otimes E$, the input step τ should not be in any run in $\phi_{\mathcal{L}^+}(\Sigma_R)$. We can find all such input steps in R by traversing all runs in $\phi_{\mathcal{L}^-}(\Sigma_R)$. But, if there exists a loop (i.e., execution fragment η with $first(\eta)=last(\eta)$) in some run, then Σ_R is an infinite set and the lengths of some runs in Σ_R, i.e., the number of steps in a run, may be also infinite. Accordingly, $\phi_{\mathcal{L}^-}(\Sigma_R)$, $\phi_{\mathcal{L}^+}(\Sigma_R)$ and the lengths of some runs in them may be infinite. Thus, it is unfeasible to traverse all runs in $\phi_{\mathcal{L}^-}(\Sigma_R)$ directly. For getting a feasible approach, we introduce the concepts of the simple run and simple loop.

Given an IA R and a set $\mathcal{L}$ of action sequences, a run $\alpha = v_0 a_0 v_1 a_1 \cdots a_{n-1} v_n$ of R is a *simple run* when it satisfies the following conditions:

1. if $\alpha \in \overline{\phi_{\mathcal{L}}(\Sigma_R)}$, then there is $v_i \neq v_j$ $(0 < i < n, 0 < j < n, i \neq j)$;
2. if $\alpha \in \phi_{\mathcal{L}}(\Sigma_R)$, then (a) for any proper inoccurrence $\eta = v_i a_i v_{i+1} \cdots a_{j-1} v_j$ $(0 \leq i < j \leq n)$ in α, there is $v_s \neq v_t$ $(i \leq s \leq j, i \leq t \leq j, s \neq t)$; and (b) for any proper occurrence ζ of $\varrho = \varrho(0)\varrho(1)\cdots\varrho(m) \in \mathcal{L}$ in α, if there is $\zeta' = v_i a_i v_{i+1} a_{i+1} \cdots a_{j-1} v_j \sqsubseteq \zeta$ $(0 \leq i < j \leq n)$, and $a_i = \varrho(k)$, $a_{j-1} = \varrho(k+1)$, $0 \leq k < m$, then there is $v_s \neq v_t$ $(i < s < j, i < t < j, s \neq t)$.

We put some constrains on runs to get the definition of the simple run. The meaning of the condition 1. is that there is not any loop in a simple run without occurrence of action sequences in $\mathcal{L}$. The meaning of the condition 2a is that there is not any loop in a proper inoccurrence of action sequences in a simple run. The meaning of the condition 2b is that in a proper occurrence of an action sequence in a simple run, there is not any loop between the occurrence of two neighbor actions in the action sequence. The set of all simple runs of IA R under $\mathcal{L}$ is denoted by $\Omega_R^{\mathcal{L}}$. Similarly, $\Omega_R^{\mathcal{L}}$ can be partitioned as $\phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right)$ and $\overline{\phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right)}$.

Given an IA R and a set $\mathcal{L}$ of action sequences, an execution fragment $\eta = v_i a_i v_{i+1} a_{i+1} \cdots a_{j-1} v_j \in \Gamma_R$ $(i < j)$ is a *simple loop* if: (1) $v_i = v_j$, $v_i, v_j \notin V_R^{init}$, (2) $v_s \neq v_t$ $(i \leq s < j, i \leq t < j, s \neq t)$ and (3) $\forall\, \alpha \in \phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right) . \eta \not\sqsubseteq \alpha$.

The first and second conditions ensure that except the first and the last states, there aren't duplicate states in a simple loop. The third condition ensures that a simple loop isn't the loop in a proper occurrence of some action sequence in $\mathcal{L}$. For given IA R and set $\mathcal{L}$ of action sequences, $\Theta_R^{\mathcal{L}}$ denotes the set of all simple loops of R. We say that simple loop $\eta \in \Theta_R^{\mathcal{L}}$ *associates* with simple run $\alpha \in \Omega_R^{\mathcal{L}}$ if $V(\eta) \cap V(\alpha) \neq \emptyset$ or $V(\eta) \cap V(\eta') \neq \emptyset$, where $\eta' \in \Theta_R^{\mathcal{L}}$ associates with α. Let $\psi_{\mathcal{L}}\left(\Theta_R^{\mathcal{L}}\right) = \left\{\eta \in \Theta_R^{\mathcal{L}} \,|\, \exists\, \alpha \in \phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right) . \eta \text{ associates with } \alpha\right\}$ be the set of all simple loops associated with simple runs in $\phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right)$.

Notice that every step in any run in Σ_R corresponds to a step in some simple run in $\Omega_R^{\mathcal{L}}$ or in some simple loop in $\Theta_R^{\mathcal{L}}$. However, $\Omega_R^{\mathcal{L}}$ and $\Theta_R^{\mathcal{L}}$ are finite sets and the lengths of all simple runs and simple loops are finite. Furthermore, $\phi_{\mathcal{L}}\left(\Omega_R^{\mathcal{L}}\right)$ and $\psi_{\mathcal{L}}\left(\Theta_R^{\mathcal{L}}\right)$ are finite sets.

Additionally, we also notice that it is impossible to eliminate the undesired behavior represented by $\varrho \in \mathcal{L}^-$ from $R \otimes E$ by not constructing the corresponding step in E for any step in R "after" the proper occurrence of ϱ. A step τ "after" a proper occurrence η means that $head(\tau) = last(\eta)$ or $head(\tau)$ is reachable from $last(\eta)$.

Suppose that ζ is a proper occurrence of $\varrho \in \mathcal{L}^-$ in a simple run α of IA R. We call a prefix η of α as the *minimal simple prefix* about ζ if ζ is a suffix of η, where a prefix of α is an execution fragment η in α and $first(\eta) = first(\alpha)$; a suffix of α is an execution fragment η in α and $last(\eta) = last(\alpha)$. Let $\lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$ denote the set of all minimal simple prefixes about all proper occurrences of any action sequence in $\mathcal{L}^-$ in any simple run of R, i.e.,

$$\lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right) = \Big\{\eta \,\Big|\, \exists\, \varrho \in \mathcal{L}^- . \exists\, \alpha \in \Omega_R^{\mathcal{L}^-} . (\zeta \text{ is a proper occurrence of } \varrho \text{ in } \alpha) \wedge (\eta \text{ is the minimal simple prefixabout } \zeta \text{ in } \alpha)\Big\}.$$

For any $\eta \in \lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$, there must be a proper occurrence of some $\varrho \in \mathcal{L}^-$ in η, and there is not any step "after" the proper occurrence in η.

Theorem 1. *For arbitrary IA R and sets $\mathcal{L}^-, \mathcal{L}^+$ of action sequences, if there exist $\eta \in \lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$ such that $\forall\, \tau \sqsubseteq \eta\,.\, label(\tau) \notin \mathcal{A}_R^I$, then $CXE\,(R{:}\mathcal{L}^-,\mathcal{L}^+) = \emptyset$.*

In [5], we prove that there maybe exist some kind of execution fragments in one IA, say P, which cannot be covered by any run of $P \otimes E$, for any environment E of P. Accordingly, we have the theorem as follows.

Theorem 2. *For arbitrary IA R and sets $\mathcal{L}^-, \mathcal{L}^+$ of action sequences, there does not exist any $E \in CXE\,(R : \mathcal{L}^-, \mathcal{L}^+)$ if there are $\eta_1, \eta_2 \in \Gamma_R$, $\eta_1 \sqsubseteq \alpha$ and $\eta_2 \sqsubseteq \beta$, for some $\alpha \in \left(\Omega_R^{\mathcal{L}^+} \cup \Theta_R^{\mathcal{L}^+}\right)$ and $\beta \in \left(\phi_{\mathcal{L}^+}\left(\Omega_R^{\mathcal{L}^+}\right) \cup \psi_{\mathcal{L}^+}\left(\Theta_R^{\mathcal{L}^+}\right)\right)$, which satisfy any of the following conditions: (1) $\eta_1 = v_i a v_j$ and $\eta_2 = v_j b v_k$, where $i \neq j \neq k$, $a \notin shared(R,E)$, $b \in \mathcal{A}_R^I \cap shared(R,E)$ and $b \notin \mathcal{A}_R(v_i)$. (2) $\eta_1 = v_i a v_j$ and $\eta_2 = v_i b v_k$, where $i \neq j \neq k$, $a \notin shared(R,E)$, $b \in \mathcal{A}_R^I \cap shared(R,E)$ and $b \notin \mathcal{A}_R(v_j)$. (3) $\eta_1 = v_i a_i v_{i+1} a_{i+1} \cdots a_{j-1} v_j$ and $\eta_2 = v_i b v_i'$, where $i < j$, $v_i' \notin V(\eta_1)$, $a_k \notin shared(R,E)$, $k = i, i+1, \ldots, j-1$, $b \in \mathcal{A}_R^I \cap shared(R,E)$ and $\exists\, v \in V(\eta_1)\,.\, b \notin \mathcal{A}_R(v)$.*

4.2 Algorithm of Constructing CXE

The skeleton of the constructive algorithm for CXE is described as follows. Step one, for every minimal simple prefix about the proper occurrence of any action sequence in $\mathcal{L}^-$ in some simple run of R, traverse it from the first state and find the first input step in it, which is not in any simple run with occurrence of action sequences in $\mathcal{L}^+$ or any simple loop associated with it. Step two, remove these input steps from R and all unreachable states after the removal. Step three, construct corresponding steps in one IA for all residual steps in R.

Make the convention of $\mathcal{A}_E^H = \emptyset$ and $\mathcal{A}_E^O = \mathcal{A}_R^I$ [5]. Let $R \upharpoonright \mathcal{T}$ to denote the IA obtained by removing all steps in $\mathcal{T} \subset \mathcal{T}_R$ from R and all unreachable states in R after the removal. The algorithm of constructing CXE $E \in CXE\,(R : \mathcal{L}^-, \mathcal{L}^+)$ is shown in Algorithm 1.

We can prove that the return (in line 24) of Algorithm 1 is a CXE of R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$ since it is consistent to Definition 5. Thus, Algorithm 1 is correct.

About line 1 in Algorithm 1, we had given an algorithm to find which simple run in an IA has the occurrence of a given action sequence in [6] and we can obtain those sets in line 1 based on the algorithm. About line 22 in Algorithm 1, we had given a method of constructing corresponding steps in [5].

Suppose that the maximal length of all elements in the set $\Omega_R^{\mathcal{L}^-} \cup \Omega_R^{\mathcal{L}^+} \cup \Theta_R^{\mathcal{L}^+}$ is $n = \max\left\{ length(\eta) \,|\, \eta \in \left(\Omega_R^{\mathcal{L}^-} \cup \Omega_R^{\mathcal{L}^+} \cup \Theta_R^{\mathcal{L}^+}\right)\right\}$, where $length(\eta)$ is the number of steps in η. Suppose that $m_1 = \left|\phi_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)\right|$, $m_2 = \left|\phi_{\mathcal{L}^+}\left(\Omega_R^{\mathcal{L}^+}\right)\right|$ are the number

Algorithm 1 Constructing CXE E of IA R under exclusion condition $\mathcal{L}^-$ and inclusion condition $\mathcal{L}^+$

Input: Interface automaton R and sets $\mathcal{L}^-, \mathcal{L}^+$ of action sequences, $\mathcal{L}^- \cap \mathcal{L}^+ = \emptyset$.
Output: CXE $E \in CXE\left(R : \mathcal{L}^-, \mathcal{L}^+\right)$.
Variables: $\mathcal{T} \subset \mathcal{T}_R$, step τ, IA R' and boolean found

1: Traverse R to get $\lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$, $\phi_{\mathcal{L}^+}\left(\Omega_R^{\mathcal{L}^+}\right)$ and $\psi_{\mathcal{L}^+}\left(\Theta_R^{\mathcal{L}^+}\right)$.
2: **if** some execution fragment satisfies the conditions of Theorem 2 **then**
3: **return** E doesn't exist // by Theorem 2
4: **else**
5: $\mathcal{T} \longleftarrow \emptyset$
6: **for all** $\eta \in \lambda_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$ **do**
7: found$\longleftarrow$ **true**
8: $\tau \longleftarrow$ the first step in η // $head(\tau) = first(\eta) \wedge \tau \sqsubseteq \eta$
9: **while** $\left(label(\tau) \notin \mathcal{A}_R^I \vee \exists\, \zeta \in \left(\phi_{\mathcal{L}^+}\left(\Omega_R^{\mathcal{L}^+}\right) \cup \psi_{\mathcal{L}^+}\left(\Theta_R^{\mathcal{L}^+}\right)\right).\, \tau \sqsubseteq \zeta\right) \wedge$ found **do**
10: **if** τ is not the last step in η **then** // $tail(\tau) \neq last(\eta) \wedge \tau \sqsubseteq \eta$
11: $\tau \longleftarrow$ the next step in η
12: **else** found$\longleftarrow$ **false**
13: **end if**
14: **end while**
15: **if** found **then** $\mathcal{T} \longleftarrow \mathcal{T} \cup \{\tau\}$
16: **else return** E doesn't exist // by Theorem 1
17: **end if**
18: **end for**
19: $R' \longleftarrow R \upharpoonright \mathcal{T}$
20: Initialize E: $V_E \longleftarrow \{u_0\}$, $V_E^{init} \longleftarrow \{u_0\}$
21: **for all** $\tau \in \mathcal{T}_{R'}$ **do**
22: Construct the corresponding step of τ in E
23: **end for**
24: **return** E
25: **end if**

of simple runs in $\phi_{\mathcal{L}^-}\left(\Omega_R^{\mathcal{L}^-}\right)$, $\phi_{\mathcal{L}^+}\left(\Omega_R^{\mathcal{L}^+}\right)$ respectively, and $k = \left|\psi_{\mathcal{L}^+}\left(\Theta_R^{\mathcal{L}^+}\right)\right|$ are the number of simple loops in $\psi_{\mathcal{L}^+}\left(\Theta_R^{\mathcal{L}^+}\right)$. In the worst case, line 6 to 18 in Algorithm 1 can be done in $O\left((m_2+k)m_1n^2\right)$ time. According to [6] and [5], line 1 and line 22 in Algorithm 1 need $O((m_1+m_2)n)$ and $O\left(|V_{R'}|\right)$ time respectively, where $|V_{R'}|$ is the number of states of IA R'. In general, there are $length(\eta) \ll length(\alpha)$ for $\eta \in \Theta_R^{\mathcal{L}^+}$ and $\alpha \in \Omega_R^{\mathcal{L}^+}$ and $|V_{R'}| \ll (m_1+m_2)n$. Hence, the complexity of Algorithm 1 is $O\left(m_1m_2n^2\right)$.

Example 3. Suppose that MSCs 'EXIT' (Fig. 2(a)) and 'SALE' (Fig. 2(b)) describe a user's undesired and desired behavior about IA $Seller$ (Fig. 1) respectively. That is, the user does not want the process of ordering to be terminated by cancellation. By Algorithm 1, we can obtain a CXE E (Fig. 4) of the IA $Seller$ under exclusion condition $\mathcal{L}_{\mathrm{E}}$ and inclusion condition $\mathcal{L}_{\mathrm{S}}$, which are two sets of action sequences derived from MSCs 'EXIT' and 'SALE' respectively (see Example 2). The intermediate result R' (see line 19 of Algorithm 1) is shown in Fig. 3. It can be found that the user's undesired behavior of $Seller$ is discarded in the composition of $Seller$ and E, i.e., $Seller \otimes E$ (Fig. 5). At the same time, the user's desired behavior of $Seller$ is preserved in $Seller \otimes E$.

5 Related Works and Conclusion

In this paper, we give an approach for filtering the undesired behavior and preserving the desired behavior of components based on scenario specifications.

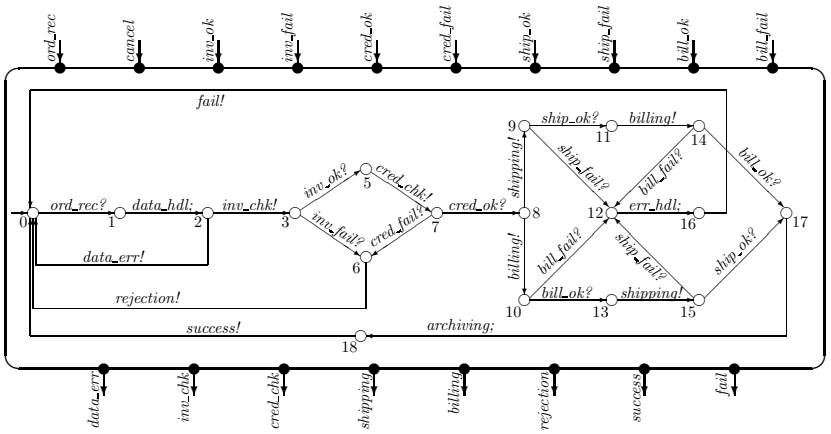


Fig. 3. IA R' . The intermediate result of Algorithm 1 with inputs $Seller$, $\mathcal{L}_{\mathrm{E}}$ and $\mathcal{L}_{\mathrm{S}}$

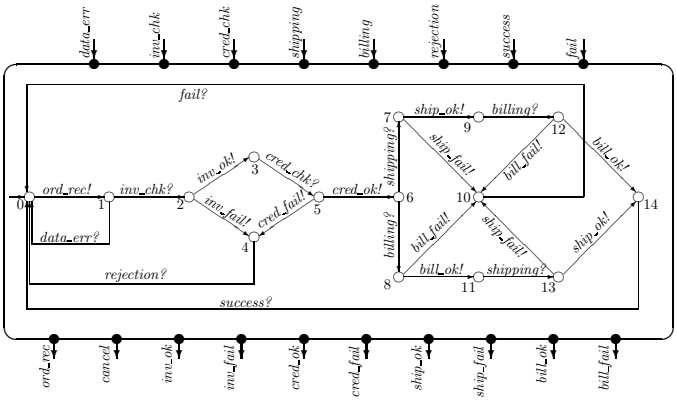


Fig. 4. The CXE E of $Seller$ under exclusion condition $\mathcal{L}_{\mathrm{E}}$ and inclusion condition $\mathcal{L}_{\mathrm{S}}$

In [1, 2, 7], the authors mainly solve the behavioral compatibility of components composition, but do not concern whether all behavior of the composition are the needs of users. By using environment, our approach can filter out undesired behavior from

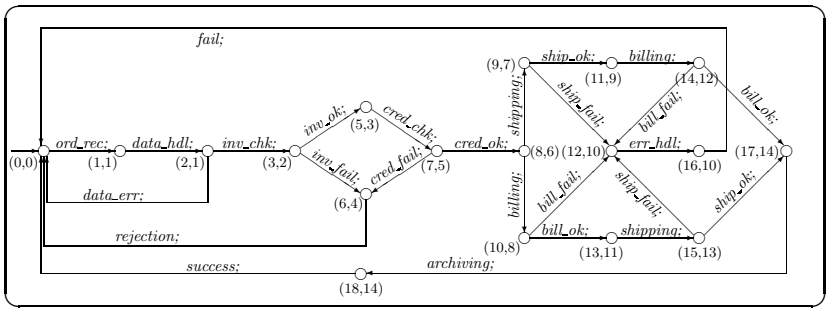


Fig. 5. $Seller \otimes E$. The composition of $Seller$ and E

components or compositions in terms of the user's requirements. The most pertinent research is to automatically synthesize a connector for restricting the behavior of the composed components to the desired behavior specified by temporal logic based specifications [8, 9]. Contrary to [8, 9], the environment in our approach adjusts the behavior of components only by the inputs, and our algorithm is better in complexity.

References

1. Bracciali, A., Brogi, A., Canal, C.: A formal approach to component adaptation. Journal of Systems and Software **74**(1) (2004) 45–54
2. Yellin, D.M., Strom, R.E.: Protocol specifications and component adaptors. ACM Transactions on Programming Languages and Systems **19**(2) (1997) 292–333
3. de Alfaro, L., Henzinger, T.A.: Interface automata. In: Proceedings of the 9th Annual ACM Symposium on Foundations of Software Engineering (FSE 2001), ACM Press (2001) 109–120
4. ITU-TS: ITU-TS recommendation Z.120: Message Sequence Chart (MSC). ITU-TS, Geneva (1999)
5. Zhang, Y., Hu, J., Yu, X., Zhang, T., Li, X., Zheng, G.: Deriving available behavior all out from incompatible component compositions. In: Proceedings of the 2nd International Workshop on Formal Aspects of Component Software (FACS'05), Electronic Notes in Theoretical Computer Science (2006) (To appear).
6. Hu, J., Yu, X., Zhang, Y., Zhang, T., Wang, L., Li, X., Zheng, G.: Scenario-based verification for component-based embedded software designs. In: Proceedings of the 34th International Conference on Parallel Processing Workshops (ICPP 2005 Workshop), IEEE Computer Society (2005) 240–247
7. Schmidt, H.W., Reussner, R.: Generating adapters for concurrent component protocol synchronisation. In: IFIP TC6/WG6.1 Fifth International Conference on Formal Methods for Open Object-Based Distributed Systems, Kluwer (2002) 213–229
8. Inverardi, P., Tivoli, M.: Software architecture for correct components assembly. In Bernardo, M., Inverardi, P., eds.: Formal Methods for Software Architectures. Volume 2804 of Lecture Notes in Computer Science. Springer-Verlag (2003) 92–121
9. Tivoli, M., Autili, M.: SYNTHESIS: a tool for synthesizing "correct" and protocol-enhanced adaptors. L'Object Journal **12**(1) (2005)