Assume-guarantee reasoning with ioco testing relation

Laura Brandán Briones

Fa.M.A.F. Universidad Nacional de Córdoba, Argentina

Abstract. Compositional reasoning is typically based on assume-guarantee reasoning principles, which consider each component separately and take into account assumptions about the context of the component. This paper presents a combination of the assume-guarantee paradigm and **ioco**, a formal conformance relation for model-based testing that works on input-output transition systems (IOTS). We show that, under certain restrictions, assume-guarantee reasoning can be applied in the **ioco** context, enabling to check **ioco**-conformance by testing components' system separately. We improve on previous results, where specifications are required to be given as components, allowing the specifications to be complete systems. Moreover, we prove that assume-guarantee reasoning can also be applied even when hiding internal communication between components.

1 Introduction

Conformance testing tries to identify if a system under test (known as SUT) behaves as expected. We consider a black-box conformance testing framework to test components of a system, where both the components and their specifications are modeled as input-output transition systems (IOTS), which formalize system descriptions that interact with their environment by receiving inputs and offering outputs. An IOTS that has the input set I and the output set U is denoted as IOTS(I, U). The **ioco** input output conformance relation asserts when an implementation behaves as expected by a given specification, both modeled by two IOTS(I, U); **ioco** checks the inclusion of specification' Straces (that is, traces with information about *quiescence*); in the presence of non-determinism, **ioco** can distinguish systems that are indistinguishable under trace inclusion.

A previous approach [6] studies compositional properties of **ioco**. Given two pairs of *input-enabled* (defined below) systems i_1, s_1 as IOTS(I_1, U_1) and i_2, s_2 as IOTS(I_2, U_2) the following compositional rule is proved: if i_1 **ioco** s_1 and i_2 **ioco** s_2 hold then $i_1||i_2$ **ioco** $s_1||s_2$ holds. In this previous framework the global specification is given as a composition of two systems ($s_1||s_2$) and required to be input-enabled. However, we believe that it is often natural and easier to initially write a global system specification as a single system (rather than to decompose it into subsystems s_1 and s_2). Also, it is often the case that components conform to their specifications only in specific *contexts* (or environments). We thus solve the following problem: is it possible to use formal testing in a compositional way, using a global specification and taking into account *assumptions* about the environments in which the components are supposed to operate?

We combine **ioco** with assume-guarantee reasoning, a "divide-and-conquer" approach that infers global system properties by checking individual components in isolation, under environment assumptions [2–4]. Assuming A, we prove that:

$$\frac{i_1||A \textbf{ ioco } S \quad \wedge \quad i_2 \textbf{ ioco } A}{i_1||i_2 \textbf{ ioco } S}$$

As a consequence, if A is provided we can test in isolation components (which may be at different stages of development and possibly written by different developer teams), having a specification of the overall system. Moreover, we prove that if the communication between the systems is hidden, a similar result also holds:($\textbf{hide} V \textbf{ in } i_1||A$) **ioco** S and i_2 **ioco** A then ($\textbf{hide } V \textbf{ in } i_1||i_2$) **ioco**$S$.

2 The ioco testing relation & composition in IOTS

This section recalls same aspects of the **ioco** theory, for more details see [5].

An input-output transition system ($IOTS$) is a tuple $\langle Q, q^0, L, T\rangle$, where: Q is a countable, non-empty set of states, with $q^0 \in Q$ the initial state. L is a countable set of labels, partitioned into input (I) and output (U) actions, with $I \cap U = \emptyset$ and $I \cup U = L$. $T \subseteq (Q \times (L \cup \{\tau\}) \times Q)$ is the transition relation.

We denote the class of all labeled transition systems over I and U by $IOTS(I, U)$. We use a special label $\tau \notin L$ to denote internal actions. For an arbitrary $L' \subseteq L$, we use L'_τ as a shorthand for $L' \cup \{\tau\}$. For a system p, we write Q_p, L_p, and so on to denote the components of p. We write $q \xrightarrow{\mu} q'$ when $(q, \mu, q') \in T$. We use "?" and "!" before a label to denote whether it is an input or output action. A state that cannot perform an internal action is called *stable*, whereas one that cannot do an output or internal action is called *quiescent*. We use the symbol δ (with $\delta \notin L_\tau$) to represent quiescence. For an arbitrary $L' \subseteq L_\tau$, we use L'_δ as shorthands for $L' \cup \{\delta\}$. An $IOTS$ is called *strongly convergent* if it does not have infinite τ-labeled sequence. The set of all traces over L' (with $L' \subseteq L$) is denoted by L'^*; a trace in L'^* is denoted by σ, with ϵ denoting the empty sequence. If $\sigma_1, \sigma_2 \in L'^*$, then $\sigma_1 \cdot \sigma_2$ is the concatenation of σ_1 and σ_2. We use the standard notation with single and double arrows for traces: $q \xrightarrow{\mu_1 \cdots \mu_n} q'$ denotes $q \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} q'$, $q \overset{\epsilon}{\Rightarrow} q'$ denotes $q \xrightarrow{\tau \cdots \tau} q'$ and $q \overset{\mu_1 \cdots \mu_n}{\Longrightarrow} q'$ denotes $q \overset{\epsilon}{\Rightarrow}\xrightarrow{\mu_1}\overset{\epsilon}{\Rightarrow} \cdots \overset{\epsilon}{\Rightarrow}\xrightarrow{\mu_n}\overset{\epsilon}{\Rightarrow} q'$, with $\mu_i \in L'_{\tau\delta}$. We write $q \overset{\mu}{\Rightarrow}$ if there exists a q' such that $q \overset{\mu}{\Rightarrow} q'$. An IOTS$(I, U)$ system $p = \langle Q, q^0, L, T\rangle$ is called *input-enabled* (denoted IOTS–**ie**) if all inputs are enabled in all states, i.e. $\forall\, q \in Q : \forall \mu \in I : q \overset{\mu}{\Rightarrow}$. Finally, for $I' \subseteq I_p$ and $\forall\, q \in Q : \forall \mu \in I' : q \overset{\mu}{\Rightarrow}$ we say that p is input-enabled with respect to I', denoted p-$\textbf{ie}(I')$. Moreover, as in typical process algebra semantics, we sometimes use a transition system and its initial state interchangeably.

Let p be an IOTS(I, U), $Q' \subseteq Q_p$ a subset of states in p, $q \in Q_p$ and $\sigma \in L^*_\delta$, then: q **after** $\sigma \triangleq \{q' | q \overset{\sigma}{\Rightarrow} q'\}$, $\textbf{out}(q) \triangleq \{\mu \in U | q \xrightarrow{\mu}\} \cup \{\delta | q \xrightarrow{\delta}\}$, $\textbf{out}(Q') \triangleq \bigcup\{\textbf{out}(q) | q \in Q'\}$ and Straces$(p) \triangleq \{\sigma \in L^*_\delta | q^0_p \overset{\sigma}{\Rightarrow}\}$.

Definition 1. *Given implementation $i \in IOTS(I, U)$-**ie** and specification $S \in IOTS(I, U)$: i **ioco** S iff $\forall\, \sigma \in Straces(S) : \textbf{out}(i \textbf{ after } \sigma) \subseteq \textbf{out}(S \textbf{ after } \sigma)$.*

Restricted ioco Since the **ioco**-relation gives implementation's freedom on inputs that do not appear in specifications, it is natural to ask if **ioco** holds when the specification inputs are included in the implementation inputs.

Definition 2. *Let $p = \langle Q_p, q_p^0, L_p, T_p\rangle$ be an IOTS(I_p, U_p) with $L_p = I_p \cup U_p$, $I \subseteq I_p$ and $U \subseteq U_p$ we define the restriction of p in (I, U), denoted by $\mathbf{R}\text{-}p(I, U)$, as the IOTS defined by $Q = Q_p$, $q^0 = q_p^0$, $L = I \cup U$ and $T = T_p \cap (Q_p \times L_\tau \times Q_p)$.*

In the case that the implementation actions are a subset of the specification actions, the **ioco** relation restricted to the specification actions follows easily (In the statement of the lemma below, we slightly abuse notation w.l.o.g. and assume that **ioco** allows inputs of the specification to be included, and not exactly equal, to the inputs of the implementation).

Lemma 1. *Let i_1 be an IOTS(I_{i_1}, U_{i_1}) and s be a IOTS(I_s, U_s) with $I_s \subseteq I_{i_1}$ and $U_{i_1} \subseteq U_s$. Let i_2 be the restriction of i_1 in (I_s, U_s), $i_2 = \mathbf{R}\text{-}i_1(I_s, U_s)$, with i_2-**ie**(I_s) then: if i_2 **ioco** s then i_1 **ioco** s*

One of the advantages of the **ioco** relation is that it gives freedom to implementation behaviors that are not specified by the specification[5]. As a result, in Lemma 1, we use this liberty to test implementations with a wider input actions set than the ones specified by the specification. Also, we relax the *input-enabled* assumption to the subset of specification's input actions.

2.1 Composition in IOTS

The integration of components can be modeled algebraically by putting the components in parallel while synchronizing their common actions. The synchronization of processes p_1 and p_2 is denoted $p_1||p_2$.

Definition 3. *Let $p_1 = \langle Q_1, q_1^0, L_1, T_1\rangle$ and $p_2 = \langle Q_2, q_2^0, L_2, T_2\rangle$ be IOTS' with $I_1 \cap I_2 = U_1 \cap U_2 = \emptyset$ then $p_1||p_2 = \langle Q, q_1^0||q_2^0, L, T\rangle$, is defined as: $Q = \{q_1||q_2 \mid q_1 \in Q_1, q_2 \in Q_2\}$, $I = (I_1 \setminus U_2) \cup (I_2 \setminus U_1)$, $U = U_1 \cup U_2$ and T is the minimal set satisfying the following inference rules ($\mu \in L_\tau$):*
$q_1 \overset{\mu}{\to} q_1', \mu \notin L_2 \vdash q_1||q_2 \overset{\mu}{\to} q_1'||q_2$; $q_1 \overset{?\mu}{\to} q_1', q_2 \overset{!\mu}{\to} q_2', \mu \notin \tau \vdash q_1||q_2 \overset{!\mu}{\to} q_1'||q_2'$
$q_1 \overset{!\mu}{\to} q_1', q_2 \overset{?\mu}{\to} q_2', \mu \notin \tau \vdash q_1||q_2 \overset{!\mu}{\to} q_1'|| q_2'$; $q_2 \overset{\mu}{\to} q_2', \mu \notin L_1 \vdash q_1||q_2 \overset{\mu}{\to} q_1||q_2'$

Here, inputs $?a$ in one system are matched with outputs $!a$ in the other system, the result being an output $!a$ in their parallel composition. Given two systems p_1 and p_2, we use $Share(p_1, p_2)$ to denote $(I_1 \cap U_2) \cup (I_2 \cap U_1)$.

Note that Definition 3 puts constraints on input and output sets. So, parallel composition may give rise to IOTS' that are not *strongly convergent*, even if their components are. Thus, we implicitly restrict to cases where parallel composition is strongly convergent. Moreover, we only use binary parallel composition.

In Figure 1, on the left, we present three IOTS representing a cash machine. We represent the initial state with double circle. First, on the left hand side we present a device that takes care of the card received by our cash machine. The CARD device receives a card, announces that it has the card, and expects for an

OK answer or an error answer. If an OK answer arrives it gives back the card. If an error answer arrives the CARD device keeps the card, assuming something went wrong with the pin so the device retains the card.

Second, in the middle, we present a PIN device. It starts receiving the announcement that there is a card in the machine, then it expects a pin number. If the pin is correct, the device gives money and sends an OK acknowledgement. If the pin is incorrect, the device sends an error acknowledgement and an specific error stating that the pin was not correct.

Third, the CARD-PIN machine is depicted, that is the parallel composition of the CARD device and PIN device after the synchronization between them. In the CARD-PIN machine we see that the system receives a card and a pin number, then two things can happen. Either the system gives money and the card back (this is the case when the pin number was correct), or the system retains the card and gives an error indication that the pin was not correct.

3 Assume-guarantee reasoning with ioco

In this section we consider the following assume-guarantee rule: if $i_1||A$ **ioco** $S \ \wedge \ i_2$ **ioco** A holds then $i_1||i_2$ **ioco** S holds. The rule says that if, under assumption A, i_1 conforms with S and i_2 conforms A, then we can be sure that the parallel composition $i_1||i_2$ conforms w.r.t. S. We show (in Theorem 1 below) that under certain restrictions, this rule is sound. Therefore, we can obtain $i_1||i_2$ **ioco** S by checking $i_1||A$ **ioco** S and i_2 **ioco** A separately.

There are two complications in applying **ioco** in assume-guarantee reasoning. Firstly, *quiescent* states are not preserved by composition. Secondly, **ioco** allows implementation freedom for inputs that are not specified. To apply **ioco** we need to assume that implementations are input-enabled with respect to specification inputs. Therefore, to assert that $i_1||i_2$ **ioco** S, we also assume that $i_1||i_2$ is input-enabled with respect to S's inputs. We prove that, for an input-enabled system A such that $i_1||A$ **ioco** S and i_2 **ioco** A, then $i_1||i_2$ is **ioco** S. Note that we improve on previous result [6], and do not require S to be input-enabled.

Definition 4. *Let $\mu \in L_\delta$, $\sigma \in L_\delta^*$ and $L' \subseteq L_\delta$, then $\epsilon\lceil L' = \epsilon$ and $(\mu \cdot \sigma)\lceil L' = \sigma\lceil L'$ if $\mu \notin L'$ or $\mu \cdot (\sigma\lceil L')$ if $\mu \in L'$.*

In the proof of Theorem 1 we use the following result from [6]: Let p_1 be a $IOTS(I_1, U_1)$ and p_2 be a $IOTS(I_2, U_2)$ with $I_{p_1} \cap I_{p_2} = U_{p_1} \cap U_{p_2} = \emptyset$ and $L = I \cup U$ where $I = I_1 \cup I_2 / Share(p_1, p_2) \wedge U = U_1 \cup U_2$, $r \in Q_{p_1||p_2}, \sigma$ be in L_δ^*, then: $p_1||p_2 \overset{\sigma}{\Rightarrow} r$ iff $\exists\, p_1', p_2' : p_1 \overset{\sigma\lceil L_{p_1}^\delta}{\Longrightarrow} p_1' \wedge p_2 \overset{\sigma\lceil L_{p_2}^\delta}{\Longrightarrow} p_2' \wedge r = p_1'||p_2'$. Basically, if p is the parallel composition of p_1 and p_2 ($p = p_1||p_2$) under some restriction, for all reachable states r in p it is possible to find a state r_1 in p_1 and a state r_2 in p_2 such that $r = r_1||r_2$. The inverse also holds (see [6]).

Theorem 1. *Let i_1 be an $IOTS(I_1, U_1)$–**ie**(I_1), A and i_2 be $IOTS(I_2, U_2)$–**ie**(I_2) and S be an $IOTS(I_3, U_3)$. With $I_1 \cap I_2 = U_1 \cap U_2 = \emptyset$ and $U_1 \cup U_2 \subseteq U_3 \wedge I_3 \subseteq I_1 \cup I_2$ then: if $i_1||A$ **ioco** $S \ \wedge \ i_2$ **ioco** A hold then $i_1||i_2$ **ioco** S holds.*

Theorem 1 allows us to apply **ioco** not only to composed systems but also to specifications given as a complete system knowing an assumption about a subpart of the system. Compared to the previous result, the new assume-guarantee approach assumes that we can have an assumption on one of the components.

4 Hiding in assume-guarantee reasoning with ioco

Definition 5. *Let $p = \langle Q, q^0, L, T\rangle$ be an IOTS with $L = I \cup U$ and $V \subseteq U$ then we define* **hide** V **in** p *as a new IOTS with $\langle Q', q'^0, L', T'\rangle$ where: $Q' = Q$, $q'^0 = q^0$, labs$' = I \cup (U \backslash V)$ and T' is the minimal set satisfying the following inference rules, with $\mu \in L_\tau$, $q_1, q_2 \in Q$ and $q'_1, q'_2 \in Q'$: $q_1 \xrightarrow{\mu} q_2$, $\mu \notin V \vdash q'_1 \xrightarrow{\mu} q'_2$ and $q_1 \xrightarrow{\mu} q_2$, $\mu \in V \vdash q'_1 \xrightarrow{\tau} q'_2$.*

Again, note that Definition 5 gives constraints on the input and output sets. Therefore, hiding may gives rise to IOTS' that are not *strongly convergent*, even if their components are. So, we implicitly restrict ourselves to cases where hiding is *strongly convergent.*

On the right hand side in Figure 1 we show the H(CARD-PIN) machine, i.e. the parallel composition of the CARD and PIN machine after hiding the synchronization actions between them.

An immediate question, given from the results from Section 3, is the following: Is it possible to hide the internal communication between components? The answer is affirmative, as shown in the next theorem.

Theorem 2. *Let $i_1 \in$ IOTS-**ie**(I_1, U_1), $A, i_2 \in$ IOTS-**ie**(I_2, U_2) and furthermore $S \in$ IOTS(I_3, U_3), with $I_1 \cap I_2 = U_1 \cap U_2 = \emptyset$. Let $V \subseteq$ Share(i_1, i_2) and $(U_1 \cup U_2)\backslash V \subseteq U_3 \wedge V \cap U_3 = \emptyset \wedge I_3 \subseteq I_1 \cup I_2$ then: if* (**hide** V **in** $i_1||A$) **ioco** S *and* i_2 **ioco** A *hold then* (**hide** V **in** $i_1||i_2$) **ioco** S *holds.*

This result enables us to test the system allowing to incorporate new interfaces between components by only changing the assumption and leaving the specification of the whole systems the way it was.

Recall the specification S of the cash machine from Figure 1(CARD-PIN). There, we can insert a card, later a pin number, and obtain two answers from the machine: positive or negative. In the case we obtain a positive answer, we receive the money and the card. In the case we obtain a negative answer, we only receive a message informing that the pin number is wrong.

Now suppose two companies are developing the internal components of this cash machine; the two components interact as shown on the left of Figure 2. The cash machine has two internal components (CARD and PIN), which interact using the labels *have-card*, *OK* and *err*, but this is not specified in CARD-PIN. So, to allow both companies to test each part separately, the system that interacts with the CARD component is specified in A, shown in Figure 2.

Given one CARD implementation i_{CARD} and one PIN i_{PIN}, let V be the internal communication s.t. $V = \{\text{have-card}, \text{OK}, \text{err}\}$. Using Theorem 2, it is enough to prove that (**hide** V **in** $i_{\text{CARD}}||A$) **ioco** S and i_{PIN} **ioco** A, to establish that (**hide** V **in** $i_{\text{CARD}}||i_{\text{PIN}}$) **ioco** S.

Conclusions To the best of our knowledge, ours is the first application of assume-guarantee with **ioco**. We plan to extend this work considering probabilistic and realtime settings (e.g., the **tioco** timed testing relation [1]).

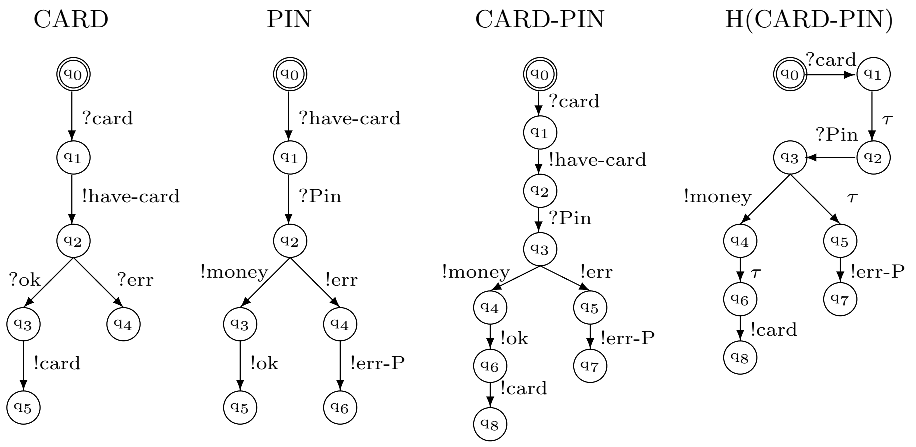

Fig. 1. Parallel composition of CARD and PIN and their hiding communication

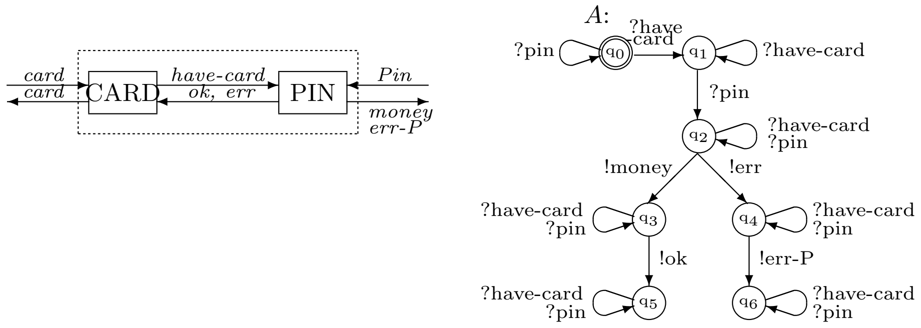

Fig. 2. Left: Internal components of the cash machine from Figure 1(left). Right: The assumption of the environment in which the CARD component is assumed to function

References

1. L. Brandán Briones and E. Brinksma. A test generation framework for quiescent real-time systems. In Brian Nielsen Jens Grabowski, editor, *Formal Approaches to Software Testing, FATES*, Linz, Austria, Sep 2004. Springer-Verlag GmbH.
2. E. M. Clarke, D. E. Long, and K. L. McMillan. Compositional model checking. In *Symposium on Logic in Computer Science*, June 1989.
3. W. Grieskamp, Y. Gurevich, W. Schulte, and M. Veanes. Generating finite state machines from abstract state machines. In *ISSTA*, July 2002.
4. A. Pnueli. In transition from global to modular temporal reasoning about programs. In *K. Apt, editor, Logic and Models of Concurrent Systems, volume 13*, 1984.
5. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. In *Software-Concepts and Tools, 17(3)*, pages 103–120, 1996.
6. Machiel van der Bijl, Arend Rensink, and Jan Tretmans. Component based testing with ioco. Technical report, University of Twente, 2003.