Input/Output Compatibility of Reactive Systems*

Josep Carmona[1] and Jordi Cortadella[2]

[1] Universitat Politècnica de Catalunya
Computer Architecture Department
Avda. Canal Olimpic, s/n. 08860 Castelldefels, Spain
jcarmona@ac.upc.es

[2] Universitat Politècnica de Catalunya
Software Department
Jordi Girona 1-3 08034 Barcelona, Spain
jordic@lsi.upc.es

Abstract. The notion of I/O compatibility of reactive systems is defined. It models the fact that two systems can be connected and establish a correct dialogue through their input and output events. I/O compatibility covers safeness and liveness properties that can be checked with a polynomial-time decision procedure. The relationship between observational equivalence, I/O compatibility and input properness is also studied with the aim at supporting the proposal of transformations for the synthesis of reactive systems. Finally, a set of Petri net transformations that preserve I/O compatibility are shown as an example of application of the theory presented in this paper.

Keywords

Reactive systems, Input/Output compatibility, Observational equivalence, Synchronous product, Trace theory, Conformation, Petri nets.

1 Introduction

This section is devoted to present the motivation of this work and a summary of the main contributions.

1.1 Reactive Systems

A system is said to be reactive when it has an explicit interaction with an environment. A reactive system can receive input stimuli from the environment,

* This work has been partially funded by the Ministry of Science and Technology of Spain under contract TIC 2001-2476, ACiD-WG (IST-1999-29119) and a grant by Intel Corporation.

Carmona, J.; Cortadella, J. Input/output compatibility of reactive systems. A: International Conference on Formal Methods in Computer-Aided Design. "Formal Methods in Computer-Aided Design, 4th International Conference, FMCAD 2002: Portland, OR, USA, November 6-8, 2002: proceedings". Springer, 2002, p. 360-377.
The final authenticated version is available online at https://doi.org/10.1007/3-540-36126-X_22

execute internal operations and produce results observable by the environment. Formally, a reactive system can be modeled as a transition system with an explicit distinction among input, internal and output events. The system can only control its own events (internal and output), but cannot prevent the environment from producing input events if it decides to do so.

Two different reactive systems can interact by connecting their inputs and outputs. We assume that the composition of reactive systems is done by synchronizing common events. An example of composition is the connection of two digital circuits, in which the transitions of any output signal are simultaneously observed by the circuit receiving them as inputs. Thus, the concept of environment is relative: each system considers the other to be its environment.

1.2 Motivation

The motivation comes from the need to formalize the fact that two systems can be connected and establish a consistent dialogue through their input and output events. The theory presented in this paper is inspired on the work by Dill [7]. The formal model for specifying a system considered here is more restricted that the one presented by Dill for complete trace structures. However, the properties covered by the model, including some notion of liveness, can be checked in polynomial time. For the type of systems that we want to deal with, the model is powerful enough. The definition of correct interaction is done by relating the states of the two systems. This state-based definition eases the proof of properties on their interaction.

When then enabledness of input events is considered, sufficient conditions can be obtained that relate the theory with the well-known concept of observational equivalence [14] and input properness [3]. Finally, we show that the theory presented can be used for the synthesis of reactive systems. A kit of Petri net transformations is presented that are proved to preserve the notion of I/O compatibility.

A practical application of this work is found in the area of synthesis of concurrent systems, e.g. asynchronous circuits [3] or codesign of embedded systems [9].

1.3 I/O Compatibility

The notion we want to model is *Input/Output compatibility*. We now illustrate this notion with some examples and show why other equivalences for concurrent systems are not appropriate.

Figure 1(a) depicts two reactive systems, X and Y, synchronized by a pair of events, a and b. Event a is an output for X and an input for Y, whereas b is an input for X and an output for Y. Moreover, X has an internal event τ. When enabled, internal and output events may take an unbounded, but finite, delay to fire. At each state, a system has only a (possibly empty) subset of input events enabled. If a non-enabled input is produced by the other partner, a communication failure is produced.

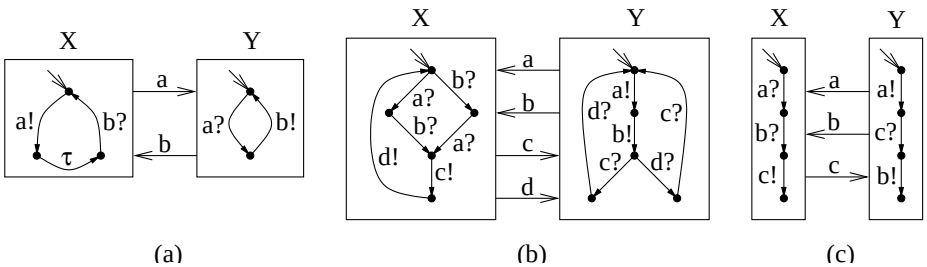

Fig. 1. Connection between different reactive systems (the suffixes ? and ! are used to denote input and output events, respectively).

The transition systems in Fig. 1(a) are observational equivalent. However, they are not I/O compatible, according to the notion presented in this paper. In the initial state, only event a (produced by X) is enabled. After firing a synchronously in both systems, a new state is reached. In this state, Y is ready to produce b. However, X is not ready to accept b before τ is produced and, thus, a communication failure occurs when Y fires b and X has not fired τ yet. Therefore, observational equivalence does not imply I/O compatibility.

Figure 1(b) shows that I/O compatibility does not imply observational equivalence. The synchronization of X and Y through the input and output events produces the following language: $(abcd)^*$. In the initial state, X is ready to accept a and b in any order, i.e. they can fire concurrently. However, Y produces a and b sequentially. This situation is reversed for events c and d, accepted concurrently by Y but produced sequentially by X. In either case, the synchronization between X and Y is correct and both systems can interact without any failure. However, it is easy to see that X and Y are not observationally equivalent.

Figure 1(c) depicts another undesired situation. After having produced event a, both systems block waiting for each other to fire some event. Thus, a deadlock is produced. This interaction would be considered "fair" in I/O automata theory [12].

Finally, there is another situation not acceptable for I/O compatible systems: livelock. This situation occurs when one of the systems can manifest an infinite internal behavior without any interaction with the other partner.

1.4 Application to the Synthesis of Reactive Systems

The main objective of this work is to provide a formal framework for characterizing valid transformations of reactive systems during synthesis. Synthesis is the process of transforming a system from a specification to an implementation that uses primitive actions available in some library. For example, a circuit is usually specified in terms of Boolean equations. However, only logic gates with limited fanin are available in a library. For this reason, Boolean equations must be decomposed and matched with logic gates. When synthesizing asynchronous

circuits [3], each logic gate introduces a new internal signal with its associated internal events.

Another example is software synthesis. A compiler is a tool that transforms a high-level specification into assembly code. In this process, many low-level internal actions are introduced (e.g. moving data across internal registers). In case of software synthesis for reactive systems, these internal actions are not observable by the environment.

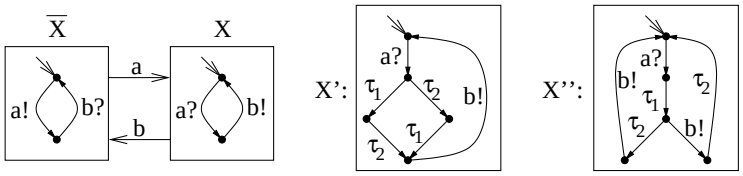

Fig. 2. Transformations for the synthesis of a reactive system

Figure 2 depicts an example of valid and invalid transformations according to the I/O compatibility criterion. The system X is I/O compatible with $\overline{X}$, the mirror of X. Let us assume that, for implementability reasons, two internal actions must be introduced in X, say τ_1 and τ_2. The transformation that leads from X to X' produces the internal events concurrently between a and b. On the other hand, the system X'' produces τ_1 after a and then τ_2 and b concurrently. Even though the transformations from X to X' and X'' preserve observational equivalence, only X' is I/O compatible with $\overline{X}$. If we analyze the interaction between $\overline{X}$ and X'', we observe that the trace $a\tau_1 b$ leads to a state in which $\overline{X}$ can produce the event a but X'' cannot accept it. In this work we will show that *input-properness* is an important property in reactive systems, that guarantees that the receptiveness of input events does not depend on the internal activity of the system.

1.5 Contributions

The contributions of this work are next summarized:

- A formal definition of I/O compatibility, as a relation between the states of two reactive systems is given.
- Safety and liveness properties of I/O compatible systems are proved.
- A polynomial-time decision procedure for I/O compatibility of finite transition systems is presented.
- The relationship between observational equivalence, input-properness and I/O compatibility is studied as a support to propose I/O-compatible transformations during synthesis.
- A kit of Petri net transformations preserving I/O compatibility is presented as an example to support the synthesis of asynchronous circuits.

For simplicity, only I/O compatibility between two systems is considered. The extension to multiple systems would make the nomenclature more tedious, the paper less readable and would not contribute to go deeply into the main concepts of this work. The extension to more than two systems is quite straightforward and left for the reader.

2 Reactive Transition Systems

An event in a reactive system can be input, output or internal. An input event represents an action produced by the environment whereas an output event represents an action produced by the system. Finally, an internal event represents internal actions not observable by the environment. Typical examples of reactive systems are a computer, a television set and a vending machine. The events executed in a reactive system are assumed to take arbitrary but finite time.

Formally, a *Reactive Transition System* is a Transition System [1] where transitions are labeled with events that can occur in a reactive system.

Definition 1 (Reactive Transition System). *A Reactive Transition System (RTS) is a* 4-tuple $A = (S, \Sigma, T, s_{in})$ *where*

- S *is the set of states*
- Σ *is the alphabet of events partitioned into three pairwise disjoint subsets of input* (Σ_I)*, output* (Σ_O) *and internal* (Σ_{INT}) *events.* $\Sigma_{OBS} = \Sigma_I \cup \Sigma_O$ *is called the set of observable events.*
- $T \subseteq S \times \Sigma \times S$ *is the set of transitions*
- $s_{in} \in S$ *is the initial state*

We will call it simply transition system (TS) when the distinction among input, output and internal events is irrelevant.

Definition 2 (Enabling). *An event* e *is* enabled *in the state* s*, denoted by* $\mathsf{En}(s, e)$*, if* $(s, e, s') \in T$*, for some* s'*.*

Reachability in an RTS. The transitions are denoted by (s, e, s') or $s \xrightarrow{e} s'$. The *reachability relation* between states is the transitive closure of the transition relation T. The predicate $s \xrightarrow{\sigma} s'$ denotes a *trace* of events σ that leads from s to s' by firing transitions in T. A state s is *terminal* if no event is enabled in s. An RTS is *finite* if S and T are finite sets. An RTS is *deterministic* if for each state s and each event e there can be at most one state s' such that $s \xrightarrow{e} s'$.

Language of an RTS. An RTS can be viewed as an automaton with alphabet Σ, where every state is an accepting state. For an RTS A, let $L(A)$ be the corresponding language, i.e. its set of traces starting from the initial state.

2.1 Properties of Reactive Transition Systems

Depending on the interpretation of the events in an RTS, different properties can be defined.

Definition 3 (Livelock). *A livelock is an infinite trace of only internal events. An RTS is* livelock-free *if it has no livelocks.*

Livelocks can be detected in polynomial time in finite RTSs. The problem is reduced to the detection of cycles in a graph in which only the edges labeled with internal events are taken into account.

Definition 4 (Input-properness). *An RTS is* input-proper *when for every internal transition $s \xrightarrow{e} s'$, with $e \in \Sigma_{INT}$ and for every input event $i \in \Sigma_I$, $\mathsf{En}(s', i) \implies \mathsf{En}(s, i)$.*

In other words, input-properness is a property that indicates that the enabledness of an input event in a given state depends only on the observable trace leading to that state. Input-properness was introduced in [3] and is a crucial concept to preserve I/O compatibility, as shown later in Sect. 5. It avoids the situations in which the system is doing some "pending" internal work when the environment is producing an input event.

The underlying idea of input-properness was previously presented by Dill [7] when, as a result of hiding an output signal, the same trace could be considered both as success and failure.

Definition 5 (Mirror). *The* mirror *of A, denoted by $\overline{A}$, is another RTS identical to A, but in which the input and output alphabets of A have been interchanged.*

2.2 Synchronous Product

The synchronous product of two transition systems is a new transition system which models the interaction between both systems that synchronize with common events [1]. We define the synchronous product for the class of transition systems, where no partition exists among the set of events. The extension to reactive transitions systems is straightforward.

Definition 6 (Synchronous Product). *Let $A = (S^A, \Sigma^A, T^A, s^A_{in})$, $B = (S^B, \Sigma^B, T^B, s^B_{in})$ be two TSs. The synchronous product of A and B, denoted by $A \times B$ is another TS (S, Σ, T, s_{in}) defined by*

- $s_{in} = \langle s^A_{in}, s^B_{in} \rangle \in S$
- $\Sigma = \Sigma^A \cup \Sigma^B$
- $S \subseteq S^A \times S^B$ *is the set of states reachable from s_{in} according to the following definition of T.*
- *Let $\langle s1, s1' \rangle \in S$.*
  - *If $e \in \Sigma^A \cap \Sigma^B$, $s_1 \xrightarrow{e} s_2 \in T^A$ and $s'_1 \xrightarrow{e} s'_2 \in T^B$, then $\langle s_1, s'_1 \rangle \xrightarrow{e} \langle s_2, s'_2 \rangle \in T$*
  - *If $e \in \Sigma^A \setminus \Sigma^B$ and $s_1 \xrightarrow{e} s_2 \in T^A$, then $\langle s_1, s'_1 \rangle \xrightarrow{e} \langle s_2, s'_1 \rangle \in T$*
  - *If $e \in \Sigma^B \setminus \Sigma^A$ and $s'_1 \xrightarrow{e} s'_2 \in T^B$, then $\langle s_1, s'_1 \rangle \xrightarrow{e} \langle s_1, s'_2 \rangle \in T$*
  - *No other transitions belong to T*

3 I/O Compatibility.

A formal description of the conditions needed for having a correct dialogue between two RTSs is given in this section. We call this set of conditions *I/O compatibility*. The properties of the I/O compatibility can be stated in natural language:

(a) Safeness: if system A can produce an output event, then B must be prepared to accept the event.
(b) Liveness: if system A is blocked waiting for a synchronization with B, then B must produce an output event in a finite period of time.

Theorems 1, 2 and 3 presented below define formally this properties.

Two RTSs are *structurally I/O-compatible* if they share the observational set of events, in a way that they can be connected.

Definition 7 (Structural I/O Compatibility). *Let* $A = (S^A, \Sigma^A, T^A, s^A_{in})$ *and* $B = (S^B, \Sigma^B, T^B, s^B_{in})$ *be two RTSs. A and B are* structurally I/O compatible *if* $\Sigma^A_I = \Sigma^B_O$, $\Sigma^A_O = \Sigma^B_I$, $\Sigma^A \cap \Sigma^B_{INT} = \emptyset$ *and* $\Sigma^B \cap \Sigma^A_{INT} = \emptyset$.

The following definition gives a concise formalization of the conditions needed for characterizing the correct interaction of two RTSs:

Definition 8 (I/O Compatibility). *Let* $A = (S^A, \Sigma^A, T^A, s^A_{in})$ *and* $B = (S^B, \Sigma^B, T^B, s^B_{in})$ *be two structurally I/O compatible RTSs. A and B are* I/O compatible, *denoted by* $A \rightleftharpoons B$, *if A and B are livelock-free and there exists a relation* $R \subseteq S^A \times S^B$ *such that:*

1. $s^A_{in} R s^B_{in}$.
2. Receptiveness (output events of one party are expected by the other party):
   (a) *If* $s_1 R s'_1$, $e \in \Sigma^A_O$ *and* $s_1 \xrightarrow{e} s_2$ *then* $\mathsf{En}(s'_1, e)$ *and* $\forall s'_1 \xrightarrow{e} s'_2 : s_2 R s'_2$.
   (b) *If* $s_1 R s'_1$, $e \in \Sigma^B_O$ *and* $s'_1 \xrightarrow{e} s'_2$ *then* $\mathsf{En}(s_1, e)$ *and* $\forall s_1 \xrightarrow{e} s_2 : s_2 R s'_2$.
3. Internal Progress (internal process preserves the interaction):
   (a) *If* $s_1 R s'_1$, $e \in \Sigma^A_{INT}$ *and* $s_1 \xrightarrow{e} s_2$ *then* $s_2 R s'_1$.
   (b) *If* $s_1 R s'_1$, $e \in \Sigma^B_{INT}$ *and* $s'_1 \xrightarrow{e} s'_2$ *then* $s_1 R s'_2$.
4. Deadlock-freeness (both parties can not be blocked at the same time):
   (a) *If* $s_1 R s'_1$ *and* $\{e \mid \mathsf{En}(s_1, e)\} \subseteq \Sigma^A_I$ *then* $\{e \mid \mathsf{En}(s'_1, e)\} \not\subseteq \Sigma^B_I$.
   (b) *If* $s_1 R s'_1$ *and* $\{e \mid \mathsf{En}(s'_1, e)\} \subseteq \Sigma^B_I$ *then* $\{e \mid \mathsf{En}(s_1, e)\} \not\subseteq \Sigma^A_I$.

Let us consider the examples of Fig. 1. In Fig. 1(a), the receptiveness condition fails and therefore X and Y are not I/O compatible. However, the RTSs of Fig. 1(b) are I/O compatible. Finally, Fig. 1(c) presents an example of violation of the deadlock-freeness condition.

Condition 4 has a strong impact on the behavior of the system. It guarantees that the communication between A and B has no deadlocks (see theorem 3).

Lemma 1. *Let A and B be two RTSs such that* $A \rightleftharpoons B$, *let R be an I/O compatible relation between A and B and let* $A \times B = (S, \Sigma, T, s_{in})$ *be the synchronous product of A and B. Then,* $\langle s, s' \rangle \in S \Rightarrow s R s'$

Proof. If $\langle s, s' \rangle \in S$, then there is a trace σ that leads from s_{in} to $\langle s, s' \rangle$. We prove the lemma by induction on the length of σ.

- Case $|\sigma| = 0$. The initial states are related in Condition 1 of Definition 8.
- Case $|\sigma| > 0$. Let $\sigma = \sigma' e$, with $|\sigma'| = n$, and assume that it holds for any trace up to length n. Let $\langle s_1, s'_1 \rangle$ be the state where the event e is enabled. The induction hypothesis ensures that s_1 is I/O compatible to s'_1. Two situations can happen in s_1 depending on the last event e of σ: either 1) $e \in \Sigma_O \cup \Sigma_{INT}$ is enabled in s_1, or 2) only input events are enabled in s_1. In situation 1), Conditions 2-3 of Definition 8 guarantee that s is I/O compatible to s'. In situation 2), applying Condition 4 of Definition 8 ensure that some non-input event is enabled in state s'_1 of B. Definition 6 and Conditions 2-3 on s'_1 and the enabled non-input event e guarantees s to be I/O compatible to s'. □

Theorem 1 (Safeness). *Let A and B be two RTSs such that $A \rightleftharpoons B$, and a trace $\sigma \in L(A \times B)$ of their synchronous product such that $s_{in} \xrightarrow{\sigma} \langle s, s' \rangle$. If A can fire an output event in s, then the same event is enabled in state s' of B.*

Proof. It immediately follows from Lemma 1 and the condition of receptiveness in the definition of I/O compatibility. □

Theorem 2 (Absence of Livelocks). *Let A and B be two RTSs such that $A \rightleftharpoons B$, and let $A \times B$ be the synchronous product of A and B. Then, $A \times B$ is livelock-free.*

Proof. The definition of synchronous product implies that only livelocks appear in $A \times B$ if either A or B has a livelock. But A and B are livelock-free because $A \rightleftharpoons B$. □

The following theorem is the one that proves the absence of deadlocks produced by the interaction between two I/O compatible RTSs.

Theorem 3 (Liveness). *Let A, B be two RTSs such that $A \rightleftharpoons B$, and a trace $\sigma \in L(A \times B)$ of their synchronous product such that $s_{in} \xrightarrow{\sigma} \langle s, s' \rangle$. If only input events of A are enabled in s, then there exists some trace $\langle s, s' \rangle \xrightarrow{\sigma'} \langle s, s'' \rangle$ such that some of the input events of A enabled in s are also enabled in s'' as output events of B.*

Proof. By Lemma 1 we have that sRs'. We also have that $\{e \mid \mathsf{En}(s, e)\} \subseteq \Sigma_I^A$. By Condition 4 of Definition 8 we know that $\{e \mid \mathsf{En}(s'_1, e)\} \not\subseteq \Sigma_I^B$. Theorem 2 guarantees the livelock-freeness of $A \times B$, and therefore from $\langle s, s' \rangle$ there exists a trace of internal events reaching a state $\langle s, s'' \rangle$ where no internal event is enabled. We know by Lemma 1 that sRs''. Condition 4 of Definition 8, together with the fact that no internal event is enabled in s'' implies that there exists an output event enabled in s'', which is enabled as input in s. □

4 A Polynomial-time Decision Procedure for I/O Compatibility

A procedure for deciding if two finite RTS are I/O compatible is presented in this section. It is based on the synchronous product of transition systems.

Theorem 4. *Let $A = (S^A, \Sigma^A, T^A, s^A_{in})$, $B = (S^B, \Sigma^B, T^B, s^B_{in})$ be two livelock-free RTSs. $A \rightleftharpoons B$ iff $A \times B = (S, \Sigma, T, s_{in})$ fulfills the following properties:*

1. *(a) For each state $s \in S^A$, for each event $e \in \Sigma^A_O$:*
   if $\mathsf{En}(s, e)$ holds and $\langle s, s' \rangle \in S$ then $\mathsf{En}(\langle s, s' \rangle, e)$ holds.
   (b) For each state $s' \in S^B$, for each event $e \in \Sigma^B_O$:
   if $\mathsf{En}(s, e)$ holds and $\langle s, s' \rangle \in S$ then $\mathsf{En}(\langle s, s' \rangle, e)$ holds.
2. *For every $\langle s, s' \rangle \in S$, if $\langle s, s' \rangle \in S$ is a terminal state, then s and s' are terminal states in A and B, respectively.*

Proof. The proof is divided into two parts:

Sufficiency.
Let R be an I/O compatibility relation between A and B and $\langle s, s' \rangle \in S$. Lemma 1 guarantees that sRs'.

1. Since sRs', then $\mathsf{En}(s', e)$ holds in B. By the definition of synchronous product, $\mathsf{En}(\langle s, s' \rangle, e)$ holds. (Similarly for 1(b)).
2. Every non-input event e enabled in s or s' induces e to be enabled in $\langle s, s' \rangle$. If only input events are enabled in one of the states, condition 4 of Definition 8 guarantees the enabling in the other state of a non-input event, and the definition of synchronous product ensures the existence of a transition leaving from $\langle s, s' \rangle$.

Necessity.
We will proof that S is an I/O compatible relation between A and B. State $\langle s^A_{in}, s^B_{in} \rangle$ belongs to S by definition of synchronous product. Let $\langle s, s' \rangle \in S$. Property 1, together with the definition of synchronous product implies the receptiveness condition of Definition 8. Condition 3 (internal progress) of Definition 8 holds by the definition of synchronous product: every internal event e enabled in s (s') is also enabled in $\langle s, s' \rangle$, and the state(s) of S reached by the firing of e in $\langle s, s' \rangle$ are exactly the pairs of I/O compatible states induced by Condition 3 with s and s'. Condition 4 (deadlock-freeness) of Definition 8 also holds: if the events enabled in s are input events, then given that $\langle s, s' \rangle$ is not terminal (due to Property 2), the only possibility for having an event enabled in $\langle s, s' \rangle$ in Definition 6 is when a non-input event is enabled in s'. □

Theorem 4 enables the use of the synchronous product for deciding the I/O compatibility of two finite RTSs in polynomial-time[1]. It consists in computing

[1] Figure 3 shows why it is necessary to consider only livelock-free RTSs in Theorem 4. Systems 1 and 2 are I/O compatible, but System 1 could have a livelock in the state reached after the sequence $b\tau_1 a$.

the synchronous product in the first step, and then checking the conditions 1 and 2 of the theorem.

5 I/O Compatibility and Observational Equivalence.

In the first part of this section, the *observational equivalence* relation [13] is defined. Section 5.2 presents the relationship between I/O compatibility and observational equivalence.

The proofs for the theorems in this sections are not difficult, but tedious. For this reason, they are presented in the appendix.

5.1 Observational Equivalence

The *observational equivalence* relation between two reactive systems was first introduced by Milner in [13]. The relation identifies those systems whose observable behavior is indistinguishable.

Definition 9. *Let* $A = (S^A, \Sigma^A, T^A, s^A_{in})$ *and* $B = (S^B, \Sigma^B, T^B, s^B_{in})$ *be two* RTSs. *A and B are* observational equivalent *(*$A \approx B$*) iff* $\Sigma^A_{OBS} = \Sigma^B_{OBS}$ *and there exists a relation* $R \subseteq S \times S'$ *satisfying*

1. $s^A_{in} R s^B_{in}$.
2. *(a)* $\forall s \in S^A$, $\exists s' \in S^B$ *s.t.* sRs'.
   (b) $\forall s' \in S^B$, $\exists s \in S^A$ *s.t.* sRs'.
3. *(a)* $\forall s_1 \in S^A$, $s'_1 \in S^B$*: if* $s_1 R s'_1$, $e \in (\Sigma^A_{OBS})$ *and* $s_1 \xrightarrow{e} s_2$ *then* $\exists \sigma_1, \sigma_2 \in (\Sigma^B_{INT})*$ *such that* $s'_1 \xrightarrow{\sigma_1 e \sigma_2} s'_2$, *and* $s_2 R s'_2$.
   (b) $\forall s_1 \in S^A$, $s'_1 \in S^B$*: if* $s_1 R s'_1$, $e \in (\Sigma^A_{OBS})$ *and* $s'_1 \xrightarrow{e} s'_2$ *then* $\exists \sigma_1, \sigma_2 \in (\Sigma^A_{INT})*$ *such that* $s_1 \xrightarrow{\sigma_1 e \sigma_2} s_2$, *and* $s_2 R s'_2$.

The two RTSs of Fig. 1(a) are observational equivalent, because every observable sequence of one of them can be executed in the other. Figures 1(b)-(c) depict examples of non-observationally equivalent systems.

5.2 A Sufficient Condition for I/O Compatibility.

A sufficient condition for having I/O compatibility between two reactive systems can be obtained when combining the notions of observational equivalence and input-properness:

Theorem 5. *Let* $A = (S^A, \Sigma^A, T^A, s^A_{in})$, $B = (S^B, \Sigma^B, T^B, s^B_{in})$ *be two livelock-free* RTSs *with* $\Sigma^A_I = \Sigma^B_O$ *and* $\Sigma^A_O = \Sigma^B_I$. *If A and B are input proper and* $A \approx B$, *then* $A \rightleftharpoons B$.

Proof. See appendix.

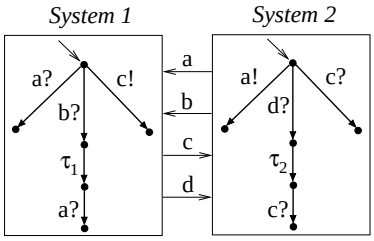

Fig. 3. Two I/O compatible systems that are not input-proper.

When considering a system A and some I/O compatible system B, any transformation of B preserving both input-properness and observational equivalence will lead to another I/O compatible system:

Theorem 6. *Let* $A = (S^A, \Sigma^A, T^A, s^A_{in})$, $B = (S^B, \Sigma^B, T^B, s^B_{in})$ *and* $C = (S^C, \Sigma^C, T^C, s^C_{in})$ *be three* RTSs. *If* $A \rightleftharpoons B$, $B \approx C$, *and* C *is input-proper then* $A \rightleftharpoons C$.

Proof. See appendix.

Figure 2 shows an example of application of Theorem 6. The transformation of X which leads to X' preserves both observational equivalence and input-properness, and then, $\overline{X}$ and X' can safely interact.

Finally, it must be noted that I/O compatibility does not require input-properness, as shown in Fig. 3. This occurs when the non-input-proper situations are not reachable by the interaction of the two systems.

6 Application to the Synthesis of Asynchronous Circuits

Synthesis is the process of transforming a model in such a way that the observable behavior is preserved and the final model commits a set of implementability properties. This section presents a simple synthesis example in the area of asynchronous circuits modeled with Petri nets. I/O compatibility is the property we want to preserve across all transformations from the specification. A good survey on Petri net theory can be found in [15].

A kit of synthesis rules is presented that is valid for deterministic free-choice live and safe Petri nets (FCLSPN) [6]. Under certain conditions, the rules in the kit preserve I/O compatibility. Formal definitions and proofs can be found in [4]. Section 6.2 presents a simple example that shows the usefulness of the transformations.

6.1 I/O Compatible Petri Net Transformations

Three rules are presented for modifying the structure of a Petri net. The rule ϕ_r is used for serializing two concurrent transitions. It was first defined in [2]. Here

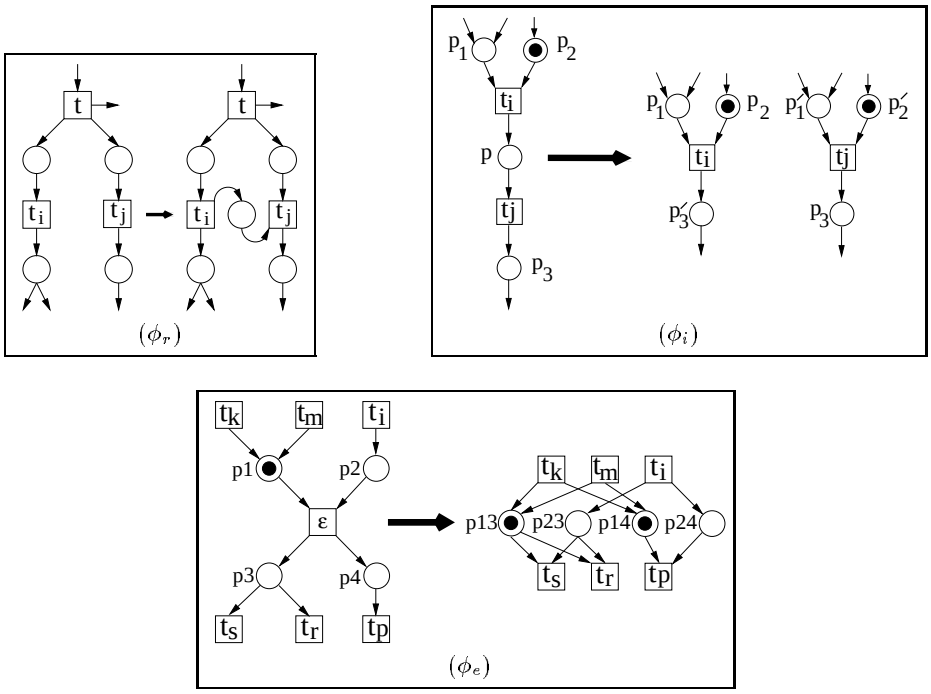

Fig. 4. Kit of Petri net transformations: (ϕ_r) concurrency reduction, (ϕ_i) increase of concurrency, (ϕ_e) transition elimination.

a simplified version is presented. Rule ϕ_i does the opposite: it increases the concurrency between two ordered transitions. ϕ_i can be obtained as a combination of the ones appearing in [15]. Finally, rule ϕ_e hides a transition. It was first presented in [11]. All three rules preserve the liveness, safeness and free-choiceness of the Petri net. In each rule, the conditions for preserving I/O compatibility are also described.

Rule ϕ_r. The purpose of the rule ϕ_r is to eliminate the concurrency between two transitions of the Petri net. This is done by inserting a place that connects the two transitions, ordering their firing. Figure 4 (top left) presents an example of concurrency reduction between transitions t_i and t_j. Rule ϕ_r preserves I/O compatibility when neither t_i nor t_j are transitions labeled with an input event.

Rule ϕ_i. Inversely to rule ϕ_r, rule ϕ_i removes the causality relation between two ordered transitions, making them concurrent. Figure 4 (top right) presents an example of increase of concurrency between transitions t_i and t_j. Rule ϕ_i preserves I/O compatibility when: 1) either t_i or t_j represent a transition of an internal event, and 2) no input-properness violations are introduced by the transformation.

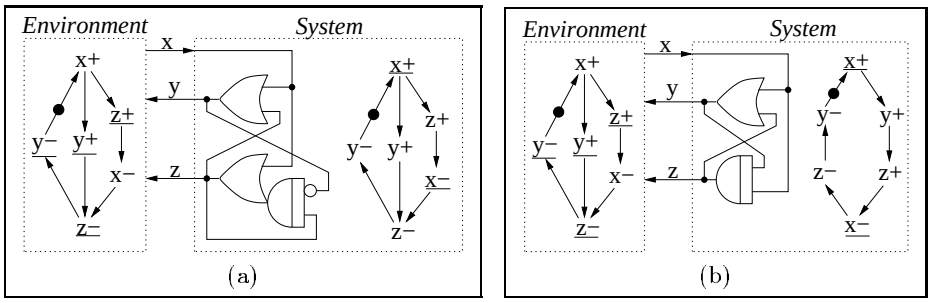

Fig. 5. (a) Mirrored implementation of an asynchronous circuit, (b) valid implementation with concurrency reduction.

Rule ϕ_e. The rule ϕ_e eliminates a transition from the Petri net. Figure 4 (bottom) presents an example of elimination of transition ε. Rule ϕ_e preserves I/O compatibility when ε represents an internal event.

6.2 Synthesis of a Simple Circuit

Figures 5(a-b) depict the example. The models used to describe behavior are marked graphs, a subclass of Petri nets with no choice places, in which events represent rising (+) or falling (-) transitions of digital signals. The goal is to synthesize a circuit that can have a correct dialogue with the environment. We will assume that the components of the circuit have arbitrary delays. Likewise, the environment may take any arbitrary delay to produce any enabled output event.

Let us first have a look at Fig. 5(a). The marked graph in the environment can be considered as a specification of a circuit. The underlined transitions denote input events. Thus, an input event of the environment must have a correspondence with an output event of the system, and vice versa. The behavior denoted by this specification can be informally described as follows:

> *In the initial state, the environment will produce the event $x+$. After that, the environment will be able to accept the events $y+$ and $z+$ concurrently from the system. After the arrival of $z+$, the environment will produce $x-$, that can occur concurrently with $y+$. Next, it will wait for the system to sequentially produce $z-$ and $y-$, thus leading the environment back to the initial state.*

The circuit shown in Fig. 5(a) behaves as specified by the adjacent marked graph. In this case, the behavior of the system is merely a mirror of the behavior of the environment. For this reason, the dialogue between both is correct.

Let us analyze now the system in Fig. 5(b). The marked graph in the system part has been obtained by reducing concurrency between events $y+$ and $z+$, from the marked graph of Fig. 5(a). Still, the system can maintain a correct dialogue, since the environment is able to accept more behaviors than the ones

produced by the system, i.e. the transformation performed preserves I/O compatibility. We can observe that, even though the behavior is less concurrent, the implementation is simpler.

7 Related Work

7.1 Conformation

The notion of *conformation* was defined in [7], where the model used for specifying circuits is a *trace structure*. Conformation models the fact that a specification is correctly realized by a given implementation. A *complete trace structure* is a four-tuple containing the set of input signals (I), the set of output signals (O), the set of traces leading to a success (S) and the set of traces leading to a failure (F), with $S, F \subseteq (I \cup O)^{\infty}$. A complete trace structure models complete executions of a circuit. This allows to express liveness properties.

Given two complete trace structures T and T', T *conforms* to T' ($T \leq T'$) if the composition of T and the mirror of T' is failure-free (i.e. set of failures of the resulting trace structure is empty).

The I/O compatibility can be reformulated to define a concept similar to conformation: for specification A, the system $\overline{A}$ represents a model of the environment where a possible implementation B must correctly interact [7]. We call this relation *I/O preserving realization*:

Definition 10 (I/O Preserving Realization). *Let A and B be two RTSs, A representing the specification of a reactive system. B realizes A ($A \models B$) if $\overline{A} \rightleftharpoons B$.*

I/O preserving realization inherits the liveness property from I/O compatibility: if no deadlocks exist in the interaction between the specification and its environment then the same occurs with its I/O realizable implementation.

7.2 Other Relations

I/O automata [12] is a model similar to RTS. In fact, any RTS can be expressed as an I/O automata by including a failure state that is the sink of transitions labeled with the input events not enabled at each state. In [12], a notion of *automata satisfaction* is presented, expressing when an I/O automata specification is correctly implemented by another I/O automata. The main difference between their satisfaction notion and our realization notion is that we guarantee the absence of deadlock situations in the dialogue between the system and its environment. Moreover, the fact that systems are assumed to be livelock-free allows a local definition of the I/O compatibility, in contrast to the trace-based definition in I/O automata. I/O compatibility has also relations with other equivalences like *testing equivalence* [5], built-in at CIRCAL [8].

In the area of asynchronous systems, several authors have defined different relations to model the concepts of refinement and realization [3, 18, 16, 17,

10]. Among them, we emphasize the one proposed by Brzozowski and Seger [3]. They introduced the concept of input-properness and defined a realization notion stronger than I/O compatibility, that requires language equivalence. In particular, the following theorem can be easily proved.

Theorem 7. *Let A, B be two livelock-free RTSs such that A realizes B under the conditions defined in [3]. Then, $A \models B$.*

Finally, Verhoeff proposed the XDI refinement for delay-insensitive systems. This type of refinement assumes that the dialogue between two systems is produced by introducing any arbitrary delay in the communication, i.e. an event is received some time later than it is produced. Analogously to [7], the expressive power of the XDI model allows to include progress concerns in the model. Differently to the RTS model, the XDI model can not express internal progress (only input/output events are allowed in the model).

8 Conclusions

The theory presented in this paper is only the starting point to support synthesis frameworks that require a kit of transformations that preserve a correct interaction with the environment.

Transformations such as insertion of internal events, reduction/increase of concurrency and so on, are crucial for the synthesis of asynchronous circuits or embedded software, in which concurrent models, e.g. Petri nets, are used to specify the behavior of the system.

Further research is needed to extend the results of this work and derive necessary and sufficient conditions for the preservation of I/O compatibility.

References

1. A. Arnold. *Finite Transition Systems.* Prentice Hall, 1994.
2. G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1986.
3. Janusz A. Brzozowski and Carl-Johan H. Seger. *Asynchronous Circuits.* Springer-Verlag, 1995.
4. J. Carmona, J. Cortadella, and E. Pastor. Synthesis of reactive systems: application to asynchronous circuit design. In J. Cortadella, A. Yakovlev, and G. Rozenberg, editors, *Advances in Concurrency and Hardware Design (ACHD).* Springer-Verlag, 2002. (To appear). Available at http://www.lsi.upc.es/~jcarmona/achd02.ps.gz.
5. R. de Nicola and M. C. B. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34(1-2):83–133, November 1984.
6. J. Desel and J. Esparza. *Free Choice Petri Nets.* Cambridge University Press, Cambridge, Great Britain, 1995.
7. David L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits.* ACM Distinguished Dissertations. MIT Press, 1989.

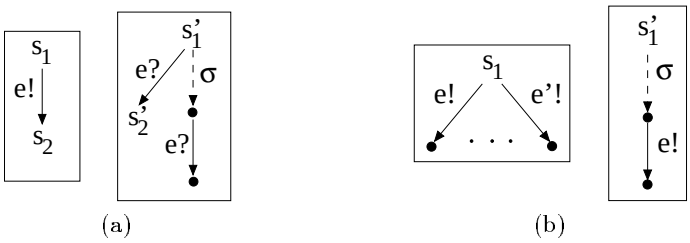

Fig. 6. Conditions 2(a) and 4(a) from the proof of Theorem 5.

8. G.J. Milne. CIRCAL: A calculus for circuit descriptions. *Integration, the VLSI Journal*, 1(2–3):121–160, October 1983.
9. A. Jerraya. Hardware-software codesign. *IEEE Design & Test of Computers*, 17:92–99, March 2000.
10. Mark B. Josephs. A state-based approach to communicating processes. *Distributed Computing*, 3:9–18, 1988.
11. A. V. Kovalyov. On complete reducibility of some classes of Petri nets. In *Proceedings of the 11th International Conference on Applications and Theory of Petri Nets*, pages 352–366, Paris, June 1990.
12. Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. In *CWI-Quarterly*, volume 2, pages 219–246, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, September 1989.
13. R. Milner. *A Calculus for Communicating Processes*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.
14. Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
15. Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–574, April 1989.
16. Radu Negulescu. *Process Spaces and Formal Verification of Asynchronous Circuits*. PhD thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, August 1998.
17. Tom Verhoeff. Analyzing specifications for delay-insensitive circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 172–183, 1998.
18. M. Yoeli and A. Ginzburg. Lotos/cadp-based verification of asynchronous circuits. Report CS-2001-09-2001, Technion - Computer Science Department, September 2001.

A Proofs of Section 5

Proof of Theorem 5.

Proof. Let R be the relation induced by the observational equivalence between A and B. We will prove that R is also an I/O compatibility relation between A and B. R must fulfill the conditions of the I/O compatibility relation:

- **Condition 1:** $s_{in}^{A} R s_{in}^{B}$ by Definition 9.

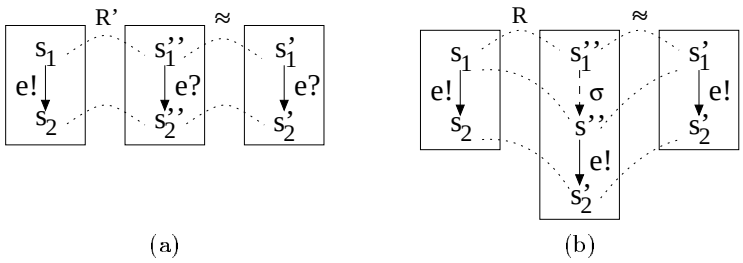

Fig. 7. Conditions 2(a) and 2(b) from the proof of Theorem 6.

- **Condition 2(a):** let $s_1 R s'_1$, and assume $s_1 \xrightarrow{e} s_2$, with $e \in \Sigma_O^A$. Figure 6(a) depicts the situation. The observational equivalence of s_1 and s'_1 implies that a trace σ of internal events exists in s'_1 enabling e. The event e is an input event in B, and therefore the input-properness of B ensures that in every state s' of σ, $\mathsf{En}(s', e)$ holds. In particular, it also holds in the first state and, thus, $\mathsf{En}(s'_1, e)$. The definition of R ensures that every s'_2 such that $s'_1 \xrightarrow{e} s'_2$ is related with s_2 by R.
- **Condition 3(a):** let $s_1 R s'_1$ and assume $s_1 \xrightarrow{e} s_2$, with $e \in \Sigma_{INT}^A$. The definition of R implies that $s_2 R s'_1$.
- **Condition 4(a):** let $s_1 R s'_1$, and suppose $\{e \mid \mathsf{En}(s_1, e)\} \subseteq \Sigma_I^A$. Figure 6(b) depicts the situation. Let e be one of the input events enabled in s_1. The observational equivalence between s_1 and s'_1 requires that a sequence σ of internal events exists enabling e starting in s'_1, and given that e in not input in B implies $\{e \mid \mathsf{En}(s'_1, e)\} \not\subseteq \Sigma_I^B$.

An identical reasoning can be applied in the symmetric cases (conditions 2(b), 3(b) and 4(b)). □

Proof of Theorem 6.

Proof. Let R' be the relation between A and B, and $\approx$ the observational equivalent relation between states from B and C. Define the relation R as:

$$\forall s \in S^A, s'' \in S^B, s' \in S^C : sR's'' \wedge s'' \approx s' \Leftrightarrow (s, s') \in R$$

The conditions that R must satisfy are the ones of Definition 8. Remember that $A \rightleftharpoons B$ implies that $\Sigma_O^B = \Sigma_I^A$ and $\Sigma_I^B = \Sigma_O^A$. Moreover, relation $B \approx C$ implies that $\Sigma_{OBS}^B = \Sigma_{OBS}^C$.

- **Condition 1:** the initial states are related in R by definition.
- **Condition 2(a):** let $s_1 R s'_1$, and suppose $s_1 \xrightarrow{e} s_2$ with $e \in \Sigma_O^A$. Figure 7(a) depicts the situation. Given that $s_1 R' s''_1$, e is enabled in s''_1 and for each s''_2 such that $s''_1 \xrightarrow{e} s''_2$, $s_2 R' s''_2$. The observational equivalence of s''_1 and s'_1,

together with the fact that C is input-proper implies that e is also enabled in s'_1 (identical reasoning of condition 2(a) in Theorem 5), and the definition of $\approx$ implies that each s'_2 such that $s'_1 \xrightarrow{e} s'_2$ must be related in $\approx$ with s''_2. Then each s'_2 such that $s'_1 \xrightarrow{e} s'_2$ is related by R with s_2.

- **Condition 2(b):** let $s_1 R s'_1$, and suppose $s'_1 \xrightarrow{e} s'_2$ with $e \in \Sigma_O^B$. Figure 7(b) depicts the situation. The observational equivalence of s''_1 and s'_1 implies that there is a sequence σ of internal events starting in s''_1 and enabling e, and every state of σ is observational equivalent to s'_1. Moreover, every state of σ is also related to s_1 by the condition 3(b) of R'. In particular, s_1 is related by R' with the state s'' of σ s.t. $s'' \xrightarrow{e} s''_2$; applying Condition 2(b) of R', $\mathsf{En}(s_1, e)$ holds and for each e s.t. $s_1 \xrightarrow{e} s_2$, $s_2 R' s''_2$. The definition of R and $\approx$ induces that each such s_2 is related with s'_2 by R.
- **Condition 3(a):** let $s_1 R s'_1$, and suppose $s_1 \xrightarrow{e} s_2$ with $e \in \Sigma_{INT}^A$. Then Condition 3(a) of R' ensures $s_2 R' s''_1$ and then applying the definition of R implies $s_2 R s'_1$.
- **Condition 3(b):** let $s_1 R s'_1$, and suppose $s'_1 \xrightarrow{e} s'_2$ with $e \in \Sigma_{INT}^C$. Then $s''_1 \approx s'_2$, and then $s_1 R s'_2$.
- **Condition 4(a):** let $s_1 R s'_1$, and suppose $\{e | \mathsf{En}(s_1, e)\} \subseteq \Sigma_I^A$. Condition 4(a) of R' ensures that $\{e | \mathsf{En}(s''_1, e)\} \not\subseteq \Sigma_I^B$: let a be an event such that $s''_1 \xrightarrow{a} s''_2$, with $a \notin \Sigma_I^B$. If $a \in \Sigma_O^B$, the related pair $s''_1 \approx s'_1$ ensures that in s'_1 there is a feasible sequence of internal events (which can be empty) enabling a, and therefore $\{e | \mathsf{En}(s'_1, e)\} \not\subseteq \Sigma_I^C$. If $a \in \Sigma_{INT}^B$, applying Condition 3(b) of R' and the definition of $\approx$, $s_1 R' s''_2$ and $s''_1 \approx s'_1$ is obtained, respectively. The same reasoning applied to s_1, s''_1 and s'_1 can now be applied to s_1, s''_2 and s'_1. Given that B is livelock-free, the sequence of internal events starting in s''_1 and passing through s''_2 must end in a state s'' where a observable event a' is enabled. State s'' is also related by R' with s_1, and by $\approx$ with s'_1 (applying inductively the same reasoning applied to s''_2). Event a' belongs to Σ_O^B because otherwise a violation of Condition 2(b) in R' arise. The previous case ($a \in \Sigma_O^B$, enabled in s''_1) can be applied to s''.
- **Condition 4(b):** let $s_1 R s'_1$, and suppose $\{e | \mathsf{En}(s'_1, e)\} \subseteq \Sigma_I^C$. Let a such that $s''_1 \xrightarrow{a} s''_2$. If $a \in \Sigma_O^B$, then a contradiction arise because $s''_1 \approx s'_1$ and $\{e | \mathsf{En}(s'_1, e)\} \subseteq \Sigma_I^C$. If $a \in \Sigma_I^B$, then identical conditions make $\mathsf{En}(s'_1, a)$ to hold. If $a \in \Sigma_{INT}^B$, then Conditions 3(a) of R' and $\approx$ ensure that $s_1 R' s''_2$ and $s''_2 \approx s'_1$, and the same reasoning of s_1, s'_1 and s''_1 can be applied to s_1, s'_1 and s''_2 (but not infinite times, because B is livelock-free). Therefore a feasible sequence of internal events (which can be empty) exist from s''_1 reaching a state s'' such that $\{e | \mathsf{En}(s'', e)\} \subseteq \Sigma_I^C$, with $s_1 R' s''$ and $s'' \approx s'_1$. Condition 4(b) of R' ensures that $\{e | \mathsf{En}(s_1, e)\} \not\subseteq \Sigma_I^A$. □