On the Complexity of Input Output Conformance Testing

Neda Noroozi[1], Mohammad Reza Mousavi[2], and Tim A.C. Willemse[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Center for Research on Embedded Systems, Halmstad University, Sweden
n.noroozi@tue.nl, m.r.mousavi@hh.se,t.a.c.willemse@tue.nl

Abstract. Input-output conformance (ioco) testing is a well-known approach to model-based testing. In this paper, we study the complexity of checking ioco. We show that the problem of checking ioco is PSPACE-complete. To provide a more efficient algorithm, we propose a more restricted setting for checking ioco, namely with deterministic models and show that in this restricted setting ioco checking can be performed in polynomial time.

1 Introduction

Motivation. Testing is a major part of the software development process and, together with debugging, accounts for more than half of the development cost and effort [14]. Model-based testing is a structured and rigorous discipline of testing, which is likely to improve the current practice of testing [24, 23, 17]. Input-output conformance (**ioco**) testing is a well-known formal approach to model-based testing, which is used extensively in various practical applications, see [13, 5] and the references therein, and which has been the subject of much theoretical research, see [21] and the references therein.

In this paper, we study the complexity of checking **ioco**, a topic which—as far as we could trace—has not been addressed in the literature. Our study sheds some light on the theoretical boundaries for this popular notion and possible enhancements in its efficiency and efficacy by considering restricted forms of specifications and implementations.

We first show that the upper bound on the complexity of checking **ioco** is exponential in the size of the model. This is as expected due to the trace-based nature of (the intensional definition of) **ioco**. We show that this exponential complexity bound is indeed tight, by proving that the problem is PSPACE-complete. This means that, unless the complexity class hierarchy from P to PSPACE collapses, the exponential time complexity in deciding **ioco** is unavoidable in the worst case. Next, we identify a more restricted setting for checking **ioco** which still admits a polynomial time algorithm. In this restricted setting, implementations are still permitted to behave non-deterministically, but specifications must be deterministic. In order to obtain this result, we first give a coinductive (simulation-like) definition of **ioco** for deterministic specifications and we subsequently show that it can be decided in polynomial time.

Our study is based on the intensional representation of **ioco**, which allows for defining exact complexity bounds on checking conformance. The complexity bounds for the intensional representation also hold for the extensional representation, but it remains to be checked under which conditions these bounds can be realized in the practical setting by using test-cases.

Compositional testing concerns about testing of composite systems consisting of communication components which can be separately tested. Though **ioco** lacks the compositionality property in general, several researches were conducted to adapt **ioco** for compositional testing. In [6, 9, 15] some variants of **ioco** have been introduced for testing of component-based systems. Our results in this paper can be easily adapted and applied for those relations as well. For instance, in [6], a variant of **ioco** for testing components is introduced such that the correctness of the integration of the conformed components is guaranteed. That aforementioned relation coincides with the standard **ioco** for a restricted class of specification, namely deterministic models.

Related work. Our polynomial time algorithm for checking **ioco** (for deterministic models) is inspired by [18], which is based on the reduction of checking **ioco** into the **NHORNSAT** problem [8]. The coinductive definition of **ioco**, which is an important means for this result, is akin to the alternating refinement [4] of Interface Automata [3] (see also [1, 10]). In [22], it is shown that for deterministic models and implementations, alternating refinement coincides with **ioco**. In this paper, we show that our coinductive definition of **ioco** coincides with **ioco** for deterministic models (and possibly nondeterministic implementations). In a recent paper [10], a simulation-like relation, called iocos, is presented. It is shown that iocos is finer than **ioco**.

In [11], the author proves that testing conformance under asynchronous communication is in general EXPTIME-hard. Then, by restricting to a particular class of models, called observable IOTSs, the author gives a polynomial time algorithm for checking conformance under asynchronous FIFO communication. (The problem remains equally hard for observable models under arbitrary asynchronous communication.) Apart from being in the asynchronous setting, the notion of conformance used in [11] differs from **ioco** (e.g., its theory does not treat quiescence).

Structure of the Paper. In Section 2, we recall some basic definitions regarding labeled transition systems and the input-output conformance relation. In Section 3, we study the complexity of checking **ioco**. In order to obtain more efficient bounds for checking conformance of deterministic models, we first give a coinductive definition of **ioco** in Section 4 and use this definition in Section 5 to show that in this restricted setting, conformance checking is indeed possible in polynomial time. We conclude the paper in Section 6.

2 Preliminaries

In this section, we briefly repeat the definitions of the formal models used in our context for specifying system behavior, as well as the notion of input-output conformance testing. Throughout this paper, we use variants of Labeled Transition Systems (LTSs) for modeling the behavior of specifications and implementations. The LTS model assumes that systems can be represented using a set of states and transitions, labeled with events or actions, between such states. The events leading to new states can be observed by the tester, but the states cannot be inspected. We assume the presence of a special action, denoted by τ, which models an event that is unobservable to the tester.

Definition 1 (LTS). *A labeled transition system (LTS) is a 4-tuple $\langle S, L, \rightarrow, \bar{s} \rangle$, where S is a set of states, L is a finite alphabet of actions that does not contain the internal action τ, $\rightarrow \subseteq S \times (L \cup \{\tau\}) \times S$ is the transition relation, and $\bar{s} \in S$ is the initial state.*

Throughout this section, we assume a fixed yet arbitrary LTS $\langle S, L, \rightarrow, \bar{s} \rangle$. We tend to refer to LTSs by referring to their initial state, i.e., $\bar{s}$ in the case of the above mentioned LTS. Let $s, s' \in S$ and $x \in L \cup \{\tau\}$. In line with common practice, we write $s \xrightarrow{x} s'$ rather than $(s, x, s') \in \rightarrow$. Furthermore, we write $s \xrightarrow{x}$ whenever $s \xrightarrow{x} s'$ for some $s' \in S$, and $s \not\xrightarrow{x}$ when not $s \xrightarrow{x}$. The transition relation is generalized to a relation over a sequence of actions by the following deduction rules:

$$\frac{}{s \xrightarrow{\epsilon}{}^{*} s} \qquad \frac{s \xrightarrow{\sigma}{}^{*} s'' \quad s'' \xrightarrow{x} s' \quad x \neq \tau}{s \xrightarrow{\sigma x}{}^{*} s'} \qquad \frac{s \xrightarrow{\sigma}{}^{*} s'' \quad s'' \xrightarrow{\tau} s'}{s \xrightarrow{\sigma}{}^{*} s'}$$

We tacitly adopt the same notational conventions both for $\rightarrow$ and $\rightarrow^{*}$.

An LTS $\bar{s}$ is said to be *deterministic* if the set of states reached after executing any sequence of actions is always a singleton set; that is, for all $s, s', s'' \in S$ and all $\sigma \in L^{*}$, if $s \xrightarrow{\sigma}{}^{*} s'$ and $s \xrightarrow{\sigma}{}^{*} s''$ then $s' = s''$.

A state in the LTS $\bar{s}$ is said to *diverge* if it is the source of an infinite sequence of τ-labeled transitions. The LTS $\bar{s}$ is *divergent* if one of its reachable states diverges. Throughout this paper, we confine ourselves to non-divergent LTSs.

Definition 2. *Let $s' \in S$ and $S' \subseteq S$. The set of* traces*,* enabled actions *and* weakly enabled actions *for s and S' are defined as follows:*

- $\mathsf{traces}(s) = \{\sigma \in L^{*} \mid s \xrightarrow{\sigma}{}^{*}\}$, *and* $\mathsf{traces}(S') = \bigcup_{s' \in S'} \mathsf{traces}(s')$.
- $\mathsf{init}(s) = \{x \in L \cup \{\tau\} \mid s \xrightarrow{x}\}$, *and* $\mathsf{init}(S') = \bigcup_{s' \in S'} \mathsf{init}(s')$.
- $\mathsf{Sinit}(s) = \{x \in L \mid s \xrightarrow{x}{}^{*}\}$, *and* $\mathsf{Sinit}(S') = \bigcup_{s' \in S'} \mathsf{Sinit}(s')$.

Input, output, Quiescence and Suspension Traces. When engaging in interaction with another system, the actions of an LTS are often assumed to be partitioned into two subcategories, reflecting which of the systems has the initiative in executing the action. Output actions are under the control of the system, whereas input actions are under the control of the environment of the system. We refine the LTS model to reflect this distinction in initiative.

Definition 3 (IOLTS). *An input-output labeled transition system (IOLTS) is a tuple $\langle S, I, U, \rightarrow_s, \bar{s}\rangle$ such that the tuple $\langle S, I \cup U, \rightarrow, \bar{s}\rangle$ is an LTS in which the alphabet L is partitioned into a set I of inputs and a set U of outputs.*

Testers often not only have the power to observe the events produced by an implementation, but also can observe the *absence* of events, or *quiescence* [21]. A state $s \in S$ is said to be *quiescent* if it does not produce outputs and it is *stable*, that is, it cannot, through internal computations, evolve to a state that *is* capable of producing outputs. Formally, state s is quiescent, denoted $\delta(s)$, whenever $\mathsf{init}(s) \subseteq I$. In order to formally reason about the observations of inputs, outputs and quiescence, we introduce the set of *suspension traces*. To this end, we first generalize the transition over a sequence of input, output and quiescence actions. Let L_δ denote the set $L \cup \{\delta\}$.

$$\frac{s \xrightarrow{\sigma}{}^{*} s'}{s \overset{\sigma}{\Longrightarrow} s'} \qquad \frac{\delta(s)}{s \overset{\delta}{\Rightarrow} s} \qquad \frac{s \overset{\sigma}{\Longrightarrow} s'' \qquad s'' \overset{\rho}{\Longrightarrow} s'}{s \overset{\sigma\rho}{\Longrightarrow} s'}$$

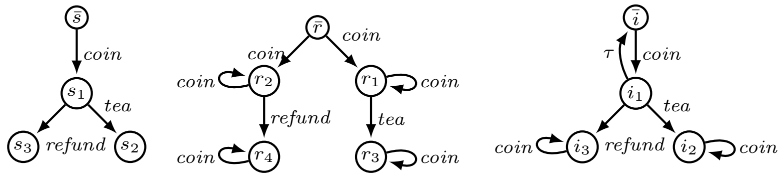
The following definition formalizes the set of suspension traces.

Definition 4. *Let $s \in S$ and $S' \subseteq S$. The set of* suspension traces *for s, denoted by $\mathsf{Straces}(s)$ is defined as the set $\{\sigma \in L_\delta^* \mid s \overset{\sigma}{\Rightarrow}\}$; we set $\mathsf{Straces}(S') = \bigcup\limits_{s' \in S'} \mathsf{Straces}(s')$.*

Input-Output Conformance Testing with Quiescence. Tretmans' **ioco** testing theory [21] is a formal approach to conformance testing. It assumes that the behavior of implementations can be described adequately using a class of IOLTSs, called *input output transition systems*; this assumption is the so-called *testing hypothesis*. Input output transition systems are essentially plain IOLTSs with the additional assumption that inputs can always be accepted.

Definition 5 (IOTS). *Let $\langle S, I, U, \rightarrow, \bar{s}\rangle$ be an IOLTS. A state $s \in S$ is* input-enabled *iff $I \subseteq \mathsf{Sinit}(s)$; the IOLTS $\bar{s}$ is an* input output transition system *(IOTS) iff every state $s \in S$ is input-enabled. The class of input output transition systems ranging over inputs I and outputs U is denoted $\mathsf{IOTS}(I, U)$.*

While the **ioco** testing theory assumes input-enabled implementations, it does not impose this requirement on specifications. This facilitates testing using partial specifications, *i.e.*, specifications that are under-specified. To simplify presenting the input-output conformance relation (**ioco**), we first introduce the formal definitions below.



(a) IOLTS $\bar{s}$ (b) IOTS $\bar{r}$ (c) IOTS $\bar{i}$

Fig. 1. A specification $\bar{s}$ of a tea vending machine, a correct implementation $\bar{r}$ and an incorrect implementation $\bar{i}$.

Definition 6. *Let $\langle S, I, U, \rightarrow, \bar{s}\rangle$ be an IOLTS. Let $s \in S$, $S' \subseteq S$ and let $\sigma \in L_\delta^*$.*

- s after $\sigma = \{s' \in S \mid s \overset{\sigma}{\Rightarrow} s'\}$*, and* S' after $\sigma = \bigcup_{s' \in S'} s'$ after σ.
- $\mathsf{out}(s) = \{x \in L_\delta \setminus I \mid s \overset{x}{\Rightarrow}\}$*, and* $\mathsf{out}(S') = \bigcup_{s' \in S'} \mathsf{out}(s')$.

The **ioco** conformance relation [21] is then defined as follows.

Definition 7 (ioco). *Let $\langle Q, I, U, \rightarrow, \bar{r}\rangle$ be an IOTS representing a realization of a system, and let IOLTS $\langle S, I, U, \rightarrow, \bar{s}\rangle$ be a specification. We say that $\bar{r}$ is* input output conform *with specification $\bar{s}$, denoted by $\bar{r}\,$***ioco***$\,\bar{s}$, iff*

$$\forall \sigma \in \mathsf{Straces}(\bar{s}):\;\; \mathsf{out}(\bar{r}\ \mathsf{after}\ \sigma) \subseteq \mathsf{out}(\bar{s}\ \mathsf{after}\ \sigma)$$

Example 1. Consider the IOLTSs pictured in Figure 1. The IOLTS $\bar{s}$ is a specification of a vending machine which sells tea. After receiving a coin, it either delivers tea or refunds the coin. The IOLTS $\bar{r}$ is a formal model of a possible implementation of this vending machine. Upon receiving a coin, the machine $\bar{r}$ chooses non-deterministically between serving tea or refunding the coin. Note that IOLTS $\bar{r}$ is input-enabled, because it accepts input action *coin* (as the only input action) at every state. The set $\mathsf{Straces}(\bar{s})$ is given by the regular expression $(\delta^*) \mid (\delta^*\, coin) \mid (\delta^*\, coin(tea|refund)\delta^*)$. Clearly, for all $\sigma \in \mathsf{Straces}(\bar{s})$, we have $\mathsf{out}(\bar{r}\ \mathsf{after}\ \sigma) \subseteq \mathsf{out}(\bar{s}\ \mathsf{after}\ \sigma)$. Thus, $\bar{r}\,$**ioco**$\,\bar{s}$.

The IOLTS $\bar{i}$ is a formal model of an implementation of a malfunction vending machine. After receiving a coin, it either delivers tea, refunds the coin or does nothing. Similar to IOTS $\bar{r}$, it accepts input action *coin* at every state. Thus, the IOLTS $\bar{i}$ is input-enabled. Consider the trace *coin* after which $\mathsf{out}(\bar{s}) = \{refund, tea\}$ while $\mathsf{out}(\bar{i}) = \{refund, tea, \delta\}$. As a result, we find that $\bar{i}\,$~~**ioco**~~$\,\bar{s}$.

3 Conformance Checking for Nondeterministic Models

In this section, we study the complexity of input-output conformance checking in full generality. We prove that, in the general case, checking **ioco** is PSPACE-complete. To this end, we first show that checking **ioco** is in PSPACE. Subsequently, we show that checking **ioco** is at least as hard as other canonical

PSPACE-complete problems and hence, is also PSPACE-complete. We prove these results by means of two polynomial time reductions, respectively, to and from the language inclusion problem for regular expressions (or NFAs); the latter problem is well-known to be PSPACE-complete; see [19] for the classical result and [2, 16] for some recent developments.

Theorem 1. *The problem of checking* **ioco** *is in PSPACE.*

Proof. We prove the thesis by showing that the problem of checking **ioco** for input-enabled specifications is reducible to the language inclusion problem of NFAs with ϵ-moves (hereafter simply referred to as NFAs) in polynomial time. Observe that without loss of generality we can restrict our attention to the problem of checking **ioco** for input-enabled specifications. Indeed, for a non input-enabled specification A_2, we can construct an input-enabled specification $\bar{A}_2$ in polynomial time such that for any implementation A_1, we have $A_1 \,\mathbf{ioco}\, A_2$ iff $A_1 \,\mathbf{ioco}\, \bar{A}_2$. The construction uses a standard angelic completion, adding missing input transitions to A_2 that lead to fresh input-enabled states that accept all outputs and quiescence.

Assume that, for $i \in \{1, 2\}$, we have IOTSs A_i of the form $\langle S_i, I, U, \rightarrow_i, \bar{s_i}\rangle$. Our reduction proceeds as follows. We define NFAs $A'_i = \langle Q_i, \Sigma, \Delta_i, q_i, F\rangle$ as follows:

- $Q_i = S_i$ is the set of states,
- $\Sigma = L_\delta \cup \{\epsilon\}$ where $L_\delta = I \cup U \cup \{\delta\}$ is the common alphabet,
- $\Delta_i = \{(q, a, q') \mid q \xrightarrow{a} q' \wedge a \in L\} \cup \{(q, \epsilon, q') \mid q, q' \in S_i \wedge q \xrightarrow{\tau} q'\} \cup \{(q, \delta, q) \mid q \in S_i \wedge \delta(q)\}$ is the transition relation, which is that of the corresponding IOTS union with δ-labeled self-loop for single and each quiescent state.
- $q_i = \bar{s_i}$ is the initial state, and
- $F = Q_i$ is the set of final states.

Note that the above reduction is carried out linearly in the size of the transition relations of A_1 and A_2. Moreover, observe that $L(A'_1) = \mathsf{Straces}(A_1)$ and $L(A'_1) = \mathsf{Straces}(A_2)$. We next proceed by showing that $A_1 \,\mathbf{ioco}\, A_2$ if and only if $\mathsf{Straces}(A'_1) \subseteq \mathsf{Straces}(A'_2)$. We prove the contraposition of both implications separately.

- Assume that $A_1 \,\not\mathbf{ioco}\, A_2$. By definition of **ioco** there is a suspension trace σ in specification A_2 such that for some output x, $\sigma x \in \mathsf{Straces}(A_1)$, but $\sigma x \notin \mathsf{Straces}(A_2)$. Since $L(A'_1) = \mathsf{Straces}(A_1)$ and $L(A'_2) = \mathsf{Straces}(A_2)$ we find $L(A'_1) \not\subseteq L(A'_2)$.
- Assume $L(A'_1) \not\subseteq L(A'_2)$. Then there is a word $\sigma \in L(A'_1) \setminus L(A'_2)$. Without loss of generality, assume $\sigma = \rho x$ for some $\rho \in L(A'_1) \cap L(A'_2)$ and $x \in \Sigma$. Since $L(A'_1) = \mathsf{Straces}(A_1)$ and $L(A'_2) = \mathsf{Straces}(A_2)$, we have $\rho x \in \mathsf{Straces}(A_1) \setminus \mathsf{Straces}(A_2)$. We distinguish two cases:
  - Suppose $x \in I$. Since, A_2 is input-enabled, we know that $\rho a \in \mathsf{Straces}(A_2)$ for all $a \in I$. In particular, $\rho x \in \mathsf{Straces}(A_2)$, contradicting $\rho x \notin \mathsf{Straces}(A_2)$. Therefore, $x \in I$ cannot be the case.

- Suppose $x \in U \cup \{\delta\}$. Therefore, $x \in \mathsf{out}(\bar{s_1} \ \mathsf{after} \ \rho)$ but $x \notin \mathsf{out}(\bar{s_2} \ \mathsf{after} \ \rho)$. By definition of **ioco** relation we have $A_1 \,\cancel{\textbf{ioco}}\, A_2$.

We next establish that the problem of checking **ioco** is in fact PSPACE-complete.

Theorem 2. *The problem of checking **ioco** is PSPACE-complete.*

Proof. We prove the thesis by providing a linear reduction of the PSPACE-complete language inclusion problem for regular expressions to checking **ioco**. Every regular expression can be translated linearly to a language equivalent NFA, following Kleene's theorem. In particular, we may assume that the language-equivalent NFA of a regular expression has one initial and one final state, all states are reachable from the initial state and can reach the final state, there is no incoming transition to the initial state and no outgoing transition from the final state (e.g., by applying Thompson's algorithm for converting regular expressions to NFAs [20]).

Formally, let RE_1 and RE_2 be two regular expressions over alphabet Σ and assume A_1 and A_2 are the language-equivalent NFAs for RE_1 and RE_2. The inclusion problem of regular expressions of RE_1 and RE_2 is equivalent to the problem whether $L(A_1) \subseteq L(A_2)$. As stated above, we may assume that NFA A_i is of the form $\langle Q_i, \Sigma \cup \{\epsilon\}, \Delta_i, q_i, \{f_i\}\rangle$. We define IOTSs $A'_i = \langle S_i, I, U, \rightarrow_i, \bar{s}_i\rangle$ as follows:

- $S_i = Q_i$,
- $I = \{i\}$, where $i \notin \Sigma$ is a fresh symbol,
- $U = \Sigma$,
- $\rightarrow_i = \{(q, a, q') \mid (q, a, q') \in \Delta_i \wedge a \in \Sigma\} \cup \{(q, \tau, q') \mid (q, \epsilon, q') \in \Delta_i\} \cup \{(q, i, q) \mid q \in Q_i\}$, i.e., the transition relation is that of the corresponding automaton union with i-labeled self-loops for each and every state,
- $\bar{s}_i = q_i$.

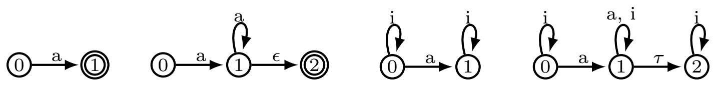
Note that the two IOLTSs A'_1 and A'_2 obtained from the above reduction are input-enabled, because $\{i\} \subseteq \mathsf{Sinit}(s)$ for all s in both A'_1 and A'_2. Moreover, the accepting states in A_1 and A_2 are the only quiescent states in A'_1 and A'_2. We proceed to show that language inclusion of A_1 in A_2 can be decided by checking for **ioco**; that is, we prove $L(A_1) \subseteq L(A_2)$ if and only if $A'_1 \,\textbf{ioco}\, A'_2$. The contraposition of each implication is again proved separately.

- Assume that $L(A_1) \not\subseteq L(A_2)$. Thus, there is a word $\sigma \in \Sigma^*$ such that $\sigma \in L(A_1)$ but $\sigma \notin L(A_2)$. Therefore, the accepting state f_1 in NFA A_1 is reachable after σ. By construction, the state f_1 state in A'_1 is quiescent. Thus, $\delta \in \mathsf{out}(A'_1 \ \mathsf{after} \ \sigma)$. We distinguish two cases.
  - Suppose there is a state in automaton A_2 which is reachable after σ. Thus, $\sigma \in \mathsf{Straces}(A'_2)$. Since $\sigma \notin L(A_2)$, we know that A_2 does not reach its accepting state f_2 after σ. Therefore, $\delta \notin \mathsf{out}(A'_2 \ \mathsf{after} \ \sigma)$. Since $\delta \in \mathsf{out}(A'_1 \ \mathsf{after} \ \sigma)$ and $\sigma \in \mathsf{Straces}(A'_2)$, we find that $A'_1 \,\cancel{\textbf{ioco}}\, A'_2$, which was to be shown.

  - Assume there is no state in automaton A_2 that is reachable after σ. Then also $\sigma \notin \mathsf{Straces}(A'_2)$. Let $\rho x \in \Sigma^+$ be a prefix of σ such that $\rho \in \mathsf{Straces}(A'_2)$ but $\rho x \notin \mathsf{Straces}(A'_2)$. Note that such a prefix must exist. Since $\sigma \in \mathsf{Straces}(A'_1)$, we find that $\rho x \in \mathsf{Straces}(A'_1)$. Therefore $A'_1 \mathrel{\not\!\mathbf{ioco}} A'_2$, which was to be shown.

- Assume that $A'_1 \mathrel{\not\!\mathbf{ioco}} A'_2$. Thus, there is a $\sigma \in \mathsf{Straces}(A'_1) \cap \mathsf{Straces}(A'_2)$ and an output $x \in \Sigma \cup \{\delta\}$ such that $\sigma x \in \mathsf{Straces}(A'_1)$ but $\sigma x \notin \mathsf{Straces}(A'_2)$.
  We first define the projection operator $\_{}_{\downarrow}$ over the sequences in $\Sigma \cup \{i, \delta\}$. Let $\gamma \in (\Sigma \cup \{i, \delta\})^*$ and $a \in \Sigma \cup \{i, \delta\}$. Then $(\gamma a)_\downarrow = (\gamma)_\downarrow a$ when $a \in \Sigma$, and $(\gamma a)_\downarrow = (\gamma)_\downarrow$ otherwise.
  Since, by construction, only transitions labeled with an action in Σ invoke state changes in IOLTS A'_1, for all γ we have $(A'_1 \;\mathsf{after}\; \gamma) = (A'_1 \;\mathsf{after}\; \gamma_\downarrow)$, and, similarly for A'_2. Therefore, without loss of generality we may assume that $\sigma x = (\sigma x)_\downarrow$; i.e., $\sigma x \in (\Sigma \cup \{\delta\})^*$. We distinguish two cases, based on the type of x.
  - Assume that $x \in \Sigma$. Since $\sigma x \in \mathsf{Straces}(A'_1) \cap \Sigma^*$, there is a state in A_1 that is reachable after σx. Since the accepting state f_1 in A_1 is reachable from every all states in A_1, there must be a $\rho \in \Sigma^*$ such that $\sigma x \rho \in L(A_1)$. From $\sigma x \notin \mathsf{Straces}(A'_2)$ and $\sigma x \in \Sigma^*$ we can deduce that no state in A_2 can be reached via the word σx. Consequently, the extended word $\sigma x \rho$ is also not accepted by A_2; i.e., $\sigma x \rho \notin L(A_2)$. Since $\sigma x \rho \in L(A_1)$, we conclude that $L(A_1) \not\subseteq L(A_2)$.
  - Assume that $x \notin \Sigma$; it then follows that $x = \delta$. Thus, $\delta \in \mathsf{out}(A'_1 \;\mathsf{after}\; \sigma)$. By our construction, a δ-labeled transition is enabled only at state f_1 in A'_1 and state f_2 in A'_2. From this, it follows that word σ is accepted by A_1, i.e., $\sigma \in L(A_1)$. Following a similar line of reasoning, we conclude from $\delta \notin \mathsf{out}(A'_2 \;\mathsf{after}\; \sigma)$ that the word σ is not accepted by A_2, i.e., $\sigma \notin L(A_2)$. But then $L(A_1) \not\subseteq L(A_2)$, which we needed to show.

Since the reduction we used is linear in the size of A_1 and A_2 and since checking **ioco** conformance is in PSPACE (Theorem 1), it follows that checking **ioco** conformance is PSPACE-complete.



(a) A_1 (b) A_2 (c) A'_1 (d) A'_2

Fig. 2. NFA's A_1 and A_2 represents the regular expression a and aa^* respectively. IOLTS's A'_1 and A'_2 depicts two input-enabled IOLTS's over the language $L = I \cup U$, where $I = \{i\}$ and $U = \{a\}$

Example 2. Consider automata A_1 and A_2 over $\Sigma = \{a\}$ depicted in Figure 3. Automata A_1 and A_2 accept regular languages a and aa^*, respectively. Thus,

$L(A_1) \subset L(A_2)$. Now consider the IOLTS's A'_1 and A'_2 in Figure 3 with $\{a\}$ as the output alphabets and $\{i\}$ as the set of inputs. Observe that IOLTS's A'_1 and A'_2 can be obtained from A_1 and A_2 according to the reduction algorithm presented in the proof of Theorem 2. Both models are input-enabled because they have a transition labeled with i as the only input action at every state. The set $\mathsf{Straces}(A'_2)$ is given by the regular expression $((\delta|i)^*a) \mid ((\delta|i)^*a(a|i)^*(\delta|i)^*)$. It is clear that $A'_1 \,\mathbf{ioco}\, A'_2$, because for all $\sigma \in \mathsf{Straces}(A'_2)$, $\mathsf{out}(A'_1 \ \mathsf{after}\ \sigma) \subseteq \mathsf{out}(A'_2 \ \mathsf{after}\ \sigma)$.

4 Coinductive Definition of IOCO

In the previous section, we showed that checking **ioco** is in general inefficient and requires exponential time and space (in the size of the specification). In the next section, we show that checking **ioco** can be performed in polynomial time when specifications are deterministic. To accommodate proving this result, we first show that checking **ioco** for deterministic specifications reduces to checking a simulation-like preorder which we call *coinductive **ioco*** in this section. This preorder closely resembles *alternating refinement* [4] for Interface Automata [3].

Definition 8 (Coinductive ioco). *Let deterministic IOLTS $\langle S, I, U, \rightarrow_s, \bar{s}\rangle$ be a specification, and let IOTS $\langle Q, I, U, \rightarrow, \bar{r}\rangle$ be an implementation. A binary relation $R \subseteq Q \times S$ is called a* coinductive **ioco** relation *from $\bar{r}$ to $\bar{s}$ when $(\bar{r}, \bar{s}) \in R$ and for each $(q, p) \in R$, then*

- *(Input simulation) if $p \xrightarrow{a}_s p'$, for $a \in I$, then $(q \ \mathsf{after}\ a) \neq \emptyset$ and for all $q' \in q \ \mathsf{after}\ a$, we have $(q', p') \in R$*
- *(Output simulation) if $q \overset{a}{\Rightarrow} q'$, and $a \in U \cup \{\delta\}$, then $a \in \mathsf{out}(p)$ and for all $p' \in p \ \mathsf{after}\ a$, we have $(q', p') \in R$.*

*We write $q \preceq p$, when there exists a coinductive **ioco** relation relating q to p.*

If the intent is clear from the context, we will simply say that a relation is a coinductive **ioco** relation rather than a coinductive **ioco** relation from $\bar{r}$ to $\bar{s}$.

The following theorem is the main result of this section.

Theorem 3. *For deterministic specifications, coinductive **ioco** and **ioco** coincide.*

Before giving the proof of the theorem, we need to show the correctness of the lemma given below.

Lemma 1. *Let* deterministic *IOLTS $\langle S, I, U, \bar{s}, \rightarrow_s\rangle$ be a specification, and let IOTS $\langle Q, I, U, \bar{r}, \rightarrow\rangle$ be an implementation. Let $R \subseteq S \times Q$ be a coinductive **ioco** relation, and let $\sigma \in \mathsf{Straces}(\bar{s}) \cap \mathsf{Straces}(\bar{r})$ with length $n \geq 1$. Then, $(q, p) \in R$ for all $p \in (\bar{s} \ \mathsf{after}\ \sigma)$ and $q \in (\bar{r} \ \mathsf{after}\ \sigma)$.*

Proof. Because R is a coinductive **ioco** relation, we have $(\bar{r}, \bar{s}) \in R$. We proceed with an induction on the length of σ.

- For the base case, assume that $\sigma \in L_\delta$ is a suspension trace of length 1. We distinguish two cases. Suppose that $\sigma \in I$. Following the *input simulation* condition with $(\bar{r}, \bar{s}) \in R$, we immediately find that $(q, p) \in R$ for $p \in (\bar{s}\ \mathsf{after}\ \sigma)$ and $q \in (\bar{r}\ \mathsf{after}\ \sigma)$. Suppose $\sigma \in U \cup \{\delta\}$. It follows from the *output simulation* condition together with $(\bar{r}, \bar{s}) \in R$ and the fact that $\bar{s}$ is deterministic that $(q, p) \in R$ for all $p \in (\bar{s}\ \mathsf{after}\ \sigma)$ and $q \in (\bar{r}\ \mathsf{after}\ \sigma)$. Both cases lead to the desired result.
- Assume that the induction hypothesis holds for all sequences of length $n-1$ and consider a sequence $\sigma \in \mathsf{Straces}(\bar{s}) \cap \mathsf{Straces}(\bar{r})$ with length $n \geq 2$. We may assume that σ is of the form ρa. Let $q, q' \in Q$ be arbitrary states such that $\bar{r} \stackrel{\rho}{\Rightarrow} q' \stackrel{a}{\Rightarrow} q$ (we know these exist since $\sigma \in \mathsf{Straces}(\bar{r})$). Likewise, there are unique $p, p' \in S$ such that $\bar{s} \stackrel{\rho}{\Rightarrow}_s p' \stackrel{a}{\Rightarrow}_s p$ (note that unicity follows from the fact that $\bar{s}$ is deterministic). Following the induction hypothesis, we have that $(q', p') \in R$. Therefore, the pair (q', p') satisfies the input and output simulation conditions. We distinguish two cases. Suppose that $a \in I$. Due to the *input simulation* condition, we find that $(q, p) \in R$. Suppose that $a \in U \cup \{\delta\}$; then $(q, p) \in R$ follows from the *output simulation* condition. Therefore, $(q, p) \in R$ for arbitrary $p \in (\bar{s}\ \mathsf{after}\ \sigma)$ and $q \in (\bar{r}\ \mathsf{after}\ \sigma)$.

Now, we are in the position to give the proof of Theorem 3.

Proof (Theorem 3). The proof of each implication is given separately.

- We suppose that $\bar{r} \mathrel{\not\mathbf{ioco}} \bar{s}$, then we show that there is no binary relation R such that $(\bar{r}, \bar{s}) \in R$ and R is such that the input and output simulation conditions hold for all pairs $(q, p) \in R$. By definition of **ioco**, we know that there exists a sequence $\sigma \in \mathsf{Straces}(\bar{r}) \cap \mathsf{Straces}(\bar{s})$ and there exits an output x such that $\sigma x \in \mathsf{Straces}(\bar{r})$ but $\sigma x \notin \mathsf{Straces}(\bar{s})$. We distinguish two cases; $\sigma = \epsilon$ and $\sigma \in L_\delta^+$.
  - Suppose that $\sigma = \epsilon$. Clearly, the pair $(\bar{r}, \bar{s})$ violates the output simulation condition, because $\bar{r} \stackrel{x}{\Rightarrow}$, whereas $\bar{s} \stackrel{x}{\not\Rightarrow}_s$. Therefore, there can be no relation R that simultaneously satisfies the required simulation properties *and* $(\bar{r}, \bar{s}) \in R$.
  - Suppose $\sigma \in L_\delta^+$. Towards a contradiction, assume that there is a coinductive **ioco** relation R. Thus, $(\bar{r}, \bar{s}) \in R$. It follows from $\sigma x \in \mathsf{Straces}(\bar{r})$ that there is some $q \in (\bar{r}\ \mathsf{after}\ \sigma)$ such that $x \in \mathsf{out}(q)$. Since $\bar{s}$ is deterministic, from $\sigma \in \mathsf{Straces}(\bar{s})$ we find that there is some unique $p \in S$ such that $\bar{s} \stackrel{\sigma}{\Rightarrow}_s p$. Because $\sigma x \notin \mathsf{Straces}(\bar{s})$, we find that $x \notin \mathsf{out}(p)$. From lemma 1, we obtain that $(q, p) \in R$. However, the pair (q, p) violates the output simulation condition since $q \stackrel{x}{\Rightarrow}$ but $p \stackrel{x}{\not\Rightarrow}_s$. This contradicts the assumption that there is a coinductive **ioco** relation R.
- We suppose that $\bar{r}\,\mathbf{ioco}\,\bar{s}$. We construct the relation $R = \{(\bar{r}, \bar{s})\} \cup \{(q, p) \mid \exists \sigma \in L_\delta^+ \bullet \bar{r} \stackrel{\sigma}{\Rightarrow} q \wedge \bar{s} \stackrel{\sigma}{\Rightarrow}_s p\}$. We proceed to show that R is a coinductive **ioco** relation. Clearly, $(\bar{r}, \bar{s}) \in R$. So it suffices to show that for arbitrary pair $(q, p) \in R$ the input and output simulation conditions are met. We assume arbitrary pair $(q, p) \in R$. Thus, there exists $\sigma \in L_\delta^*$ such that $\bar{r} \stackrel{\sigma}{\Rightarrow} q$

and $\bar{s} \overset{\sigma}{\Rightarrow} p$. Because $\bar{r}$ is input-enabled, q has a matching (weak) transition for any input action performed by p, *i.e.*, for all $a \in I$ such that $p \overset{a}{\Rightarrow}_s$, q after $a \neq \emptyset$. By definition of R, we find that $(q', p') \in R$ for an action $a \in I$ such that $p \overset{a}{\Rightarrow}_s p'$ and any $q \overset{a}{\Rightarrow} q'$. Thus, (q, p) satisfies the input simulation condition. Using $\bar{r}\,\mathbf{ioco}\,\bar{s}$, by construction of R, we know that $\mathsf{out}(q) \subseteq \mathsf{out}(p)$. Combining this observation with the definition of R results in $(q', p') \in R$ for all actions $a \in U \cup \{\delta\}$ for which $p \overset{a}{\Rightarrow}_s p'$ and $q \overset{a}{\Rightarrow} q'$. Thus, (q, p) also satisfies the output simulation condition. Hence, the pair (q, p) fulfills both input and output simulation conditions which was to be shown.

Since R satisfies both simulation conditions and $(\bar{r}, \bar{s}) \in R$, we find that R is a coinductive **ioco** relation.

Following Theorem 3, we say that a coinductive **ioco** relation R is a *witness* for $\bar{i}\,\mathbf{ioco}\,\bar{s}$.

Example 3. Consider the IOLTS's $\bar{s}$ and $\bar{r}$ presented in Figure 1 on page 5. We define the binary relation $R = \{(\bar{r}, \bar{s}), (r_1, s_1), (r_2, s_1), (r_3, s_2), (r_4, s_3)\}$. Clearly, for all pair of states $(q, p) \in R$, the two input and output simulation conditions presented in Theorem 3 are satisfied. Thus, the relation R is an **ioco** coinductive relation and it is also a witness for $\bar{r}\,\mathbf{ioco}\,\bar{s}$.

Now, consider the IOLTS $\bar{i}$ depicted in Figure 1. Because $\mathsf{out}(\bar{i}$ after $coin) \not\subseteq \mathsf{out}(\bar{s}$ after $coin)$, it is clearly obtained that $\bar{i}$ ~~**ioco**~~ $\bar{s}$. Therefore, we find that there is no binary relation from $\bar{i}$ to $\bar{s}$ such that $(\bar{i}, \bar{s}) \in R$ and the two input and output simulation conditions hold for any pair $(q, p) \in R$. However, we for the sake of contradiction assume that there is a relation R' such that $(\bar{i}, \bar{s}) \in R'$ and R' is such that for any $(q, p) \in R'$, the two conditions in Theorem 3 holds. Regarding the input simulation condition, $(\bar{i}, \bar{s}) \in R'$ implies that $(i_1, s_1) \in R'$ as well. We know from the properties of R', that s_1 has to simulate all the outputs produced by i_1. While observation of quiescence is not possible at s_1, via an internal transition i_1 can reach to a quiescent state. Therefore, (i_1, s_1) violates the output simulation condition which contradicts with the assumption that all pairs in R' respect the output simulation condition.

5 Conformance Checking of Deterministic Specifications

In this section, we give a polynomial-time algorithm for deciding the coinductive **ioco** relation defined in the previous section. The results obtained in the remainder of this section can be adapted in a straightforward manner to some other conformance relations in the **ioco** family, such as uioco [6]. Our algorithm is inspired by [18] and is based on the reduction of checking **ioco** into the **NHORNSAT** problem [8].

5.1 NHORNSAT Problem

The *satisfiability* problem for Boolean formulas is a typical (in fact, the first identified) NP-complete problem. In a restricted setting, however, the problem becomes decidable in polynomial time.

Definition 9 ((N)HORNSAT). *A boolean clause (a disjunction of literals) containing of at most one positive literal is called a Horn clause. We call the conjunction of Horn clauses a* Horn formula*. The satisfiability of a Horn formula is known as* ***HORNSAT****. Similarly, checking the satisfiability of a conjunction of clauses containing of at most one negative literal is called* ***NHORNSAT****.*

The *size* of a **(N)HORNSAT** instance is defined as the total number of occurrences of literals in the given formula. It is well-known that **(N)HORNSAT** is decidable in polynomial time in the size of the **(N)HORNSAT** instance [8].

5.2 Reducing IOCO to NHORNSAT

Throughout this section, we assume that we have an IOTS $\langle Q, I, U, \bar{r}, \rightarrow\rangle$ and a deterministic IOLTS $\langle S, I, U, \bar{s}, \rightarrow_s\rangle$. We assume p, p', p'' are states in S and q, q', q'' are states in Q. The algorithm, which we will present shortly, intuitively uses the following encoding:

1. positive literals X_{qp} model that q is (purportedly) related to p by a coinductive **ioco** relation,
2. negative literals $\overline{X_{qp}}$ model that the pair (p, q) cannot be in a coinductive **ioco** relation, and
3. implication clauses $X_{qp} \Rightarrow X_{q'p'}$, which are shorthand for $\overline{X_{qp}} \vee X_{q'p'}$, model that the pair (p, q) can be in a coinductive **ioco** relation only if (q', p') is in the same relation.

The reduction of checking for a coinductive **ioco** relation to **NHORNSAT** is presented in Algorithm 1: this algorithm constructs a negative Horn formula F such that F is satisfiable if and only if there exists a coinductive **ioco** relation R from $\bar{r}$ to $\bar{s}$.

The algorithm takes an implementation $\bar{r}$ and a deterministic specification $\bar{s}$ as input. We assume that for $\bar{r}$, the generalized transition relation $\Rightarrow$ from $\rightarrow$ of $\bar{r}$ has been computed. This requires a pre-processing step of $\bar{r}$, involving a transitive closure computation, see e.g. [12]. Computing $\Rightarrow$ can be done in polynomial time.

It is easy to see that the algorithm terminates. In each iteration, of the outer loop, the set $V \subseteq \{X_{qp} \mid q \in Q, p \in S\}$ strictly increases and $C \subseteq \{X_{qp} \mid q \in Q, p \in S\} \setminus V$ is a loop invariant. The algorithm thus terminates after at most $|Q| \times |S|$ iterations of the outer loop. Since the set of actions L_δ is finite, termination of the two inner loops is also guaranteed. It is equally easy to see that the formula that is constructed is a **NHORNSAT** formula.

Algorithm 1 ioco-NHORN

```
procedure ioco-NHORN($\bar{s}, \bar{r}$)
    $F \leftarrow X_{\bar{r}\bar{s}}$                                  ▷ Positive literal $X_{\bar{r}\bar{s}}$ is added to Formula $F$.
    $C \leftarrow \{X_{\bar{r}\bar{s}}\}$                              ▷ Set of unprocessed variables
    $V \leftarrow \emptyset$                                           ▷ Set of processed variables
    while $C \neq \emptyset$ do
        Choose $X_{qp} \in C$
        $V \leftarrow V \cup \{X_{qp}\}$
        $C' \leftarrow \emptyset$
        for $a \in \mathsf{init}(p) \cap I$ do
            if $(q \ \mathsf{after}\ a) \neq \emptyset$ then
                Choose $p' \in p \ \mathsf{after}\ a$                  ▷ Due to determinism, $|p \ \mathsf{after}\ a| = 1$
                $F \leftarrow F \wedge \bigwedge_{q' \in (q \ \mathsf{after}\ a)} (X_{qp} \Rightarrow X_{q'p'})$   ▷ Input simulation condition
                $C' \leftarrow C' \cup \{X_{q'p'} \mid q' \in q \ \mathsf{after}\ a\}$   ▷ Add unprocessed variables
            else
                $F \leftarrow F \wedge \overline{X_{qp}}$              ▷ Violation of input simulation
            end if
        end for

        for $a \in \mathsf{out}(q)$ do
            if $a \in \mathsf{out}(p)$ then
                Choose $p' \in p \ \mathsf{after}\ a$                  ▷ Due to determinism, $|p \ \mathsf{after}\ a| = 1$
                $F \leftarrow F \wedge \bigwedge_{q' \in (q \ \mathsf{after}\ a)} (X_{qp} \Rightarrow X_{q'p'})$   ▷ Output simulation condition
                $C' \leftarrow C' \cup \{X_{q'p'} \mid q' \in q \ \mathsf{after}\ a\}$   ▷ Add unprocessed variables
            else
                $F \leftarrow F \wedge \overline{X_{qp}}$              ▷ Violation of output simulation
            end if
        end for
        $C \leftarrow C \cup (C' \setminus V)$;
    end while

    return $F$                                                         ▷ The final negative HORN formula
end procedure
```

Example 4. Reconsider IOLTSs $\bar{s}$ and $\bar{i}$ in Figure 1 on page 5. As we concluded in Example 1, $\bar{i}\ \cancel{\textbf{ioco}}\ \bar{s}$ because, e.g. $\mathsf{out}(\bar{i} \ \mathsf{after}\ coin) \not\subseteq \mathsf{out}(\bar{s} \ \mathsf{after}\ coin)$. Therefore, the **NHORNSAT** instance obtained from Algorithm 1 must be unsatisfiable. The formula F generated by Algorithm 1 is the following:

$$
\begin{aligned}
&X_{\bar{i}\bar{s}} \wedge (X_{\bar{i}\bar{s}} \Rightarrow X_{\bar{i}\bar{s}}) \wedge (X_{\bar{i}\bar{s}} \Rightarrow X_{i_1 s_1}) \wedge (X_{\bar{i}\bar{s}} \Rightarrow X_{\bar{i}s_1}) \wedge (X_{i_1 s_1} \Rightarrow X_{i_2 s_2}) \\
&\wedge (X_{i_1 s_1} \Rightarrow X_{i_3 s_3}) \wedge \overline{X_{i_1 s_1}} \wedge \overline{X_{\bar{i}s_1}} \wedge (X_{i_2 s_2} \Rightarrow X_{i_2 s_2}) \wedge (X_{i_3 s_3} \Rightarrow X_{i_3 s_3})
\end{aligned}
$$

Indeed, it is easily seen that the obtained formula F is unsatisfiable: for F to be satisfiable, $X_{\bar{i}\bar{s}}$ must be **True**, which means that $X_{i_1 s_1}$ must be **True**, but that means that $\overline{X_{i_1 s_1}}$ is **False**.

Next, reconsider IOLTS $\bar{r}$ of Figure 1. We know from Example 1 that $\bar{r}$ **ioco** $\bar{s}$. The formula F generated by Algorithm 1 is the following:

$$X_{\bar{r}\bar{s}} \wedge (X_{\bar{r}\bar{s}} \Rightarrow X_{\bar{r}\bar{s}}) \wedge (X_{\bar{r}\bar{s}} \Rightarrow X_{r_1 s_1}) \wedge (X_{\bar{r}\bar{s}} \Rightarrow X_{r_2 s_1}) \wedge (X_{r_1 s_1} \Rightarrow X_{r_3 s_2}) \\ \wedge (X_{r_2 s_1} \Rightarrow X_{r_4 s_3}) \wedge (X_{r_3 s_2} \Rightarrow X_{r_3 s_2}) \wedge (X_{r_4 s_3} \Rightarrow X_{r_4 s_3})$$

Clearly, the constructed formula is satisfiable: assigning **True** to all literals is a satisfying assignment.

5.3 Correctness of the Reduction Algorithm

The constructed formula F by Algorithm 1 has two key properties that together ensure the correctness of our algorithm. First, the existence of a coinductive **ioco** relation R implies satisfiability of F. This follows from the observation that from any coinductive **ioco** relation R the truth assignment ν for F defined by assigning **True** to every variable X_{qp} appearing in F for which $(q, p) \in R$, and assigning **False** to all remaining variables in F is a witness to the satisfiability of F. Second, satisfiability of F implies the existence of a coinductive **ioco** relation R. In a nutshell, this follows from the observation that for any given satisfying assignment ν of F, the binary relation $R \subseteq Q \times S$ defined by $(p, q) \in R$ iff variable X_{qp} appears in F and $X_{pq} =$ **True** in ν, is a coinductive **ioco** relation. We first prove these two properties, and then state our main theorem claiming correctness of the algorithm.

Proposition 1. *Let $\langle S, I, U, \rightarrow_s, \bar{s}\rangle$ be a deterministic IOLTS and let $\langle Q, I, U, \rightarrow, \bar{r}\rangle$ be an arbitrary IOTS. Let F be the* ***NHORNSAT*** *instance from Algorithm 1. If $\bar{r} \preceq \bar{s}$ then F is satisfiable.*

The correctness of the above-given proposition results from the following lemma. This lemma essentially states that the presence of a negative literal $\overline{X_{qp}}$ in formula F indicates the pair (q, p) can never be related by a coinductive **ioco** relation.

Lemma 2. *Let F be the formula obtained from Algorithm 1, and let X_{pq} be an arbitrary variable. If F contains the literal $\overline{X_{pq}}$, then no coinductive* ***ioco*** *relation R for which $(q, p) \in R$ exists.*

Proof. Towards a contradiction, assume there is a coinductive **ioco** relation R such that $(q, p) \in R$. In our algorithm, the literal $\overline{X_{qp}}$ is only added to F under one of the following two conditions:

1. there is an input action $a \in \mathsf{init}(p)$ while q after $a = \emptyset$,
2. there is an output action $a \in \mathsf{out}(q)$ while $a \notin \mathsf{out}(p)$.

We first assume that $\overline{X_{qp}}$ is generated because of the first case, i.e., there is an input $a \in \mathsf{init}(p)$ for which q after $a = \emptyset$. Then the pair $(q, p) \in R$ does not meet the input simulation condition of Definition 8, contradicting the fact that the pair (q, p) can be in a coinductive **ioco** relation R. Next, assume that $\overline{X_{qp}}$ is generated because of the second case. Following the same line of reasoning, the presence of $(q, p) \in R$ violates the output simulation condition, contradicting that R is a coinductive **ioco** relation.

Next, we return to proving Proposition 1.

Proof (Proposition 1). Consider a coinductive **ioco** relation $R \subseteq Q \times S$. Let ν be a truth assignment for the variables in F defined as follows:

$$\nu(X_{qp}) = \begin{cases} \textbf{True} & \text{if } (q,p) \in R \\ \textbf{False} & \text{otherwise} \end{cases}$$

Since $(\bar{r}, \bar{s}) \in R$, we know that the single literal clause $X_{\bar{r}\bar{s}}$ evaluates to **True**. Next, consider the other two types of clauses that are introduced in formula F: single negative literal clauses and implication clauses.

- Clauses of the form $\overline{X_{qp}}$. Due to Lemma 2 we have $(q,p) \notin R$ whenever the negative literal clause $\overline{X_{qp}}$ is added to F in line 15 or line 25. By definition we then have $\nu(X_{qp}) = \textbf{False}$. Consequently, a negative literal clause $\overline{X_{qp}}$ in F evaluates to **True**.
- Clauses of the form $X_{qp} \Rightarrow X_{q'p'}$. We distinguish the cases when $(q,p) \notin R$ and $(q,p) \in R$.
  - Assume that $(q,p) \notin R$. By definition of ν, we have $\nu(X_{qp}) = \textbf{False}$. Then the clause $X_{qp} \Rightarrow X_{q'p'}$ immediately evaluates to **True** under ν.
  - Suppose that $(q,p) \in R$. Thus $\nu(X_{qp}) = \textbf{True}$. Therefore the clause $X_{qp} \Rightarrow X_{q'p'}$ evaluates to **True** only if $\nu(X_{q'p'}) = \textbf{True}$. The implication clause $X_{qp} \Rightarrow X_{q'p'}$ in Algorithm 1 is added to F in line 12 when there is some input $a \in \mathsf{init}(p)$ or in line 22 when there is some output $a \in \mathsf{out}(q)$ for which $q' \in q \ \mathsf{after} \ a$ and $p' \in p \ \mathsf{after} \ a$. From these observations, and the fact that R is a coinductive **ioco** relation it follows that $(q', p') \in R$. But then, by definition of ν, we have $\nu(X_{q'p'}) = \textbf{True}$, which was to be shown.

  As a result, implication clauses in F of the form $X_{qp} \Rightarrow X_{q'p'}$ evaluate to **True**.

Since there are no other types of clauses in F, formula F evaluates to **True** under ν.

The proposition below formalizes the second property of algorithm **ioco**-NHORN.

Proposition 2. *Let $\langle S, I, U, \rightarrow_s, \bar{s} \rangle$ be a deterministic IOLTS and let $\langle Q, I, U, \rightarrow, \bar{r} \rangle$ be an arbitrary IOTS. Let F be the **NHORNSAT** instance from Algorithm 1. If F is satisfiable, then $\bar{r} \preceq \bar{s}$.*

Proof. Let ν be a truth assignment such that formula F evaluates to **True**. We construct a binary relation $R \subseteq S \times Q$ as follows:

$$R = \{(q,p) \mid \text{variable } X_{qp} \text{ occurs in } F \text{ and } \nu(X_{qp}) = \textbf{True}\}$$

We proceed by showing that R is a coinductive **ioco** relation. Clearly, since the single literal $X_{\bar{r}\bar{s}}$ occurs in F and F is satisfiable, we have $\nu(X_{\bar{r}\bar{s}}) = \textbf{True}$. By definition, we then have $(\bar{r}, \bar{s}) \in R$.

Let $(q,p) \in R$ be an arbitrary pair. By definition, this means that $\nu(X_{pq}) =$ **True**. Observe that this means that formula F cannot contain the single negative literal $\overline{X_{qp}}$. We next show that the pair $(q,p) \in R$ meets both the input and output simulation conditions:

- *Ad* input simulation. Suppose that $p \xrightarrow{a}_s p'$ for some $a \in I$. Since F does not contain the negative literal $\overline{X_{qp}}$, we know that q after $a \neq \emptyset$ (line 10). Therefore, F contains implication clauses of the form $X_{qp} \Rightarrow X_{q'p'}$ where $q' \in q$ after a and $p' \in p$ after a. Since, F evaluates to **True** under ν, also $X_{qp} \Rightarrow X_{q'p'}$ evaluates to **True** under ν. Since $q' \in q$ after a is chosen arbitrarily, we find that $\nu(X_{q'p'}) =$ **True** for all $q' \in q$ after a. Then by construction, $(q', p') \in R$ for all $q' \in q$ after a.
- *Ad* output simulation. Suppose that $q \overset{a}{\Rightarrow} q'$ for some $a \in U \cup \{\delta\}$. Following the same line of reasoning as in the above case, we find that the pair (q, p) meets the output simulation condition.

An immediate consequence of the preceding two propositions is the following theorem, stating that our reduction algorithm for checking **ioco** is sound.

Theorem 4. *Let $\langle S, I, U, \rightarrow_s, \bar{s}\rangle$ be a deterministic IOLTS and let $\langle Q, I, U, \rightarrow, \bar{r}\rangle$ be an arbitrary IOTS. Let F be the* ***NHORNSAT*** *instance from Algorithm 1. Then F is satisfiable if and only if $\bar{r}$* **ioco** *$\bar{s}$.*

Proof. Following Propositions 1 and 2, we know that the formula F obtained from Algorithm 1 is satisfiable if and only if there is a coinductive **ioco** relation. Combined with Theorem 3 we find that formula F is satisfiable if and only if $\bar{r}$ **ioco** $\bar{s}$.

5.4 Complexity Analysis

We next analyze the complexity of Algorithm 1. Since **NHORNSAT** is decidable in linear time [8], proving that we can decide that a possibly nondeterministic implementation conforms to a deterministic specification in polynomial time only requires showing that the Negative Horn formula F can be constructed in polynomial time.

Theorem 5. *Let $\langle S, I, U, \rightarrow_s, \bar{s}\rangle$ be a deterministic IOLTS and let $\langle Q, I, U, \rightarrow, \bar{r}\rangle$ be an arbitrary IOTS for which $\Rightarrow$ has been computed. Algorithm 1 constructs formula F, which is of size $O(|S| \times |Q|^2 \times |L_\delta|)$ in time $O(|S| \times |Q|^2 \times |L_\delta|)$.*

Proof. To facilitate writing the proof, we first introduce some auxiliary notation. Let d_a^s denote the cardinality of the set of states reachable from a state t after executing an a-labeled transition, i.e., $d_a^t = |t$ after $a|$.

The main loop of Algorithm 1 iterates over the set of variables of the form X_{qp}, for $q \in Q$ and $p \in S$. This means there are at most $|Q| \times |S|$ iterations. The complexity of a single iteration is given by the sum of the complexity of the two inner loops (lines 9-17 and lines 19-27).

Since the size of a clause $\overline{X_{qp}}$ is smaller than the size of an implication clause introduced in line 12 or line 22, the size of the constructed clause in each iteration of one of the inner loops is bounded from above by $2 \times d_a^q$ for each $a \in I$ for the first inner loop, and $2 \times d_a^q$ for each $a \in U \cup \{\delta\}$. The cumulative size of the generated clauses in the first inner loop is therefore bounded from above by

$2 \times \sum_{a \in I} d_a^q$ and the cumulative size of the generated clauses in the second inner loop is bounded from above by $2 \times \sum_{a \in U \cup \{\delta\}} d_a^q$.

Thus, the cumulative size of the clauses added in each iteration of the outer loop is at most $2 \times (\sum_{a \in I} d_a^q + \sum_{a \in U \cup \{\delta\}} d_a^q) = 2 \times \sum_{a \in L_\delta} d_a^q$. Assuming that for all $q \in Q$ and all $p \in S$, all variables X_{qp} are inspected, the total size of the **NHORNSAT** instance is bound from above as follows:

$$\begin{array}{ll} & \sum_{p \in S} \sum_{q \in Q} (2 \times \sum_{a \in L_\delta} d_a^q) \\ \leq^\dagger & \sum_{p \in S} \sum_{q \in Q} (2 \times \sum_{a \in L_\delta} |Q|) \\ = & 2 \times |S| \times |Q|^2 \times |L_\delta| \end{array}$$

Observe that at †, we used the fact that d_a^q is bounded from above by the size of the state space of $\bar{r}$, *i.e.*, $|Q|$. Hence, the size of formula F is $O(|S| \times |Q|^2 \times |L_\delta|)$. Since we assume that $\Rightarrow$ has been computed, all operations involving $\Rightarrow$, such as $\_$ after $\_$ and out($\_$) require constant time. Constructing formula F can therefore also be done in time $O(|S| \times |Q|^2 \times |L_\delta|)$.

The theorem below states the complexity of deciding **ioco** for deterministic specifications and possibly non-deterministic implementations.

Theorem 6. *Let $\langle S, I, U, \rightarrow_s, \bar{s} \rangle$ be a deterministic IOLTS and let $\langle Q, I, U, \rightarrow, \bar{r} \rangle$ be an arbitrary IOTS. Deciding whether $\bar{r}$* **ioco** *$\bar{s}$ for deterministic specifications $\bar{s}$ and possibly non-deterministic implementations $\bar{r}$ can be done in $O(|L_\delta| \times |Q|^{2.3727}) + O(|S| \times |Q|^2 \times |L_\delta|)$.*

Proof. Generating and solving the NHORN formula F obtained from Algorithm 1 requires $O(|S| \times |Q|^2 \times |L_\delta|)$, see Theorem 6 combined with the fact that F can be solved in time linear in the size of F. A precondition to the algorithm is that $\Rightarrow$ has been computed from $\rightarrow$. Following [12], this can be done in $O(|L_\delta| \times |Q|^{2.3727})$.

When both implementation and specification are deterministic, the time complexity of our algorithm reduces to $O(|S| \times |\rightarrow|)$. Note that in this case, the computation of $\Rightarrow$ only requires augmenting the transition relation with δ transitions, which can be done in $O(|\rightarrow|)$.

6 Conclusion

In this paper, we studied the complexity of checking input-output conformance (**ioco**). We proved that the problem of checking conformance is PSPACE-complete. Then, we presented a coinductive definition of **ioco** in the restricted setting of deterministic models and through a reduction to **NHORNSAT**, presented a polynomial-time algorithm for checking **ioco** in this setting.

We plan to investigate the application of our algorithm in Section 5 to checking alternating simulation. Currently, the best known algorithm for this purpose is proposed in [7], which is a game-based algorithm for deterministic models. The solution provided in this paper may offer an alternative for the existing algorithms for checking alternating refinement relation, but it must be checked to see whether the runtime complexity of the resulting algorithm would be comparable to that of existing algorithms.

Acknowledgements. We thank Sarmen Keshishzadeh and Jeroen Keiren (both TU/e) for feedback on earlier drafts of this paper.

References

1. F. Aarts and F. W. Vaandrager. Learning I/O automata. In *Proc. of CONCUR'10*, vol. 6269 of *LNCS*, pp. 71–85. Springer, 2010.
2. P. A. Abdulla, Y.-F. Chen, L. Holík, R. Mayr, and T. Vojnar. When simulation meets antichains. In *Proc. of TACAS'10*, vol. 6015 of *LNCS*, pp. 158–174. Springer, 2010.
3. L. de Alfaro and T. A. Henzinger. Interface automata. In *Proc. of FSE/ESEC'01*, pp. 109–120. ACM, 2001.
4. R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *Proc. of CONCUR'98*, vol. 1466 of *LNCS*, pp. 163–178, 1998.
5. H. R. Asadi, R. Khosravi, M. R. Mousavi, and N. Noroozi. Towards model-based testing of electronic funds transfer systems. In *Proc. of FSEN'11*, vol. 4171 of *LNCS*, pp. 253–267. Springer, 2011.
6. M. van der Bijl, A. Rensink, and J. Tretmans. Compositional testing with ioco. In *Proc. of FATES'03*, vol. 2931 of *LNCS*, pp. 86–100. Springer, 2004.
7. K. Chatterjee, S. Chaubal, and P. Kamath. Faster algorithms for alternating refinement relations. In *Proc. of the CSL'12*, vol. 16 of *LIPIcs*, pp. 167–182. Dagstuhl, 2012.
8. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *JLAP*, 1(3):267–284, 1984.
9. L. Frantzen and J. Tretmans. Model-Based Testing of Environmental Conformance of Components. In *Proc. of FMCO'06*, vol. 4709 of *LNCS*, pp. 1–25. Springer, 2007.
10. C. Gregorio-Rodriguez, L. Llana, and R. Martinez-Torres. Input-output conformance simulation (iocos) for model based testing. In *Proc. of FMOODS'13*, vol. 7892 of *LNCS*, pp. 114–129. Springer, 2013.
11. R. M. Hierons. The complexity of asynchronous model based testing. *TCS*, 451:70–82, 2012.
12. P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990.
13. W. Mostowski, E. Poll, J. Schmaltz, J. Tretmans, and R. W. Schreur. Model-based testing of electronic passports. In *Proc. of FMICS'09*, vol. 5825 of *LNCS*, pp. 207–209. Springer, 2009.
14. G. J. Myers, T. Badgett, and C. Sandler. *The Art of Software Testing (3rd Ed.)*. Wiley, 2011.
15. N. Noroozi, M. R. Mousavi and T. A. C. Willemse. Decomposability in Input Output Conformance Testing. In *Proc. of MBT'13*, vol. 111 of *EPTCS*, pp. 51–66., 2013.
16. Bas Ploeger. *Improved Verification Methods for Concurrent Systems*. PhD thesis, TU/Eindhoven, 2009.
17. A. Pretschner. One evaluation of model-based testing and its automation. In *Proc. of ICSE'05*, pp. 722–723. ACM, 2005.

18. S. K. Shukla, H. B. Hunt III, D. J. Rosenkrantz, and R. E. Stearns. *On the Complexity of Relational Problems for Finite State Processes (Extended Abstract)*, In *Proc. of ICALP'96*, vol. 1099 of *LNCS*, pp. 466–477, Springer, 1996.
19. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proc. STOC'73*, pp. 1–9. ACM, 1973.
20. K. Thompson. Regular expression search algorithms. *CACM*, 11(6):419–422, 1968.
21. J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, vol. 4949 of *LNCS*, pp. 1–38. Springer, 2008.
22. M. Veanes and N. Bjørner. Alternating simulation and ioco. *STTT*, 14(4):387–405, 2012.
23. M. Veanes, C. Campbell, W. Grieskamp, W. Schulte, N. Tillmann, and L. Nachmanson. Model-based testing of object-oriented reactive systems with Spec Explorer. In *Formal Methods and Testing*, vol. 4949 of *LNCS*, pp. 39–76. Springer, 2008.
24. V. Vishal, M. Kovacioglu, R. Kherazi, and M.R. Mousavi. Integrating model-based and constraint-based testing using SpecExplorer. In *Proc. MoTiP'12*, pp. 219–224. IEEE, 2012.