Approximate Active Learning of Nondeterministic Input Output Transition Systems*

Michele Volpato[1] and Jan Tretmans[12]

[1] Institute for Computing and Information Sciences
Radboud Universiteit Nijmegen, Nijmegen, The Netherlands
{m.volpato, tretmans}@cs.ru.nl
[2] TNO - Embedded Systems Innovation
Eindhoven, The Netherlands

Abstract. Constructing a model of a system for model-based testing, simulation, or model checking can be cumbersome for existing, third party, or legacy components. Active automata learning, a form of black-box reverse engineering, and in particular Angluin's L* algorithm, support the automatic inference of a model from a System Under Learning (SUL), through observations and tests. Most of the algorithms based on L*, however, deal with complete learning of deterministic models, thus being unable to cope with nondeterministic SULs, and always learning a complete and correct model as they are based on equivalence between the SUL and the model. We present an adaptation of Angluin's algorithm for active learning of nondeterministic, input-enabled, input-output transition systems. It enables dealing with nondeterministic SULs, and it allows to construct partial, or approximate models, by expressing the relation between the SUL and the learned model as a refinement relation, not necessarily an equivalence. Thus, we can reason about learned models being more, or less precise than others. Approximate learning has benefits in model-based regression testing: we need not to wait until a complete model has been learned; with an approximate model **ioco**-based regression testing can start.

1 Introduction

Model-based testing, model-driven design, model simulation, model checking: once you have a model of the behaviour of a software component, all kinds of analyses can be performed contributing to the construction of better software in less time. A key problem, however, is the initial construction of a model, in particular for existing, third party, or legacy components, for which no or only limited documentation is available.
Active automata learning helps with automatically inferring a state-based model from the behaviour of a System Under Learning (SUL) by

* This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs.

observing and testing that behaviour, i.e., through black-box reverse engineering. Automata learning started with the $L^\star$ algorithm by Angluin [Ang87], after which several variations and improvements were made, among others, for Mealy machines [RS93] [Nie03] for learning models of reactive systems such as controllers and network protocols. Most of them, however, concern learning of deterministic systems, which means that (i) they cannot deal with nondeterministically behaving systems, and, (ii) the correctness of a learned model is based on an isomorphism with the SUL. In particular the latter implies that a learned model is either complete and correct, or not correct at all.

We present an adaptation of $L^\star$ for active learning of nondeterministic, input-enabled, input-output transition systems [Tre96]. Nondeterminism allows to reason about partially correct models, using relations that are not equivalences, e.g., pre-orders. This enables reasoning of learned models being better than another learned model, and thus to approximate the model of the SUL. In particular we want to (i) avoid the equivalence checking step used in $L^\star$, which is impossible to implement in practice, given non-exhaustiveness of testing, and, (ii) define a relation between a partially learned model and the SUL, as an invariant through the learning process.

The notion of refinement, for the approximate models, is based on the **ioco**-theory for model-based testing [Tre96], which defines a framework comprehending a precise notion of conformance in terms of implementation relations that can deal with nondeterminism. The learning approach is based on $L^\star$, with corresponding observation table. At any moment during the learning process an under- and an over-approximation model is constructed for which we prove a refinement-relationship to the SUL. After any step the algorithm can be stopped with a correct, but perhaps not precise enough model. Continuation will lead to gradually better models, until the model is considered precise enough. The learning algorithm uses properties of input-output labelled transition systems for optimization. Moreover, it must be assumed that, due to nondeterminism, there exists an oracle that knows when all nondeterministic outputs have been observed. Without this assumption the over-approximation can only be the most imprecise, yet correct model, i.e., *chaos*. An implementation of such an oracle depends on the system under learning: does observing an output x exclude the possibility to observe the output y after the same input? Does repeating the same sequence of inputs k times produce all observable outputs after that sequence?

Approximate learning can have big advantages in regression testing, for example when testing the complete model is beyond feasibility: because already with a partial model, model-based **ioco**-regression testing can start.

Related Work This paper improves the results presented in our previous work [VT14] by weakening two assumptions on which the learning of nondeterministic systems was based.

First, we assumed in [VT14] that we can somehow obtain all nondeterministic outputs in a particular state at once (i.e., the set $out(\text{SUL } \textbf{after } \sigma)$; cf. Section 2). In this paper, more realistically, we observe all outputs in-

dividually, but we assume the presence of an oracle that knows when all nondeterministic outputs have been observed (e.g., by repeating observations often enough). The difference with [VT14] is that we are now able to deal with incomplete observations, i.e., we can still construct a model even if not all outputs have been observed.
Secondly, we assumed in [VT14] the ability to exhaustively check the equivalence of a learned model w.r.t. the SUL (called equivalence exhaustiveness, or equivalence oracle in [Ang87]). This assumption is completely dropped, implying that we can never be sure anymore to have learned the final, complete model. The assumption is replaced by the property that at any moment during the learning process we can always construct a correct approximation of the complete model, in the sense of having an **ioco**-like relation between the currently learned model and the (unknown) complete model.
Active learning of *observable nondeterministic finite state machines* (ONFSMs) [Ea10] [Pa13], attempts to learn a deterministic model of systems behaving as a nondeterministic Mealy machine. Such a behaviour is comparable to a labelled transition system with alternation between inputs and outputs. The work in [MS11] applies *learning-based testing* to reactive systems by combining *incremental learning* algorithms with formal requirements specified in *temporal logic*. The idea of incremental learning is similar to the approximate learning and it is a good candidate to be considered for future work.

2 Preliminaries

Labelled transition systems and ioco relation [Tre96] A labelled transition system is a 5-tuple $\langle Q, L_I, L_U, \rightarrow, q_0 \rangle$, where Q is a set of states, L_I and L_U are two disjoint sets of inputs and outputs (labels), respectively, $\rightarrow$ is the transition relation and $q_0 \in Q$ is the initial state. We may refer at a labelled transition system by using its initial state. We use $q \xrightarrow{\lambda} q'$ for $(q, \lambda, q') \in \rightarrow$ and we say that q enables λ. We shorten $L_I \cup L_U$ by L. A special label τ is used for internal, unobservable transitions. Let q, q' be states and ϵ be the empty sequence, we define $q \stackrel{\epsilon}{\Rightarrow} q' \iff q = q'$ or $q \xrightarrow{\tau} \ldots \xrightarrow{\tau} q'$. Given a label λ, we define $q \stackrel{\lambda}{\Rightarrow} q'$ as $q \stackrel{\epsilon}{\Rightarrow} p \xrightarrow{\lambda} p' \stackrel{\epsilon}{\Rightarrow} q'$ and extend $\Rightarrow$ for sequences of labels in the usual way. The set $traces(q) = \{\sigma \in L^* \mid \exists q' : q \stackrel{\sigma}{\Rightarrow} q'\}$ indicates the *enabled* traces from a state q. We denote the set of states reachable from q via a trace σ as $(q \textbf{ after } \sigma) = \{q' \mid q \stackrel{\sigma}{\Rightarrow} q'\}$. We say that such states are reached by *running* σ from q. A system is deterministic iff $|(q \textbf{ after } \sigma)| \leq 1$. We write $\sigma_1 \cdot \sigma_2$ or just $\sigma_1\sigma_2$ to denote the concatenation of sequences σ_1 and σ_2. We extend this notation to sets of sequences in the usual way.
A state q is called quiescent if $\forall \lambda \in L_U \cup \{\tau\} : q \stackrel{\lambda}{\not\rightarrow} q'$ for all $q' \in Q$. Let $\delta \notin L_I \cup L_U$, L_δ is defined as $L \cup \{\delta\}$ and $\langle Q, L_I, L_U \cup \{\delta\}, \rightarrow_\delta, q_0 \rangle$ is the labelled transition system $\langle Q, L_I, L_U, \rightarrow, q_0 \rangle$ to which transitions we add (q, δ, q) for all quiescent states q. We identify δ as *quiescence* and we sometimes include it in the outputs. The set of suspension traces is $Straces(q) = \{\sigma \in L_\delta^* \mid \exists q' \ . \ q \stackrel{\sigma}{\Rightarrow} q'\}$. The set of

outputs, including quiescence, that are enabled in a set of states P is $out(P) = \{\lambda \in L_U \cup \{\delta\} \mid \exists q \in P, q' \in Q : q \stackrel{\lambda}{\Rightarrow} q'\}$. In an *input-enabled labelled transition system*, also called *input-output transition system*, all inputs are enabled in every state. Given a set $F \subseteq L_\delta^*$, an input-output transition system i and a labelled transition system s, we define: i $\mathbf{ioco}_F$ $s \iff \forall \sigma \in F : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$ and i **ioco** $s \iff i$ $\mathbf{ioco}_{Straces(s)}$ s. The **ioco** relation is used for testing the conformance of an implementation w.r.t. a given specification.

Active Learning of Regular Languages Angluin's L* [Ang87] is a well known, efficient algorithm that, given an alphabet L, infers a Deterministic Finite Automaton (DFA) for a regular language over L. First some *membership* queries, in the form "Is this word in the target language?", are asked in order to construct an hypothesis DFA. Then an oracle replies to an *equivalence* query affirmatively if the hypothesis is equivalent to the system under learning (SUL), otherwise it provides a counterexample that can be used to improve the hypothesis, by asking more membership queries, and the process is repeated. The central structure of the L* algorithm is the *observation table*. The observation table is a triple (S, E, T), where S is a prefix-closed set of traces that represent *access sequences* to states of the learned DFA and E is a suffix-closed set of traces that represent *distinguishing sequences* of the states in the learned DFA. The function T maps traces in $((S \cup S \cdot L) \cdot E)$ to either `TRUE`, if that trace is accepted by the language, or `FALSE`, otherwise. In T both access sequences and their one letter extensions are considered, because this information is needed in order to construct a valid DFA. If two traces in $(S \cup S \cdot L)$ are mapped to the same boolean value by T for each suffix in E, then they represent the same state. (S, E, T) is called observation *table* because one can depict the elements of $(S \cup S \cdot L)$ as rows and the elements of E as columns. Each entry $s \cdot e$ in the table is given by the result of $T(s \cdot e)$. We do not give precise definitions for active learning of DFAs. However, in the next section, we will provide detailed definitions for active learning of input-output transition systems.

3 Manipulating the Observation Table

In this paper we aim to learn a nondeterministic input-output transition system by using the L* approach, with some differences: we do not alternate membership and equivalence queries, because we introduce the concept of preciseness that will guide to the termination of the learning process. Furthermore, we assume that the SUL behaves as a (*nondeterministic*) input-output transition system, and for this reason, in the observation table, we store sets of outputs instead of boolean values. We will present our notions of closedness and consistency, which are similar to the ones used in Angluin's L*. In the rest of the paper we use SUL for both the system under learning and the labelled transition system that represents it.

3.1 Nondeterministic Observation Table

We define a *nondeterministic observation table*, from now on simply *observation table*, as a triple (S, E, T) where S and E are non-empty, finite sets of traces over L_δ, prefix-closed and suffix-closed, respectively, and T is a function that maps traces in $((S \cup S \cdot L_\delta) \cdot E)$ to a subset of $(L_U \cup \{\delta\})$. We often use s for elements of $(S \cup S \cdot L_\delta)$ and e for elements of E. A matrix view of the table has the prefixes, i. e., elements of $(S \cup S \cdot L_\delta)$, as row labels, and suffixes, i. e., elements of E as column labels. Each entry contains the observed outputs after running the related prefix followed by the related suffix on the SUL. An image of T is a set of outputs due to the nondeterministic behaviour of the SUL. Furthermore, at any point in the process of learning, there might be some entries in the observation table whose content is not *completely* known and others for which we are sure that all the enabled outputs, for that specific trace, have been observed. In the latter case we *mark* the entry as *complete*, more precisely, if $T(s \cdot e)$ is marked as complete, then $T(s \cdot e) = \mathit{out}(\text{SUL } \textbf{after } s \cdot e)$. If an entry $T(s \cdot e)$ is *incomplete*, i. e., it is not marked as complete, then $T(s \cdot e) \subseteq \mathit{out}(\text{SUL } \textbf{after } s \cdot e)$.
In the learning process, the table is modified repeatedly, adding rows or columns and changing the content of entries. New entries are not marked as complete and T maps them to the empty set. Given an observation table (S, E, T) and a trace $s \in (S \cup S \cdot L_\delta)$, $\mathit{row}(s)$ denotes the function from E to $2^{(L_U \cup \{\delta\})}$ defined by $\mathit{row}(s)(e) = T(s \cdot e)$. Given a trace $s \in S$, if an output λ is not in $T(s \cdot \epsilon)$, because it is either not enabled or not observed yet, then $\mathit{row}(s \cdot \lambda)$ is not defined.
In classic L*, for each entry of the table, it is necessary to ask a membership query in order to fill that entry. While learning an input-output transition system, some entries can be filled automatically, and some others are not needed at all, because of some properties of labelled transition systems. We will present such special cases in the next section.

3.2 Filling the Observation Table

In order to be able to infer a valid input-output transition system from an observation table, also called *hypothesis*, we need to fill the table. Asking an *output query*, the analogue of a membership query in Section 2, consists of obtaining an output that is enabled after running a trace σ on the SUL: $\texttt{output}(\sigma)$ gives an output x (including quiescence) such that $x \in \mathit{out}(\text{SUL } \textbf{after } \sigma)$.
For deterministic systems, replying to this kind of questions is easy. If a trace is enabled from the initial state, then the output obtained from the output query on that trace is the only possible output. An implementation of the output query for deterministic systems just needs to run the sequence, waiting for all the outputs during the process. For nondeterministic systems, on the contrary, given a trace σ, processing $\texttt{output}(\sigma)$ is not so trivial. We are not sure that we will observe precisely the outputs that are contained in σ while running it. If we assume that there exists a maximum number k such that, after running a trace k times, we are sure of having observed all the possible outputs after that trace, then

the output query can be implemented. It's out of scope of this paper to give an implementation of the output query. We assume that such an implementation exists.
We assume also the existence of another type of queries: the *completeness queries*. Given a sequence of labels and a set of outputs (the observed outputs so far, after that sequence), if the completeness query replies affirmatively then the set of outputs defines exactly the outputs that are enabled after running that sequence on the SUL. As for the output query, we do not provide an implementation of the completeness query. An approach for implementing it, similarly to [Pa13], assuming that after a given number of output queries on the same sequence, we are sure to have observed all the possible outputs, is testing if that many output queries have already been asked. We couple output and completeness queries in order to fill the observation table.
When an output query is answered, there are more entries in the table that can be updated with the result of that query. The result of $\texttt{output}(\sigma_1 \cdot \delta \cdot \sigma_2)$ must also be a possible outcome of $\texttt{output}(\sigma_1 \cdot \sigma_2)$, given that, in a labelled transition system, quiescence is always modelled as a self loop. For this reason we define $\delta^*(\sigma)$ as the smallest set s.t. $\sigma \in \delta^*(\sigma)$ and $\sigma_1 \cdot \delta \cdot \sigma_2 \in \delta^*(\sigma) \Rightarrow \sigma_1 \cdot \sigma_2 \in \delta^*(\sigma)$. We use `update(S,E,T)` for populating the observation table. In Algorithm 1, for each incomplete entry

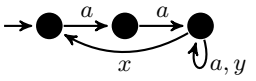
Algorithm 1 `update(S,E,T)`

```
1:  for each s ∈ (S ∪ S·L_δ), e ∈ E s.t. row(s) is defined do
2:    if T(s·e) is marked as complete then
3:      continue;
4:    if s·e ends with δ then
5:      T(s·e) ← {δ}
6:      mark T(s·e) as complete
7:    else
8:      if s = s'·δ ∧ T(s'·ϵ) = {δ} ∧ T(s'·ϵ) is complete then
9:        T(s·e) ← T(s'·e)
10:     if T(s'·e) is complete then
11:       mark T(s·e) as complete
12:   else
13:     out ← output(s·e)
14:     for each s' ∈ (S ∪ S·L_δ), e' ∈ E s.t. s'·e' ∈ δ*(s·e) do
15:       T(s'·e') ← T(s'·e') ∪ {out}
16:     if isComplete(s·e) then
17:       for each s' ∈ (S∪S·L_δ), e' ∈ E s.t. s'·e' = s·e do
18:         mark T(s'·e') as complete
```

$T(s \cdot e)$ of the observation table, if $s \cdot e$ ends with δ only quiescence, or some inputs, can be enabled, thus we set $T(s \cdot e)$ to $\{\delta\}$ and mark it as complete. If s ends with δ and the state reachable by the longest proper prefix of s enables only δ, then there will be a δ self loop in that state, thus $row(s \cdot \delta) = row(s)$, Lines 8 to 11. If none of the previous special cases is met, then we ask an output query for the trace $s \cdot e$ and update each entry identified by $s \cdot e$. Afterwards, a completeness query is asked, and the entries are updated accordingly.

Example 1. Figure 1b gives an observation table for the system under learning of Figure 1a. This is not the only possible observation table;

to a different nondeterministic reply to an output or a completeness query corresponds a different table. We will show later that the learning process starts with $S = E = \{\epsilon\}$, thus $S{\cdot}L_\delta = \{a, x, y, \delta\}$. Accordingly to Section 3.1, $row(x)$ and $row(y)$ are not defined, thus we do not add them to the table. While executing Algorithm 1, none of the entries is complete. The output queries will reply with the only possible output for each entry: $\texttt{output}(\epsilon) = \delta$ and $\texttt{output}(a) = \delta$. Let's say that the completeness queries for ϵ and a result in `FALSE`, then we do not mark those entries as complete. Because of Line 5 we know that δ will be added in the third entry and the entry will be marked as complete.



(a) An iots q, $L_I = \{a\}$ and $L_U = \{x, y\}$

		ϵ
S	ϵ	$\langle\{\delta\}, \bot\rangle$
$S{\cdot}L_\delta$	a	$\langle\{\delta\}, \bot\rangle$
	δ	$\langle\{\delta\}, \top\rangle$

(b) A possible first observation table for q.

		ϵ
S	ϵ	$\langle\{\delta\}, \bot\rangle$
	δ	$\langle\{\delta\}, \top\rangle$
$S{\cdot}L_\delta$	a	$\langle\{\delta\}, \bot\rangle$
	δa	$\langle\{\delta\}, \bot\rangle$
	$\delta\delta$	$\langle\{\delta\}, \top\rangle$

(c) The observation table after it has been stabilized.

Fig. 1. Running example used in this paper. `FALSE` is replaced by $\bot$ and `TRUE` by $\top$.

If $\texttt{isComplete}(s{\cdot}e)$ does not reply affirmatively, then we do not know if we have observed all the outputs enabled after $s{\cdot}e$. This uncertainty can be expressed in the current hypothesis induced by the learning algorithm in either a restrictive way, considering the set as complete, or in a permissive way, considering all outputs, including quiescence, as a possible outcome for $s{\cdot}e$. These two different approaches lead us to construct two different hypotheses: the restrictive $\mathcal{H}^-$ and the permissive $\mathcal{H}^+$. The hypothesis $\mathcal{H}^-$ is constructed in a way similar to the hypothesis in [VT14], where this uncertainty was not considered. In order to build the other hypothesis, $\mathcal{H}^+$, we need to give a formal notion of what its states are, and how to handle the uncertain transitions. For this reason, we define a function row^+, similar to row: given a trace $s \in (S \cup S{\cdot}L_\delta)$, row^+ denotes the function from E to $(2^{(L_U \cup \{\delta\})}, \{\texttt{TRUE}, \texttt{FALSE}\})$ defined by $row^+(s)(e) = (T(s{\cdot}e), \texttt{isComplete}(s{\cdot}e))$. Given two prefixes $s_1, s_2 \in (S \cup S{\cdot}L_\delta)$ and two suffixes $e_1, e_2 \in E$ we define the equivalence over marked entries of the table as $T(s_1{\cdot}e_1) =^+ T(s_2{\cdot}e_2)$ if and only if $T(s_1{\cdot}e_1) = T(s_2{\cdot}e_2)$ and $\texttt{isComplete}(s_1{\cdot}e_1) = \texttt{isComplete}(s_2{\cdot}e_2)$. As for the function row, given $s \in S$, if an output $\lambda \notin T(s{\cdot}\epsilon)$, $row^+(s{\cdot}\lambda)$ is not defined. If $row(s)$ is not defined, also $row^+(s)$ is not, and vice versa. The two hypotheses $\mathcal{H}^-$ and $\mathcal{H}^+$ are suspension automata [Tre96], label-deterministic versions of some labelled transition systems, where quiescence has been made explicit. Ideally, we want to stop the learning process when either the set of traces of $\mathcal{H}^-$ or the set of traces of $\mathcal{H}^+$ is equivalent to the set of suspension traces of SUL. Given the nondeterministic nature of labelled transition systems, we are satisfied when

we find a *good* approximation of SUL. We will elaborate more on this concept in Section 5.

3.3 Global closedness and consistency

In the classic L* algorithm, two properties of the observation table are necessary in order to construct an hypothesis: closedness and consistency. In this paper we derive two different hypotheses $\mathcal{H}^-$ and $\mathcal{H}^+$, and for this reason, we need to define similar properties for constructing both of them. *Global closedness*, Definition 1, extends the notion of closedness introduced in [VT14] to rows with cells that are not marked as complete. A globally closed observation table must be closed on both row and row^+ functions. Closedness on row^+ implies closedness on row, given the definition of row^+, thus it is enough to check only the most specific one.

Definition 1. *An observation table* (S, E, T) *is* globally closed *if* $\forall s' \in (S{\cdot}L_\delta)$ *s.t.* $row^+(s')$ *is defined:* $\exists s \in S$ *such that* $row^+(s') = row^+(s)$

Note that prefixes whose function row is not defined are not taken into consideration for global closedness. The same is true for global consistency. Two elements in S that are mapped in the same way by row (resp. row^+), represent the same state in $\mathcal{H}^-$ (resp. $\mathcal{H}^+$). Thus their one label extensions, in $S{\cdot}L_\delta$, must also represent the same state.

Definition 2. *An observation table* (S, E, T) *is* globally consistent *if*
$\forall s_1, s_2 \in S$*:*
$row(s_1) = row(s_2) \Rightarrow \forall \lambda \in L_\delta \; . \; row(s_1\lambda) = row(s_2\lambda)$ *AND*
$row^+(s_1) = row^+(s_2) \Rightarrow \forall \lambda \in L_\delta \; . \; row^+(s_1\lambda) = row^+(s_2\lambda)$

A globally closed and consistent observation table is called *stable*. Algorithm 2 stabilizes a given observation table. It checks for global closedness and global consistency. If any of the two properties is not valid, the algorithm takes the same actions that are taken for classic closedness and consistency, i. e., either adding some elements to S from $S{\cdot}L_\delta$, or

Algorithm 2 Stabilize observation table

```
1: while not globally closed or consistent do
2:   if not globally closed then
3:     pick an s' ∈ (S·L_δ) such that ∀s ∈ S,
       row⁺(s') ≠ row⁺(s) ∧ row⁺(s') is defined
4:     S ← S ∪ {s'}
5:     update(S, E, T)
6:   if not globally consistent then
7:     pick λ ∈ L_δ and e ∈ E such that the suffix λ·e is inconsistent with row or row⁺
8:     E ← E ∪ {λ·e}
9:     update(S, E, T)
10: return the stable observation table (S, E, T)
```

adding one or more elements to E. After each step the table is updated again, because new entries are created. Note that only prefixes whose row function is defined are added to S.

Example 2. Let us consider the observation table of Figure 1. It is not globally closed, because $row^+(\delta)$ does not have a representative in S. Thus we add δ to S. Once quiescence has been observed, no other output can be observed again, unless an input is provided. Thus $row(\delta x)$ can never be defined. After having updated the table, a possible result, due to nondeterminism, is the table of Figure 1c. Now the observation table is both globally closed and globally consistent.

Quiescence Reducibility The explicit representation of quiescence in suspension automata introduces some properties that must be satisfied by any *valid* suspension automaton to be suspension-trace equivalent to a labelled transition system. In [Wil07] four properties are identified and in [VT14] it is proven that three of them are always satisfied by the transition systems constructed from any observation table. Even though $\mathcal{H}^-$ and $\mathcal{H}^+$ are new constructions, the main points for the proofs in [VT14] are still valid. The fourth one, *quiescence reducibility*, needs to be checked on the table before the construction of the two hypotheses, after it has been stabilized. It results in adding some suffixes to E preserving the observations made so far. We refer the reader to [VT14] for an algorithm that ensures quiescence reducibility.

4 Construction of Hypotheses

If T maps a trace $s{\cdot}e$ to an empty set, then we never observed any output after running $s{\cdot}e$. This can happen because $s{\cdot}e$ contains rare outputs. Rows with an empty set in the first column represent states from which the output behaviour is not known. Empty entries in the table are, by definition, incomplete. During the construction of hypothesis $\mathcal{H}^+$, they are handled as *full* entries, where the entire set of outputs, including quiescence, is enabled. For constructing $\mathcal{H}^-$, on the contrary, such entries remain empty and if we would use the same hypothesis construction algorithm of [VT14] we would obtain some states with no output transitions. A labelled transition system, and in particular a suspension automaton, must be *non-blocking*, i. e., it must be possible to observe an output, or quiescence, in every state. Thus a naive construction of $\mathcal{H}^-$ would produce a non valid suspension automaton. For this reason we consider this kind of unknown behaviour as quiescence, and we add a δ-transition to any state which would have no output transitions. If our guess is proven wrong in a future query, then that query will also provide a valid output to add to the empty entry, allowing us to proceed with a "more" correct model. Otherwise, either our guess was correct, or that part of the system was not easily reachable and it is not possible to derive a better model of it.
The construction of $\mathcal{H}^-$ is given by Algorithm 3. It is similar to the one used in [VT14], the only difference is a quiescent state that collects δ-transitions from states with unknown output behaviour. In order

Algorithm 3 Construct $\mathcal{H}^-$ from a stable and quiescent reducible (S, E, T)

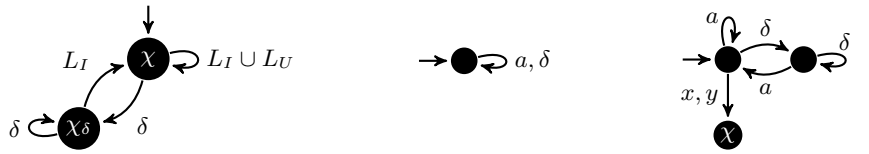
1: $Q \leftarrow \{row(s) \mid s \in S\}$
2: $Q \leftarrow Q \cup \{\Delta\}$
3: add $\Delta \xrightarrow{\delta} \Delta$ {Δ *is a quiescent state*}
4: $q_0 \leftarrow row(\epsilon)$
5: **for each** $row(s) \in Q$ **do**
6: **for each** $\lambda \in L_I$ **do**
7: add $row(s) \xrightarrow{\lambda} row(s{\cdot}\lambda)$
8: **if** $T(s{\cdot}\epsilon) \neq \emptyset$ **then**
9: **for each** $\lambda \in T(s{\cdot}\epsilon)$ **do**
10: add $row(s) \xrightarrow{\lambda} row(s{\cdot}\lambda)$
11: **else**
12: add $row(s) \xrightarrow{\delta} row(\Delta)$ {Non-blocking}

to construct $\mathcal{H}^-$, Algorithm 3 first creates a state for each row in the top part of the table and the quiescent state Δ. Then it adds, for each state and for each input label, a transition from that state to the state identified by the *row* function. Finally, for each output, a transition is added only if that output is enabled in that state. If the state enables no outputs, it adds a δ-transition from that state to Δ.

Chaotic behaviour In $\mathcal{H}^+$, an incomplete entry $T(s{\cdot}e)$ is treated as a sort of *full* entry, in which all the outputs that are not yet observed *might* be observed in future queries. However, the observation table is not able to specify any behaviour for such outputs from a state $row^+(s)$ if $T(s{\cdot}\epsilon)$ is not complete, because $row^+(s{\cdot}\lambda)$, where λ is an unobserved output, is not defined. For this reason, during the construction of $\mathcal{H}^+$, we keep a non-conservative approach and we allow any behaviour, i. e., we add $(row^+(s), \lambda, p)$ to $\rightarrow$, where λ is an output never observed after the query $s \in S$, and p is a chaotic state [BRT04]. Such a chaotic state can be defined in many ways. Figure 2a shows a representation of a chaotic state. The construction of $\mathcal{H}^+$ starts with χ and χ_δ as states. Then more states are added according to row^+. Output transitions whose behaviour is unknown, i. e., whose target state is unknown, will target χ. Transitions labelled with δ whose behaviour is unknown will target χ_δ.

Algorithm 4 Construct $\mathcal{H}^+$ from a stable and quiescent reducible (S, E, T)

1: $Q \leftarrow \{row^+(s) \mid s \in S\}$
2: $Q \leftarrow Q \cup \{\chi, \chi_\delta\}$
3: $q_0 \leftarrow row^+(\epsilon)$
4: add $\chi \xrightarrow{L_U} \chi$,$\chi \xrightarrow{\delta} \chi_\delta$ and $\chi_\delta \xrightarrow{\delta} \chi_\delta$
5: **for each** $row^+(s) \in Q$ **do**
6: **for each** $\lambda \in L_I$ **do**
7: add $row^+(s) \xrightarrow{\lambda} row^+(s{\cdot}\lambda)$
8: **for each** $\lambda \in (L_U \cup \{\delta\})$ **do**
9: **if** $\lambda \in T(s, \epsilon)$ **then**
10: add $row^+(s) \xrightarrow{\lambda} row^+(s{\cdot}\lambda)$
11: **else if** $T(s, \epsilon)$ is not complete **then**
12: **if** $\lambda = \delta$ **then**
13: add $row^+(s) \xrightarrow{\delta} \chi_\delta$
14: **else**
15: add $row^+(s) \xrightarrow{\lambda} \chi$



(a) Graphical representation of *chaotic behaviour*.

(b) The hypothesis $\mathcal{H}^-$ from Figure 1c.

(c) The hypothesis $\mathcal{H}^+$ from Figure 1c.

Fig. 2. Chaotic behaviour and the two hypotheses constructed from Figure 1c.

Example 3. The table of Figure 1c is stable and quiescence reducible. We can construct $\mathcal{H}^-$ and $\mathcal{H}^+$.
The construction of $\mathcal{H}^-$ is easy. There is only one state, with output δ (Δ is not shown in the figure because it is not reachable). All transitions are self loops: Figure 2b. The construction of $\mathcal{H}^+$ is slightly more complex. First we add the chaotic state (χ_δ is not drawn in Figure 2c). Then we create two more states, one for $row^+(\epsilon)$ and the other for $row^+(\delta)$, and finally, we add the transitions to the relevant states. From the state identified by $row^+(\epsilon)$, we need to reach the chaotic state for each output that is not in the entry $T(\epsilon{\cdot}\epsilon)$. This action is not necessary for the state represented by $row^+(\delta)$ because $T(\delta{\cdot}\epsilon)$ is complete.

Theorem 1. *Let (S, E, T) be a globally closed, globally consistent and quiescent reducible observation table, and let $\mathcal{H}^-$ and $\mathcal{H}^+$ be the hypotheses obtained with Algorithm 3 and Algorithm 4, respectively, then: $\mathcal{H}^-$ **ioco** $\mathcal{H}^+$.*

4.1 Conformance Relation on Observed Behaviour

We provided the construction of two models, based on the observation table, that represent the current information we have been able to infer from observing the behaviour of the SUL. How can we formally describe how they are related to the SUL? We need to introduce a new conformance relation based on that behaviour.
The fact that $\mathcal{H}^-$ and $\mathcal{H}^+$ are constructed from a limited amount of information, i. e., the replies to some output queries, offers the idea of reasoning in terms of relations restricted to that information only. For this reason we define $\mathbf{ioco}_{(S,E,T)}$ in Definition 3. Informally, a labelled transition system i is $\mathbf{ioco}_{(S,E,T)}$ conforming to another labelled transition system i' if and only if, for all the traces that can be constructed from the table (S, E, T) by concatenating any prefix with any suffix, the set of outputs observable after executing those traces on i can also be observed after executing the same trace on i'.

Definition 3. *Let (S, E, T) be an observation table, i and i' two labelled transition systems, then: i $\mathbf{ioco}_{(S,E,T)} i' \iff \forall \sigma = s{\cdot}e$ such that $s \in (S \cup S{\cdot}L_\delta), e \in E \wedge row(s)$ is defined $\wedge T(s{\cdot}e) \neq \emptyset$: $out(i$ **after** $\sigma) \subseteq out(i'$ **after** $\sigma)$*

The system under learning and the two hypotheses are related accordingly to $\mathbf{ioco}_{(S,E,T)}$.

Theorem 2. *Let (S, E, T) be a globally closed and consistent observation table obtained from* SUL*, and $\mathcal{H}^-$ and $\mathcal{H}^+$ be the suspension automata constructed using Algorithm 3 and Algorithm 4 respectively. Then $\mathcal{H}^-$ $\boldsymbol{ioco}_{(S,E,T)}$ SUL and* SUL $\boldsymbol{ioco}_{(S,E,T)}$ $\mathcal{H}^+$.

Proof (Proof idea.). First note that $T(s{\cdot}e) \subseteq out(\text{SUL } \mathbf{after}\ s{\cdot}e)$. Then the theorem can be proven by showing that $\mathcal{H}^-$ and $\mathcal{H}^+$ are consistent with the observation table.

Note that, fixing (S, E, T), the relation $\mathbf{ioco}_{(S,E,T)}$ is transitive, thus $\mathcal{H}^-$ $\mathbf{ioco}_{(S,E,T)}$ $\mathcal{H}^+$. Furthermore, SUL $\mathbf{ioco}_{(S,E,T)}$ $\mathcal{H}^-$ implies that each entry of the observation table must be complete and, thus, $Straces(\mathcal{H}^-) = Straces(\mathcal{H}^+)$.

Theorem 3. *Let (S, E, T) be a globally closed and consistent observation table obtained from* SUL*, and i be an input-output transition system, then i $\boldsymbol{ioco}$ SUL $\Rightarrow$ i $\boldsymbol{ioco}_{(S,E,T)}$ SUL.*

Proof (Proof idea.). It is easy to prove the contrapositive by showing that the concatenation of a prefix s and a suffix e for which $out(i\ \mathbf{after}\ s{\cdot}e) \not\subseteq out(\text{SUL } \mathbf{after}\ s{\cdot}e)$ is in $Straces(\text{SUL})$.

We gave the basis for maintaining an observation table for learning a nondeterministic system and we provided the definitions of the hypotheses that can be constructed from the observation table. In the next section we will discuss the learning process and we will show how to use the concepts introduced so far.

5 Learning Process

When learning deterministic systems, under the assumption that a correct and sound equivalence oracle exists, it is proven that the learning process ends when the correct model has been learned. Due to nondeterminism, one can never be sure of having observed all the possible outputs after a given trace, thus we do not know when to stop the learning process.
In Figure 3 we give a flow diagram that includes the nondeterministic decision to either stop learning and keep the current hypotheses as final, even tough they do not represent exactly the system under learning, or continue learning, trying to obtain a more *precise* hypothesis and postpone the decision of stopping to a later time. The learning process starts with the initialization of the observation table. Then the table is updated, stabilized and checked for quiescence reducibility. After these steps, it might be necessary to update and stabilize the table again, and, subsequently, checking again for quiescence reducibility. Only when the table is stable and quiescence reducible, its preciseness is addressed. If the table is precise enough, then the learning can stop. Otherwise we can

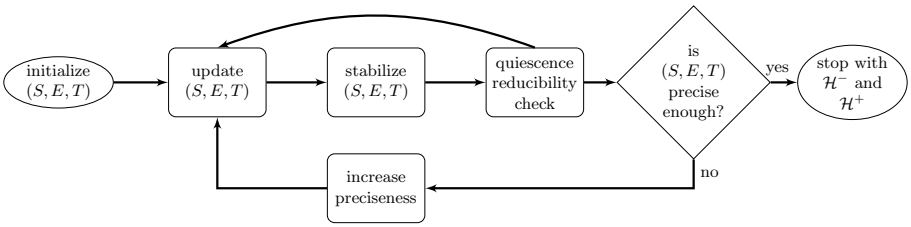


Fig. 3. Flow diagram of the learning process.

try to improve its preciseness. Given the nondeterministic behaviour, it is not always possible to increase the preciseness.
A table that is the result of updating another table, after having used a proper technique, should always be more precise than the initial one. In this section we define a qualitative preciseness relation on tables and show that the update does not decrease the preciseness. We introduce some techniques that attempt to increase it, either directly or by triggering an update of a stable table.

5.1 Preciseness of the Table

The preciseness of the observation table depends on three factors: 1) the size of the observation table, in term of rows and columns, i. e., the cardinality of $(S \cup S{\cdot}L_\delta)$ and E; 2) the cardinality of the set of outputs in each entry; and 3) the number of completed entries.
If two observation tables are comparable, i. e., they are constructed from the same SUL, and the first has a bigger observation table, bigger sets of output for some entries, and more completed entries than the second, then the first is more precise than the second.

Definition 4. *Let (S, E, T) and (S', E', T') be two observation tables. We say that (S', E', T') is more precise than (S, E, T), written as $(S, E, T) \sqsubseteq (S', E', T')$, if and only if:*

$$S \subseteq S' \textbf{ and } E \subseteq E' \textbf{ and}$$
$$\forall s \in (S \cup S{\cdot}L_\delta), e \in E : \begin{cases} T'(s{\cdot}e) =^+ T(s{\cdot}e) & \text{if } T(s{\cdot}e) \text{ is complete} \\ T'(s{\cdot}e) \supseteq T(s{\cdot}e) & \text{otherwise} \end{cases}$$

If an observation table is more precise than another one, then the relation it describes is stronger than the relation implied by the other observation table (Theorem 4). By increasing the preciseness of the table, we increase the strength of the relation.

Theorem 4. $(S, E, T) \sqsubseteq (S', E', T') \Rightarrow \textbf{\textit{ioco}}_{(S,E,T)} \supseteq \textbf{\textit{ioco}}_{(S',E',T')}$

It is easy to verify that Algorithm 1 and Algorithm 2 do not decrease the preciseness of the table, neither does the quiescence reducibility check.

Proposition 1. *During learning, each table is more (or equally) precise than the previous one.*

5.2 Techniques for Increasing the Preciseness

Classic L* stops with the exact model of the system under learning: if the current model is not the correct one, there must be a counterexample that will improve the model. In our approach, instead, we are more interested in learning an approximation of the SUL. If the approximation is not good enough we want to make it *better*. We can achieve this goal by acting on the observation table directly, adding elements to S, E and $T(s{\cdot}e)$ or by testing our current hypotheses.

Extending S and E The first method for increasing the preciseness of the table relies on directly enlarging the table, i. e., increment the number of rows and columns. The cardinality of S and E will therefore grow. Even though this is an easy modification to perform on the observation table, the choice of new elements for those sets is not trivial. It is important to keep S and E prefix and suffix closed, respectively. Thus good candidates for S are elements of $S{\cdot}L_\delta$, and good candidates for E are elements of $L_\delta{\cdot}E$. After having added elements to S and E the observation table must be updated and stabilized again.

Updating the Table Another method that acts on the table is to run Algorithm 1 with the goal of finding new outputs for existing entries, or making the table not globally closed or consistent. The former case increases the preciseness directly, while the latter will result in an increase of the preciseness due to the stabilizing step. This method is less direct than the previous one, because it might result in not increasing the preciseness at all. Consider a table where all entries are completed: updating the table will have no consequences.

Testing the Current Hypotheses The last method we mention is analogue to the classic learning process: once an hypothesis has been constructed, an equivalence oracle confirms that it is equivalent to the system under learning. If not, it provides a counterexample. In practice, this is done by testing. The hypothesis is tested for equivalence against the system under learning.
In learning nondeterministic systems, we can use an **ioco**-based model-based testing tool to test if the SUL is conforming to one of the hypotheses. We can search for a counterexample by testing against $\mathcal{H}^-$ or $\mathcal{H}^+$, using either $\mathbf{ioco}_{(S,E,T)}$ or classic **ioco**. Testing against $\mathcal{H}^+$ will find less counterexamples than testing against $\mathcal{H}^-$, because SUL $\mathbf{ioco}_{(S,E,T)}$ $\mathcal{H}^+$, see Theorem 2.
The conformance relation also plays an important role in the conformance query. If $\mathbf{ioco}_{(S,E,T)}$ is used, then the test is concentrated in finding new outputs for the entries in the table, while classic **ioco** can explore the SUL more widely. A reason for using $\mathbf{ioco}_{(S,E,T)}$ is that it is defined on a finite set of traces, making the testing process exhaustive.
By handling a counterexample, either we add an output in one or more already existing entries of the table, or we add some suffixes to E and prefixes to S, increasing the preciseness.

5.3 The Learning Algorithm

Combining all the algorithms presented in previous sections results in Algorithm 5. We start by initializing the observation table with a set

Algorithm 5 LearnLTS

```
1:  S ← {ϵ}                                  {Initialize (S, E, T)}
2:  E ← {ϵ}
3:  loop
4:    repeat
5:      update(S, E, T)
6:      Stabilize (S, E, T) using Algorithm 2
7:      Check quiescence reduc. on row and row⁺ obtaining suffix-closed sets E⁻ and E⁺
8:      E ← E ∪ E⁻ ∪ E⁺
9:    until E⁻ ∪ E⁺ = ∅
10:   Construct H⁻ using Algorithm 3
11:   Construct H⁺ using Algorithm 4
12:   if (S, E, T) is not precise enough then
13:     Try to increase preciseness
14:   else
15:     return H⁻, H⁺
```

containing only the empty trace ϵ for both prefixes and suffixes. At this point the table has one column, for the empty trace, and a row for each element of L_δ, plus a column for the empty trace. Then we fill it using Algorithm 1 and Algorithm 2. Quiescence reducibility is checked for both row and row^+ and, if that results in adding suffixes to the table, we update and stabilize it again and check again for quiescence reducibility. Once we obtain a stable and quiescence reducible table we consider whether the table is precise. If it is, then we are done: we construct the two hypotheses $\mathcal{H}^-$ and $\mathcal{H}^+$ and return them. If the table is not precise, we need to use some techniques to increase its preciseness.

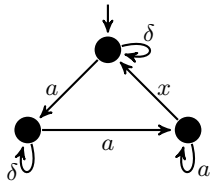
Example 4. The first three examples cover Algorithm 5 until the decision point at Line 12. Let's assume the preciseness is not high enough. We try to improve the preciseness by updating the table, and as a result we discover that $T(\epsilon{\cdot}\epsilon)$ is complete. The table is not globally closed any more, accordingly to row^+: we add a to S and update the table. The table is still not globally closed because of $row(aa)$, thus we add a to S obtaining Figure 4a. The table is globally closed, but not globally consistent. For row we have that $row(\epsilon) = row(a)$ but $row(\epsilon a) = row(a) \neq row(aa)$. We solve the inconsistency by adding a to E. This makes the table not globally closed, because of $a\delta$ and aax which are added to S obtaining the table in Figure 4b. This table is globally closed and consistent. It also is quiescence reducible, thus we can construct $\mathcal{H}^-$ and $\mathcal{H}^+$. Figure 4c shows $\mathcal{H}^-$, while $\mathcal{H}^+$ is shown in Figure 5a.
Compared with the previous hypotheses, these new ones contain more information. We want to increase the preciseness of the observation table by testing. We test the SUL against $\mathcal{H}^-$ for $\mathbf{ioco}_{(S,E,T)}$ and we find that, $out(\text{SUL } \textbf{after } aa) \neq out(\mathcal{H}^- \textbf{ after } aa)$ because we observed a y during our tests. Assume also that we tested the system enough to obtain `TRUE` in some entries. A possible observation table could be the one in Figure 5b

	ϵ
ϵ	$\langle\{\delta\}, \top\rangle$
δ	$\langle\{\delta\}, \top\rangle$
a	$\langle\{\delta\}, \bot\rangle$
aa	$\langle\{x\}, \bot\rangle$
δa	$\langle\{\delta\}, \bot\rangle$
$\delta\delta$	$\langle\{\delta\}, \top\rangle$
$a\delta$	$\langle\{\delta\}, \top\rangle$
aaa	$\langle\{x\}, \bot\rangle$
aax	$\langle\{\delta\}, \bot\rangle$

(a) An inconsistent observation table.

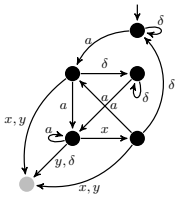
	ϵ	a
ϵ	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \bot\rangle$
δ	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \bot\rangle$
a	$\langle\{\delta\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
aa	$\langle\{x\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
$a\delta$	$\langle\{\delta\}, \top\rangle$	$\langle\{x\}, \bot\rangle$
aax	$\langle\{\delta\}, \bot\rangle$	$\langle\{\delta\}, \bot\rangle$
δa	$\langle\{\delta\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
$\delta\delta$	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \bot\rangle$
aaa	$\langle\{x\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
$a\delta a$	$\langle\{x\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
$a\delta\delta$	$\langle\{\delta\}, \top\rangle$	$\langle\{x\}, \bot\rangle$
$aaxa$	$\langle\{\delta\}, \bot\rangle$	$\langle\{x\}, \bot\rangle$
$aax\delta$	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \bot\rangle$

(b) A stable, more precise, observation table.



(c) The hypothesis $\mathcal{H}^-$ for Figure 4b.

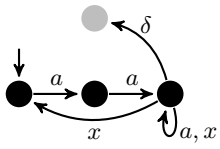
Fig. 4. The top part of the tables represents S and the bottom part represents $S{\cdot}L_\delta$.



(a) $\mathcal{H}^+$ for Figure 4b.

	ϵ	a
ϵ	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \top\rangle$
δ	$\langle\{\delta\}, \top\rangle$	$\langle\{\delta\}, \top\rangle$
a	$\langle\{\delta\}, \top\rangle$	$\langle\{x, y\}, \bot\rangle$
aa	$\langle\{x, y\}, \bot\rangle$	$\langle\{x, y\}, \bot\rangle$
$a\delta$	$\langle\{\delta\}, \top\rangle$	$\langle\{x, y\}, \bot\rangle$
aax	$\langle\{\delta\}, \bot\rangle$	$\langle\{\delta\}, \top\rangle$

(b) A more precise observation table.



(c) $\mathcal{H}^+$ for Figure 5b.

Fig. 5. The grey state represents chaotic behaviour.

($S{\cdot}L_\delta$ is not shown). If we construct $\mathcal{H}^-$ from Figure 5b, we obtain the SUL, $\mathcal{H}^+$ is shown in Figure 5c. We consider the table precise enough and we stop.

6 Conclusions

We presented an algorithm for learning nondeterministic input-output transition systems by using an active learning L*-style approach. The observation table has been modified to handle nondeterminism and unknown behaviour. We defined two different hypotheses, that can be derived from the modified observation table, which are able to describe the unknown behaviour in two different ways. We also adapted the properties that the table must satisfy for successfully inferring such hypotheses. The hypotheses are an under and an over-approximation of the SUL according to a newly defined relation $\mathbf{ioco}_{(S,E,T)}$. We uncoupled the membership and equivalence queries, used in classic L*, by following a learning process based on preciseness of the observation table: the learning stops, always with an $\mathbf{ioco}_{(S,E,T)}$ conforming model, when the table is considered precise enough, otherwise some actions can be taken to increase its preciseness. Thus, a conformance test, analogue of the equivalence query,

is not used directly in the learning process, but only as a mechanism to increase the preciseness. Stopping without reaching an isomorphism of the system under learning, contrary to $L^\star$, allows to obtain a valid, conforming, but approximate model which can be used to start (regression) testing.

Future Work In this paper we focus on qualitative approximation for the learned model. As future work we want to quantitatively define the preciseness using some measures. A possible measure is given by a combination of the size of S and E, the number of outputs in the table and the number of complete entries, avoiding redundancy (see Algorithm 1). Another measure we are considering is the discounted reachability of χ when $\mathcal{H}^+$ is considered as a *Markov chain*. After each modification of the table such measures are calculated and the learning process might end, because a certain value for the preciseness has been reached, or because the preciseness is not changing rapidly any more.
Moreover, we wish to investigate which hypothesis, $\mathcal{H}^-$ or $\mathcal{H}^+$ is better for regression testing, and for other testing and verification techniques where the use of a model is of help. Another topic concerns the techniques used in *incremental learning-based testing* [MS11] and their adaptation to active learning of nondeterministic systems. Finally, an experimental evaluation of our approach is necessary to assess its practical feasibility.

References

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation* 75(2):87–106, 1987.

[BRT04] M. van der Bijl, A. Rensink, J. Tretmans. Compositional Testing with ioco. In Petrenko and Ulrich (eds.), *FATES*. LNCS 2931, pp. 86–100. Springer, 2004.

[Ea10] K. El-Fakih, et al. Learning Finite State Models of Observable Nondeterministic Systems in a Testing Context. In *ICTSS*. Pp. 97–102. 2010.

[MS11] K. Meinke, M. Sindhu. Incremental Learning-Based Testing for Reactive Systems. In Gogolla and Wolff (eds.), *TAP*. LNCS 6706, pp. 134–151. Springer, 2011.

[Nie03] O. Niese. *An integrated approach to testing complex systems*. PhD thesis, University of Dortmund, 2003.

[Pa13] W. Pacharoen, et al. Active Learning of Non-deterministic Finite State Machines. *Mathematical Problems in Engineering* 2013:11, 2013.

[RS93] R. Rivest, R. Schapire. Inference of finite automata using homing sequences. In Hanson et al. (eds.), *Machine Learning: From Theory to Applications*. LNCS 661, pp. 51–73. Springer, 1993.

[Tre96] J. Tretmans. Test Generation with Inputs, Outputs and Repetitive Quiescence. *Software-Concepts and Tools* 3:103–120, 1996.

[VT14] M. Volpato, J. Tretmans. Active Learning of Nondeterministic Systems from an ioco Perspective. In Margaria and Steffen (eds.), *Leveraging Applications of Formal Methods, Verification and Validation.* LNCS 8802, pp. 220–235. Springer, 2014.

[Wil07] T. Willemse. Heuristics for ioco-Based Test-Based Modelling. In Brim et al. (eds.), *Formal Methods: Applications and Technology*. LNCS 4346, pp. 132–147. Springer, 2007.