2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)

Learning and Adaptive Testing of Nondeterministic State Machines

Alexandre Petrenko
CRIM - Computer Research Institute of Montréal
Montréal, Canada
Alexandre.Petrenko@crim.ca

Florent Avellaneda
CRIM - Computer Research Institute of Montréal
Montréal, Canada
Florent.Avellaneda@crim.ca

***Abstract*—The paper addresses the problems of active learning and conformance testing of systems modeled by nondeterministic Mealy machines (NFSM). It presents a unified SAT-based approach originally proposed by the authors for deterministic FSMs and now generalized to partial nondeterministic machines and checking experiments. Learning a nondeterministic black box, the approach neither needs a Teacher nor uses it a conformance tester to approximate equivalence queries. The idea behind this approach is to infer from a current set of traces not one, but two inequivalent conjectures, use an input sequence distinguishing them in an output query, and update the current trace set with an observed trace to obtain a new pair of distinguishable conjectures, if possible. The classical active learning problem is further generalized by adding a nondeterministic specification FSM, which defines the solution space. The setup unifies the learning and adaptive testing problems and makes them equisolvable with the proposed approach.**

Keywords—active learning, passive inference, nondeterministic FSM, adaptive testing, SAT solving

I. INTRODUCTION

State machines take an important place in model based testing of software. There exists a significant body of work addressing their use for automated test generation. Recent work also focuses on model inference of software and reactive systems. In fact, active learning and test generation are shown to be closely related [4], [22]. Most of the existing work considers that a deterministic state machine is an adequate model for testing purposes. However, a relatively little work has been done by the testing and model inference communities for nondeterministic models. It is sufficient to mention that the widely used benchmarks for automata learning and conformance testing [33] do not contain nondeterministic models. Nondeterminism typically this comes either from some abstraction that has been applied or there being a number of acceptable output sequences in response to some input sequence [12]. Moreover, the complexity and evolving nature of current systems increase uncertainty about their behavior and nondeterministic state machines can formalize such uncertainty in many applications.

The problem of inferring automata models, such as DFAs and FSMs, from sets of strings (traces) has been extensively studied in the literature; Kella [14] and Gold [8] seem to be among the first contributors. The problem is known to be computationally very hard, nevertheless, numerous proposals have been made, mainly on developing state merging techniques to transform a tree machine representing a given set of strings into a machine as small as possible that is consistent with this set [18], [19]. While the passive inference problem is important by itself from both, theoretical and practical, points of view, it is also considered as an essential step of active inference of automata models.

The existing methods for active inference, i.e., query learning of FSMs follow the basic idea of L^* approach [3] of using a Minimally Adequate Teacher, also called an oracle, [17], [4], [29], [5], [30], [31] to answer equivalence queries. An equivalence query is about a conjecture, for which the oracle is capable to provide a counterexample, i.e., an input sequence that distinguishes the conjecture from the FSM to be inferred. Equivalence queries are not realistic and a practical solution is to approximate them by random or complete test suites generated by conformance testing methods [21], [31]. Thus, in reality the role of the oracle is played by a black box FSM, supported by the additional assumption on the number of states in the black box.

Recently, we suggested an alternative approach for learning deterministic machines that eliminates the need to generate conformance tests for each intermediate conjecture and does not use any state identification facilities [22]. This is achieved by changing the objects of equivalence queries before converting them into output (membership) queries. The traditional equivalence query is about the target and a current conjecture FSM, while we proposed to use instead of the target FSM, another conjecture FSM. The idea is thus to infer from a current set of traces not one, but two inequivalent (distinguishable) conjectures, use an input sequence distinguishing them in an output query, and update the current trace set with an observed trace to obtain a new pair of distinguishable conjectures, if possible. The process converges as the number of conjectures with the fixed number

978-1-7281-3927-2/19/$31.00 ©2019 IEEE
DOI 10.1109/QRS.2019.00053

of states is bounded. The conjecture generation relies on satisfiability (SAT) solvers, as in similar work [1], [9]. A set of input sequences allows to infer an unknown FSM if and only if it is a checking experiment (complete test suite) for the FSM. This demonstrates that learning and conformance testing are closely related and can be solved by the same approach proposed for deterministic FSMs [22]. The incremental nature of the approach contributes to its scalability and allows for a premature termination resulting in an approximate inferred model and an incomplete but yet high-yield test suite.

In this paper, we further generalize the SAT-based approach for learning and testing to nondeterministic FSMs. The main challenge is that while for a nondeterministic automaton there exists an equivalent deterministic one, this does not hold for FSMs. An NFSM cannot be converted to a trace equivalent DFSM. As a result, inference and testing approaches for NFSMs cannot be straightforward extensions of their deterministic counterparts.

First, we observe that given a set of input/output strings (traces) it may not be possible to infer from it a deterministic FSM. A set of traces is not deterministic if it contains traces that have the same input sequence but different output sequences. The problem of inferring a nondeterministic FSM (NFSM) from such a set of traces has received much less attention compared to the deterministic case. There exist the state minimization methods for NFSMs [13] that can be used to merge states. They allow to obtain a single conjecture, but cannot construct several inequivalent conjectures. These are needed when passive inference of FSMs is just a step in the active inference via checking experiment generation, as in the deterministic case [22]. As soon as no more distinguishable conjectures can be found we conclude that the set of traces is a checking experiment identifying the machine. In this paper, we elevate the SAT-based approach for passive inference to the nondeterministic traces.

As to the problem of active inference of NFSMs, we are aware of only very few works [7], [20], [2]. All of them focus on extending the Angluin's algorithm L^* to nondeterministic machines and thus use the oracle to answer equivalence queries. The main difference from the deterministic case is that output queries need to be repeatedly made to a black box, until the assumption about fairness of nondeterministic implementations is satisfied. This assumption is also called "all weather conditions" [16] and complete testing assumption [15]. It is used in all the existing work on conformance testing from NFSMs [23], [32], [24], [12]. In practical terms, it means that for implementations from a given domain, the tester knows how many times and under which conditions tests must be re-executed to have a sufficient confidence about the completeness of the observed reactions of the implementations. While we also rely on this assumption, our goal here is to elaborate an active inference approach that does not need the oracle (Teacher) even if it is given a nondeterministic black box, as opposed to the existing work.

The existing approaches for testing from NFSMs follow offline or online testing scenarios. In the offline testing scenario, we need to generate from a given specification NFSM a test suite complete for a chosen fault model. Such test suites are in fact checking experiments. The resulting tests are then repeatedly executed against an implementation FSM unless it is known that it is deterministic, even though its specification is not. In this scenario, a conforming implementation must produce for each test only output sequences allowed by the specification NFSM, so the trace inclusion, often called a reduction, is the conformance relation [26], [25]. Repeated test execution satisfying the complete testing assumption is needed if the implementations cannot be assumed to be deterministic. In this case, we can use the reduction conformance relation or even a more stringent trace equivalence conformance relation. According to this relation, a conforming implementation should produce all the output sequences of the specification NFSM and only them for each test. Trace equivalence is the relation to be used when we want to learn an NFSM.

In the online testing scenario, test generation and execution are merged into one process allowing the tests to be adapted to an implementation under test [28], [6], [10], [11]. The process aims to verify whether a given implementation conforms to its specification. The online testing is an adaptive process and uses the trace inclusion conformance relation. The expected result of adaptive testing is a learned reduction of a given specification NFSM. The problem of adaptive testing from an NFSM is in fact a special case of a more general active inference problem statement we propose in this paper. Given a known NFSM and an unknown NFSM, infer the latter if it is a reduction of the former or determine a test that distinguishes the machines, otherwise. The classical setting of the FSM learning problem does not explicitly include the known FSM, but if we assume that it is in fact an NFSM with the set of all possible traces over the given input and output alphabet, often called a chaos machine, then it becomes a special case of the generalized active inference problem statement. In this paper, we demonstrate how our approach can be adopted to address this problem.

All the existing methods for test generation from NFSMs compose tests by concatenating the test fragments, namely, state preambles needed to reach states, state identifiers to check the reached state and trace traversal sets, which allow to execute transitions and to reach additional states if they are present in a given implementation. These fragments can be seen as generalization of the three types of input sequences constructed by the test generation methods previously developed for deterministic machines [15], [24]. Those are transfer, state identification and traversal sequences. The extension to the nondeterministic case is not trivial [24], [27]. Considering DFSMs, our approach based on SAT solving allows to determine complete test suites for the equivalence

relation and now we demonstrate how it can be enhanced for NFSMs to construct tests for both, equivalence and reduction relations.

The remaining of this paper is organized as follows. Section 2 recalls the basic definitions and notions for state machines. Section 3 presents a SAT-based method for passive inference of nondeterministic conjectures which is used for test generation from an NFSM in Section 4. Section 5 explains a method that allows to learn an unknown NFSM. Section 6 focuses on a more general active inference problem where we need to learn an unknown NFSM if it is a reduction of a known machine or determine a test that distinguishes the machines, otherwise. Section 7 concludes.

II. Definitions

A *Finite State Machine* or simply a (Mealy) machine M is a 5-tuple (S, s_0, I, O, T), where S is a finite set of states with an initial state s_0; I and O are finite non-empty disjoint sets of inputs and outputs, respectively; T is a transition relation $T \subseteq S \times I \times O \times S$, $(s, a, o, s') \in T$ is a transition. When we need to refer to the machine M initialized in a state $s \in S$, we write M/s.

M is *complete* (completely specified) if for each tuple $(s, a) \in S \times I$ there exists transition $(s, a, o, s') \in T$, otherwise it is *partial*. The machine is *trivial*, denoted Φ, if $T = \emptyset$. M is *deterministic* if for each $(s, a) \in S \times I$ there exists at most one transition $(s, a, o, s') \in T$, otherwise it is *nondeterministic*. M is *observable* if for each tuple $(s, a, o) \in S \times I \times O$ there exists at most one transition $(s, a, o, s') \in T$. In this paper we consider only observable machines, as any complete NFSM can be transformed into an observable machine. M is a *submachine* of $M' = (S', s_0, I, O, T')$ iff $S \subseteq S'$ and $T \subseteq T'$.

An *execution* of M/s is a finite sequence of transitions forming a path from s in the state transition diagram of M. The machine M is *initially* connected, if for each state $s \in S$ there exists an execution from s_0 to s. A string in $(IO)^*$ which labels an execution of M in some state is called a *trace* of M. Let $Tr(s)$ denote the set of all traces of M/s and $Tr(M)$ denote the set of traces of M. A *prefix* of trace $\omega \in Tr(s)$ is a trace $\omega' \in Tr(s)$ such that $\omega = \omega'\alpha$. For a trace $\omega \in Tr(s)$, we use s-after-ω to denote the state of M reached after the execution of ω, for an empty trace ε, s-after-$\varepsilon = s$. We also write M-after-ω instead of s_0-after-ω.

Given a trace ω over alphabets I and O, the *I-restriction* of ω is obtained by deleting from ω all symbols that are not in I, denoted $\omega{\downarrow_I}$. The *O-restriction* of ω, denoted $\omega{\downarrow_O}$, is similarly defined. The length of a trace is defined as the length of its $I(O)$-restriction.

Let $Out(s, \alpha)$ be the set of all output sequences produced by the input sequence $\alpha \in I^*$ in M/s, i.e., $Out(s, \alpha) = \{\omega{\downarrow_O} \mid \omega \in Tr(s), \omega{\downarrow_I} = \alpha\}$.

Given an input sequence $\alpha \in I^*$, states $s, s' \in S$ are *equivalent on* α, denoted $s \simeq_\alpha s'$, if $Out(s, \alpha) = Out(s', \alpha)$; they are not equivalent or *distinguishable*, denoted $s \not\simeq s'$, if there exists $\alpha \in I^*$ such that $Out(s, \alpha) \neq Out(s', \alpha)$. s' is a *reduction* of s *on* α, denoted $s' \lesssim_\alpha s$, if $Out(s', \alpha) \subseteq Out(s, \alpha)$; s' is *distinguishable* from s w.r.t. the reduction relation, denoted $s' \not\lesssim s$, if there exists $\alpha \in I^*$ such that $Out(s', \alpha) \not\subseteq Out(s, \alpha)$.

Complete FSMs M and M' are said to be *equivalent*, denoted $M \simeq M'$, if their initial states are equivalent on all input sequences in I^*, i.e., $Tr(M) = Tr(M')$; M' is a *reduction* of M, denoted $M' \lesssim M$, if the initial state of M' is a reduction of that of M on all input sequence in I^*, i.e., $Tr(M') \subseteq Tr(M)$.

Defining similar relations for partial machines in the context of conformance testing, we assume henceforth that while a specification FSM might be partial, any implementation is input-enabled, i.e., it is modelled by a complete FSM.

Given an FSM M, a state $s \in S$, and an input sequence $\alpha \in I^*$, α is said to be *defined* in s, if there exists a trace $\omega \in Tr(s)$, such that $\alpha = \omega{\downarrow_I}$. We use $\Theta(s)$ to denote the set of all defined input sequences for state s and $\Theta(M)$ for the state s_0, i.e., for M. If M is a complete machine then $\Theta(M) = Tr(M){\downarrow_I} = I^*$. A sequence $\alpha a \in I^*$ is said to be *undefined* in s, if $\alpha \in \Theta(s)$ but $\alpha a \notin \Theta(s)$. Furthermore, we assume that traces of M are *harmonized* [23], i.e., they satisfy the following property. If there exists a trace $\omega \in Tr(M)$, such that an input $a \in I$ is defined in M-after-ω then for all $\gamma \in Tr(M)$ such that $\gamma{\downarrow_I} = \omega{\downarrow_I}$, a is defined in M-after-γ. Henceforth, we consider that FSMs have only harmonized traces. Obviously, complete machines have only such traces.

Given a possibly partial FSM M and complete FSM M', M' is *quasi-equivalent* to M, denoted $M' \geq M$, if the initial state of M' is equivalent to that of M on all input sequences in $\Theta(M)$; M' is a *quasi-reduction* of M, denoted $M' \gtrsim M$, if M' is a reduction of M on all input sequences in $\Theta(M)$. Quasi-equivalence implies quasi-reduction, but not vice versa. We note that the traditional trace inclusion is called here the reduction, and the quasi-reduction is the trace inclusion only for defined input sequences, as the quasi-reduction allows traces for undefined sequences as well.

M' is said to be *distinguishable* from M for the quasi-equivalence or quasi-reduction if there exists an input sequence $\alpha \in \Theta(M)$ such that $Out(s'_0, \alpha) \neq Out(s_0, \alpha)$ or $Out(s'_0, \alpha) \not\subseteq Out(s_0, \alpha)$, respectively.

Next we define a *fault model* as a tuple of a specification FSM, conformance relation and fault domain [28]. In this paper, we have that a specification FSM $M = (S, s_0, I, O, T)$ can be partial and nondeterministic, hence the conformance relation is either the (quasi-) equivalence or (quasi-) reduction and the fault domain is the universe of all possible complete observable FSMs over the inputs I and a given number of states n, denoted $FD(n, I)$. Since the quasi-equivalence and quasi-reduction become the equivalence and reduction when M is complete, we shall consider just two fault models $\langle M, \geq, FD(n, I)\rangle$ and $\langle M, \gtrsim, FD(n, I)\rangle$.

Given a specification FSM M and its set of defined input sequences $\Theta(M)$, an input sequence $\alpha \in \Theta(M)$ is a *test* of M; a finite set of tests is a *test suite* of M. A test suite is said to be *exhaustive* for $\langle M, \geq, FD(n, I)\rangle$ or $\langle M, \gtrsim, FD(n, I)\rangle$ if for each

FSM $N \in FD(n, I)$ that is not (quasi-) equivalent to or a (quasi-) reduction of M, respectively, there exists a test that distinguishes N from M for the (quasi-) equivalence or (quasi-) reduction, respectively. Since a test suite contains only defined input sequences, it is sound and we will call a test suite *complete* for a given fault model if it is exhaustive. Such test suites are also called (preset) checking experiments [24].

III. Passive Inference of Nondeterministic Conjectures

Given the input I and output O alphabets, let Ω be a finite prefix-closed set of strings in $(IO)^*$. We represent this set by an FSM. Let $W(\Omega) = (X, x_0, I, O, P)$ be an observable tree FSM for which there exists a bijection $f: X \to \Omega$, such that $f(x_0) = \varepsilon$, $(x, a, o, x') \in P$ iff $f(x)ao = f(x')$, called the Ω-*machine*.

An FSM $C = (S, s_0, I, O, T)$ is called an Ω-*conjecture*, if $\Omega \subseteq Tr(C)$. The states of the Ω-machine $W(\Omega)$ and an Ω-conjecture C are closely related to each other. Formally, there exists a mapping $\mu: X \to S$, such that $\mu(x) = s_0$-after-$f(x)$. The mapping μ induces a partition π_C on the set X such that x and x' belong to the same block of the partition π_C, denoted $x =_{\pi_C} x'$, iff $\mu(x) = \mu(x')$.

Given an Ω-conjecture C with the partition π_C, let D be an Ω'-conjecture with the partition π_D, such that $\Omega' \subseteq \Omega$, we say that the partition π_C is an *expansion* of the partition π_D, if its projection onto Ω' coincides with the partition π_D.

The set Ω is said to be *nondeterministic* if there exist $\omega, \omega' \in \Omega$, such that $\omega{\downarrow_I} = \omega'{\downarrow_I}$ and $\omega{\downarrow_O} \neq \omega'{\downarrow_O}$, otherwise it is *deterministic*.

Addressing the problem of inferring nondeterministic FSMs, we further elaborate our SAT-solving approach [22] starting from its basic step, passive inference of an FSM. To generalize the approach to NFSMs, we need to first enhance this step to deal with nondeterministic traces.

To achieve this, we retain the procedure, formalized in Algorithm 1[9] to infer a conjecture that differs from already considered conjectures represented by the partitions on the set of states of the Ω-machine. As before, we aim at obtaining an Ω-conjecture with a given number of states n. Note that a minimal number could be found by iteration. The modifications extend the encoding of deterministic traces into a Boolean formula *formula* [22] to a nondeterministic set of traces as follows.

Let $W(\Omega) = (X, x_0, I, O, P)$ be an Ω-machine. We need to find an Ω-conjecture $C = (S, s_0, I, O, T)$ with at most n states, i.e., $|S| \leq n$. To this end, a mapping $\mu: X \to S$ should fulfill the following constraints:

$\forall x, y \in X$: if $x \not\simeq y$ then $\mu(x) \neq \mu(y)$ and
if $Out(x, a) = Out(y, a)$ for some $a \in I$
then $\mu(x) = \mu(y) \Rightarrow \mu(x)$-after-$ao = \mu(y)$-after-$ao$ for $\forall o \in Out(x, a)$ (1)

A mapping μ satisfying (1) defines a partition on X and each block becomes a state of the Ω-conjecture.

These formulas are then translated to SAT using unary coding for integer variables, represented by n Boolean variables $v_{x,0}, \ldots, v_{x,n-1}$. For each $x \in X$, we have the clause:

$$v_{x,0} \vee \ldots \vee v_{x,n-1} \quad (2)$$

These clauses mean that each state should be in at least one block.

For each state $x \in X$ and all $i, j \in \{0, \ldots, n - 1\}$ such that $i \neq j$, we have the clauses:

$$\neg v_{x,i} \vee \neg v_{x,j} \quad (3)$$

These clauses mean that each state should be in at most one block. The clauses 2 and 3 encode the constraint that each state should be in exactly one block.

We also use auxiliary variables $e_{x,y}$. For every $x, y \in X$ such that $x \not\simeq y$ we have

$$\neg e_{x,y} \quad (4)$$

This ensures that distinguishable states are not merged into one state in the Ω-conjecture.

For every $x, y \in X$ such that $Out(x, a) = Out(y, a)$, we have

$$e_{x,y} \Rightarrow e_{x\text{-after-}ao,y\text{-after-}ao}$$
for $\forall o \in Out(x, a)$ (5)

This ensures that the successors of two merged states for each input/output pair are also merged and the resulting FSM is observable.

For every $x, y \in X$ and all $i \in \{0, \ldots, n - 1\}$

$$e_{x,y} \wedge v_{x,i} \Rightarrow v_{y,i} \quad (6)$$

$$\neg e_{x,y} \wedge v_{x,i} \Rightarrow \neg v_{y,i} \quad (7)$$

The resulting Boolean formula is the conjunction of clauses (2) - (7). To check its satisfiability one can use an existing solver, calling the function *call-solver*(*formula*), as follows.

Algorithm 1. *Infer_conjecture*(Ω, n, Π)
Input: A set of traces Ω, an integer n, and a set of partitions Π
Output: An Ω-conjecture with at most n states such that its partition does not expand any partition in Π or False.
1. *formula* = conjunction of the clauses (2) - (7)
2. **for all** $\pi \in \Pi$ do
3. *clause* = False
4. **for all** x, y such that $x =_\pi y$ **do**
5. *clause* = *clause* $\vee \neg e_{x,y}$
6. **end for**
7. *formula* = *formula* $\wedge$ *clause*
8. **end for**
9. **return** *call-solver*(*formula*)

If a solution exists then we have an Ω-conjecture with n or fewer states. The latter is obtained from the determined partition on X.

In the context of conformance testing and inference of NFSMs, which are the focus of this paper, we have no Ω-machine, as we need to determine traces which define a checking experiment or infer an unknown NFSM. Building a checking experiment, the number of states of an implementation machine in a chosen fault model can be assumed to be that of a given specification NFSM or even exceeding it by a given number. Active inference, i.e., learning of an NFSM, also needs a bound on the possible number of states of a black box in order to be able to terminate the process. Knowing the maximal number of states is also needed to terminate random testing and conformance testing which approximate equivalence queries in all the existing learning methods.

IV. Test Derivation from Nondeterministic FSM

Addressing the conformance testing problem, we assume that a specification machine $M = (S, s_0, I, O, T)$ could be nondeterministic and partial (with harmonized traces), while the fault domain $FD(n, I)$ contains complete FSMs. To simplify our discussion, we also assume that the fault domain $FD(n, I)$ includes nondeterministic machines only if the specification is nondeterministic.

In this section we propose a method for offline testing. Online testing is addressed in Section 6.

Considering quasi-equivalence as a conformance relation we need to compare the behavior of a complete implementation machine to that of its possibly partial specification FSM. To this end, we define their intersection, called here the product for quasi-equivalence.

Given two FSMs $M = (S, s_0, I, O, T)$ and $M' = (S', s'_0, I, O, T')$, the FSM (P, p_0, I, O, H), where $p_0 = (s_0, s'_0)$ such that P and H are the smallest sets satisfying the following rule: if $(s, s') \in P$, $Out(s, a) = Out(s', a)$, $(s, a, o, t) \in T$, $(s', a, o, t') \in T'$, then $(t, t') \in P$ and $((s, s'), a, o, (t, t')) \in H$, is called the *product for quasi-equivalence*, denoted $M \times_{\geq} M'$. For complete machines, we shall also use $M \times_{\simeq} M'$ to denote the product.

The definition extends the definition of a product (aka intersection) for deterministic machines. Undefined input sequences of the product can indicate that the given machines are not quasi-equivalent, as the following lemma states.

Lemma 1. $M' \geq M$ if and only if $\Theta(M) = \Theta(M \times_{\geq} M')$.

Proof. If $M' \geq M$, then by definition, for each $\alpha \in \Theta(M)$, $Out(s_0, \alpha) = Out(s'_0, \alpha)$. Let $M \times_{\geq} M' = (P, p_0, I, O, H)$. Then by construction, we know that $Out(p_0, \alpha) = Out(s_0, \alpha) = Out(s'_0, \alpha)$. So $\Theta(M) = \Theta(M \times_{\geq} M')$.

Assume now that $\Theta(M) = \Theta(M \times_{\geq} M')$. So if $\alpha \in \Theta(M)$ then $\alpha \in \Theta(M \times_{\geq} M')$ and then by construction $Out(s_0, \alpha) = Out(s'_0, \alpha)$. Then by definition, $M' \geq M$. ■

We use this property to conclude that a conjecture is quasi-equivalent to the specification NFSM.

The following procedure builds a complete test suite incrementally. It extends the one developed recently [22] in several aspects:

- it derives checking experiments instead of checking sequences and thus does not need FSMs be strongly connected,
- the specification FSM can be nondeterministic, moreover, it can be a partial machine, while previously we considered only complete and deterministic machines.

The procedure iteratively generates from a current set of tests a conjecture that has not yet been considered and adds a new test if the conjecture is not quasi-equivalent to the specification machine. It terminates when no more conjecture with at most n states distinguishable from the specification FSM is left.

Algorithm 2. Generating a complete test suite for the fault model $<M, \geq, FD(n, I)>$

Input: An FSM $M = (S, s_0, I, O, T)$ and n, $n \geq |S|$.

Output: A complete test suite for the fault model $<M, \geq, FD(n, I)>$

1. $\Omega := \varnothing$
2. $\Psi := \varnothing$
3. $\Pi := \varnothing$
4. **while** a conjecture C is returned by *Infer_conjecture*(Ω, n, Π) **do**
5. **if** $\Theta(M) = \Theta(M \times_{\geq} C)$ **then**
6. $\Pi := \Pi \cup \{\pi_C\}$
7. **else**
8. Determine a shortest trace ω such that $\omega\downarrow_I \in \Theta(M) \cap \Theta(M \times_{\geq} C)$ and state $(M \times_{\geq} C)$-after-ω has no transition for some input a while state M-after-ω has
9. $\Psi := \Psi \cup \{\omega\downarrow_I a\}$
10. $\Omega := \Omega \cup \{\alpha \in Tr(M) \mid \alpha\downarrow_I = \omega\downarrow_I a\}$
11. **end if**
12. **end while**
13. **return** Ψ

Algorithm 2 returns a set of input sequences Ψ that is a checking experiment, i.e., a complete test suite for the fault model $<M, \geq, FD(n, I)>$. It means that any implementation of the fault domain $FD(n, I)$ to be conforming for the quasi-equivalence must produce all the output sequences of the specification machine M in response to each test in Ψ and only them. On the other hand, if it has no output sequences M cannot produce, but fails to produce all the output sequences of M in response to each test then it is a quasi-reduction of M, as stated in the following.

Theorem 2. Given an FSM M and a test suite Ψ generated by Algorithm 2, for each FSM $N \in FD(n, I)$,

- $N \geq M$ if and only if $N \simeq_{\Psi} M$

- $N \gtrapprox M$ if and only if $N \lesssim_\Psi M$.

Proof. $N \geq M$ implies $N \simeq_\Psi M$ because $\Psi \subseteq \Theta(M)$. Let us demonstrate that $N \simeq_\Psi M$ implies $N \geq M$. Assume that for some complete FSM $N = (S', s'_0, I, O, T')$, it holds that $N \simeq_\Psi M$, but N is not quasi-equivalent to M. According to the post-condition of *Infer_conjecture*(Ω, n, Π), there is no more conjecture from the set $FD(n, I)$ left that is distinguishable from M. The FSM N could have been excluded if its partition was placed in the set Π, but it is not quasi-equivalent to M, so its partition is not in Π. Then $N \notin FD(n, I)$, i.e., it has more states than n. A contradiction proves the statement.

$N \gtrapprox M$ implies $N \lesssim_\Psi M$ because $\Psi \subseteq \Theta(M)$. Let us demonstrate that $N \lesssim_\Psi M$ implies $N \gtrapprox M$. Assume that for some complete FSM $N = (S', s'_0, I, O, T')$, it holds that $N \lesssim_\Psi M$, but N is not a quasi-reduction to M i.e., there exists an $\alpha \in \Theta(M)$ such that $Out(N, \alpha) \not\subseteq Out(M, \alpha)$. So $\Theta(M) \neq \Theta(M \times_\geq N)$. The FSM N could have been excluded if its partition was placed in the set Π, but it is not quasi-equivalent to M, so its partition is not in Π. Then $N \notin FD(n, I)$, i.e., it has more states than n. A contradiction proves the statement. ■

Algorithm 2 can also be used to check whether a given test suite is complete for the fault model $<M, \geq, FD(n, I)>$ and to find additional tests to make it complete, if needed. It is sufficient to initialize Ψ to the given test suite.

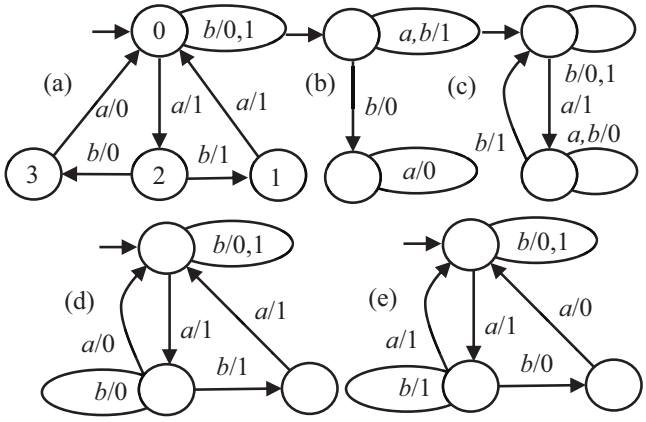

Fig. 1. Constructing a checking experiment for the fault model $<M, \geq, FD(n, I)>$.

Example. Consider the FSM M in Fig. 1 (a), it is partial and nondeterministic. We let n be four, the number of states in M and use Algorithm 2. In this and other examples, for brevity, we present mostly the intermediate conjectures and tests used to create them. Also to simplify the formulas we do not include all the prefixes in the sets Ω and Ψ.

Inputs a and b lead to a complete Ω-conjecture C_1 obtained from the set of traces $\Omega = \{a1, b0, b1\}$. It is a single state machine with three transitions. $\Psi = \{a, b\}$. The product $M \times_\geq C_1$ has a state $(M \times_\geq C)$-after-$a1b0$, where input a is not defined, since the FSM M has output 0 for this input, but C_1 has output 1. The input sequence aba is included into the set $\Psi = \{aba, b\}$. We have that $\Omega = \{a1, b0, b1\} \cup \{a1b0a0, a1b1a1\} = \{a1b0a0, a1b1a1, b0, b1\}$. The conjecture C_2 is shown in Fig. 1 (b). The product $M \times_\geq C_2$ has a state $(M \times_\geq C_2)$-after-$b0$, where input a is not defined, since the FSM M has output 1 for this input, but C_2 has output 0. The input sequence ba is included into the set $\Psi = \{aba, ba\}$. Then traces $b0a1$, $b1a1$ are added to $\Omega = \{b0, b1, a1b0a0, a1b1a1\} \cup \{b0a1, b1a1\} = \{a1b0a0, a1b1a1, b0a1, b1a1\}$. The conjecture C_3 is shown in Fig. 1 (c). The product $M \times_\geq C_3$ has a state $(M \times_\geq C_3)$-after-$a1b0a0$, where input a is not defined, since the FSM M has output 1 for this input, but C_3 has output 0. The input sequence $abaa$ is included into the set $\Psi = \{abaa, ba\}$. Traces $a1b1a1a1$, $a1b0a0a1$ are added to $\Omega = \{a1b0a0, a1b1a1, b0a1, b1a1\} \cup \{a1b1a1a1, a1b0a0a1\} = \{a1b1a1a1, a1b0a0a1, b0a1, b1a1\}$. Fig. 1 (d) shows the conjecture C_4 that is quasi-equivalent to M, since $\Theta(M) = \Theta(M \times_\geq C_4)$. We determine a partition on the states of the Ω-machine induced by C_4, $\pi_{C4} = \{\varepsilon, b0, b1, a1b1a1, a1b0a0; a1, a1b0, b0a1, b1a1, a1b1a1a1, a1b0a0a1; a1b1\}$[1], by grouping all traces leading to each state and include it into $\Pi = \{\pi_{C4}\}$. Updated constraints result in a new conjecture C_5, shown in Fig. 1 (e), that is also quasi-equivalent to M, since $\Theta(M) = \Theta(M \times_\geq C_5)$. We add to Π the partition induced by C_5, $\pi_{C5} = \{\varepsilon, b0, b1, a1b1a1, a1b0a0; a1, a1b1, b0a1, b1a1, a1b1a1a1, a1b0a0a1; a1b0\}$. Now $\Pi = \{\pi_{C4}, \pi_{C5}\}$. The updated constraints are not satisfiable. Therefore, the set of input sequences $\Psi = \{abaa, ba\}$ is a checking experiment for the fault model $<M, \geq, FD(n, I)>$. ■

Compared to the existing methods for test generation from nondeterministic FSMs [15], [12], [24] the proposed method exhibits the following advantages:

- It is the only method which can check completeness of a given test suite for an NFSM and the quasi-equivalence relation.
- It is also the only incremental test generation method for NFSMs. Test generation process can thus be terminated to avoid test explosion and fault coverage of the resulting tests can be estimated by considering intermediate conjectures.
- The method eliminates several potential sources of test redundancy inherent to the existing methods. In fact, to determine each test it solves only the shortest path problem, while the existing methods construct tests from fragments such as transfer sequences to reach states, state identification and traversal sequences so each fragment is determined as a (approximate) solution of a (non-trivial) optimization problem [32], [27].

V. Active Learning of Nondeterministic FSM

We now elevate our approach for active learning of deterministic FSM that is based on SAT solving to nondeterministic machines.

[1] We separate elements of blocks by comma and blocks by semicolon.

When dealing with a black box NFSM N, we rely, as before, on the traditional complete testing assumption, which sets a bound on the number of repetitive applications of input sequences in output queries. Using output queries we can still determine all the traces triggered in N by an input sequence, even though the whole set of traces $Tr(N)$ is a priory unknown. Thus, given a sequence $\beta \in I^*$, the set of traces $\{\alpha \in Tr(N) \mid \alpha\downarrow_I = \beta\}$ can be determined using the black box.

The following algorithm follows the steps of Algorithm 2, the main difference is that in the absence of a specification machine to check equivalence, it uses a current conjecture instead and when non-equivalence is established to determine a counterexample trace it uses the black box.

Algorithm 3. Learning an NFSM and determining its checking experiment for the equivalence relation

Input A black box that behaves like an unknown FSM N over the input set I with at most n states

Output The FSM N and a complete test suite for the fault model $<N, \simeq, FD(n, I)>$

1. $\Omega := \varnothing$
2. $\Psi := \varnothing$
3. $\Pi := \varnothing$
4. $C := \Phi$ (the trivial FSM)
5. **while** a conjecture D is returned by *Infer_conjecture*(Ω, n, Π) **do**
6. **if** $C \times_{\geq} D$ is complete **then**
7. $\Pi := \Pi \cup \{\pi_D\}$
8. **else**
9. Determine a shortest trace ω such that state $(C \times_{\geq} D)$-after-ω has no transition for some input $a \in I$
10. $\Psi := \Psi \cup \{\omega\downarrow_I a\}$
11. $\Omega := \Omega \cup \{\alpha \in Tr(N) \mid \alpha\downarrow_I = \omega\downarrow_I a\}$ * the black box N is used to obtained new traces caused by the input sequence $\omega\downarrow_I a$ *
12. **if** $\Omega \not\subset Tr(C)$ **then**
13. $C :=$ *Infer_conjecture*(Ω, n, Π)
14. **end if**
15. **end if**
16. **end while**
17. **return** C and Ψ

Theorem 3. If a black box behaves like an FSM N with the input set I and n states, Algorithm 3 infers it and returns a complete test suite for the fault model $<N, \cong, FD(n, I)>$.

Proof. When Algorithm 3 terminates, there is no more a satisfiable conjecture *Infer_conjecture*(Ω, n, Π). So, for each $N' \in FD(n, I)$, such that $\Omega \subseteq Tr(N')$, its partition expands a partition in Π. This means that $N \times_{\geq} N'$ is complete. Then Ψ is a complete test suite for the fault model $<N, \simeq, FD(n, I)>$. The termination is ensured by the fact that in each execution of the loop, the number of satisfiable conjectures *Infer_conjecture*(Ω, n, Π) strictly decreases. ■

Note that the procedure may have a jumpstart if provided with some input sequences which are a priori known to expose a rich behavior of the unknown NFSM. It may well be the case that the domain expert has some background knowledge of at least some features of the machine to infer.

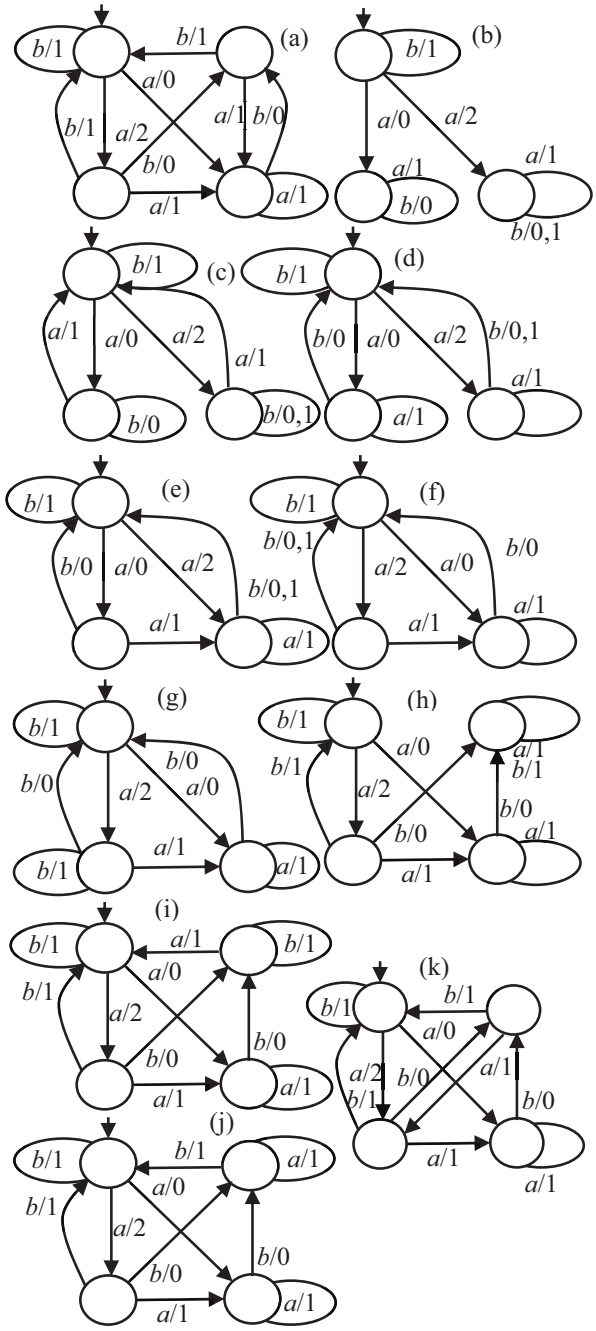

Fig. 2. Illustrating active inference of an NFSM.

Example. We illustrate Algorithm 3 using an NFSM N shown in Fig. 2 (a), assuming that the black box has at most four states, so $n = 4$.

The first conjecture C_1, obtained after we applied the inputs in $\Psi = \{a, b\}$ to the black box, observed the traces $\Omega = \{a0, a2, b1\} \subset Tr(N)$, and resolved the constraints, is a single state NFSM with three transitions. The conjecture is regenerated again as D_1. The partition $\pi_{D1} = (\varepsilon, a0, a2, b1\}$ is included into the set Π.

The conjecture D_2 is a two-state machine, $a2$ and $b1$ label self-looping transitions in the initial state, $a0$ labels another transition to the second state. The product $C_1 \times_{\geq} D_2$ has no transitions in the state $(C_1 \times_{\geq} D_2)$-after-$a0$, so we use now aa to obtain $\Psi = \{aa, b\}$ and $\Omega = \{a0a1, a2a1, b1\}$. The conjecture C_2 is a two-state NFSM, the initial state has a self-looping transition labeled with $b1$; $a0$ and $a2$ label transitions to the second state, which has a self-looping transition with $a1$.

To complete conjectures, we add the sequence ab to Ψ, obtaining $\Psi = \{aa, ab, b\}$. The set of observed traces of N becomes $\Omega = \{a0a1, a0b0, a2a1, a2b0, a2b1, b1\}$. The conjecture C_3 has now three states, it is shown in Fig. 2 (b). The conjecture is regenerated again as D_3. The partition $\pi_{D3} = \{\varepsilon, b1; a0, a0a1, a0b0; a2, a2a1, a2b0, a2b1\}$ is included into the set Π.

The conjecture C_4 is shown in Fig. 2 (c). $(C_4 \times_{\geq} D_3)$-after-$a0a1$ has no transition on a; in fact, two machines are distinguished by the input sequence aaa: $\{a0a1a1, a2a1a1\} \subset Tr(C_3)$, but $\{a0a1a0, a2a1a2, a2a1a0\} \subset Tr(D_1)$. The input sequence is added into $\Psi = \{aaa, ab, b\}$. The set of observed traces of N becomes $\Omega = \{a0a1a1, a0b0, a2a1a1, a2b0, a2b1, b1\}$.

The conjecture C_5 is shown in Fig. 2 (d). $(C_5 \times_{\geq} D_3)$-after-$a0b0$ has no transition on b; we determine the input sequence abb that distinguishes C_5 and D_3. Ψ becomes $\{aaa, abb, b\}$. The set of observed traces of N is now $\Omega = \{a0a1a1, a0b0b1, a2a1a1, a2b0b1, a2b1b1, b1\}$. $\Omega \subset Tr(C_5)$, then the conjecture is generated again as D_4. The partition $\pi_{D4} = \{\varepsilon, b1, a2b0, a2b1, a2b0b1, a2b1b1, a0b0, a0b0b01; a0, a0a1, a0a1a1; a2, a2a1, a2a1a1\}$ is added into the set Π.

The conjecture D_5 is shown in Fig. 2 (e). $(C_5 \times_{\geq} D_5)$-after-$a0a1$ has no transition on b, in fact, the two machines are distinguished by the input sequence aab: $\{a0a1b0, a2a1b0, a2a1b1\} \subset Tr(C_5)$, but $\{a0a1b0, a0a1b1, a2a1b0, a2a1b1\} \subset Tr(D_5)$. The input sequence aab is added into $\Psi = \{aaa, aab, abb, b\}$. The set of observed traces of N is now $\Omega = \{a0a1a1, a0a1b0, a0b0b1, a2a1a1, a2a1b0, a2b0b1, a2b1b1, b1\}$.

The conjecture C_6 is shown in Fig. 2 (f). When it is generated again as D_6 its partition $\pi_{D6} = \{\varepsilon, b1, a0b0, a0b0b1, a0a1b0, a2b0, a2b1, a2b0b1, a2b1b1, a2a1b0; a2; a0, a0a1, a0a1a1, a2a1, a2a1a1\}$ is added into the set Π.

Fig. 2 (g) shows the conjecture D_7. $(C_6 \times_{\geq} D_7)$-after-$a2b1$ has no transition on a, in fact, the two machines are distinguished by the input sequence aba: $\{a0b0a0, a0b0a2, a2b0a0, a2b0a2, a2b1a0, a2b1a2\} \subset Tr(C_6)$, but $\{a0b0a0, a0b0a2, a2b0a0, a2b0a2, a2b1a1\} \subset Tr(D_7)$. The sequence aba distinguishing C_6 and D_7 is added into $\Psi = \{aaa, aab, aba, abb, b\}$. The set of observed traces of N is now $\Omega = \{a0a1a1, a0a1b0, a0b0a1, a0b0b1, a2a1a1, a2a1b0, a2b0a1, a2b1a0, a2b1a2, a2b0b1, a2b1b1, b1\}$.

This set yields a new conjecture C_7 shown in Fig. 2 (h). When it is generated again as D_8 its partition π_{D8} is added into the set Π. A subsequent conjecture generation results in D_9 shown in Fig. 2 (i). The product $C_7 \times_{\geq} D_9$ has the input sequence $abaa$ that distinguishes C_7 and D_9. Ψ becomes $\{aaa, aab, abaa, abb, b\}$. $\Omega = \{a0a1a1, a0a1b0, a0b0a1a1, a0b0b1, a2a1a1, a2a1b0, a2b0a1a1, a2b1a0a1, a2b1a2a1, a2b0b1, a2b1b1, b1\}$.

C_8 is shown in Fig. 2 (j). When it is generated again as D_{10} its partition π_{D10} is added into the set Π. Fig. 2 (k) shows a next conjecture D_{11}. $(C_8 \times_{\geq} D_{11})$-after-$a0b0a1a1$ has no transition on b, in fact, the two machines are distinguished by the input sequence $abaab$. Ψ becomes $\{aaa, aab, abaab, abb, b\}$. The observed traces of N are $\Omega = \{a0a1a1, a0a1b0, a0b0a1a1b0, a0b0b1, a2a1a1, a2a1b0, a2b0a1a1b0, a2b0b1, a2b1a0a1b0, a2b1a2a1b0, a2b1b1, b1\}$. The generated conjecture C_9 is isomorphic to the NFSM N in Fig. 2 (a). Adding its partition to Π results in insatisfiable constraints. The set $\Psi = \{aaa, aab, abaab, abba, b\}$ is a complete test suite for the fault model $<N, \simeq, FD(n, I)>$. ■

Using this example, we now compare our approach to the traditional approach of simulating equivalence queries by output queries using conformance testing methods [21], [31]. The W-method applied to the NFSM in Fig. 2 (a) returns a complete test suite of 17 tests of the total length 70, reset included. These tests are used only in the final step of learning, each of a dozen of intermediate conjectures would need a test suite of a comparable size as well. In the worst case situation, all of these test suites would have to be executed as test queries. Algorithm 3 returns a test suite of just five tests of the total length 21, reset included.

VI. Adaptive Testing as Active Learning with a Specification NFSM

We now consider a problem that is more general than the one considered in the previous section. Namely, given a complete (specification) NFSM $M = (S, s_0, I, O, T)$ and a black box that behaves like a reduction of M with at most n states, learn the reduction.

On the one hand, this problem reduces to the classical FSM inference problem when M is a single state chaos machine such that $Tr(M) = (IO)^*$. Traditional work addresses the deterministic case, while in Section 5 we proposed an approach to learn nondeterministic machines; in either case, there was in fact no need to consider the chaos machine, as it contains all the possible behaviors. If, however, the NFSM M is less nondeterministic than the chaos machine then it does constrain the solution space for the learning problem. Adding this machine to the learning problem's setup might be useful in practice when the learner has some hypotheses about the behavior the system to be learned, which are then formalized in

an NFSM. For example, the learner may have its earlier model and some information about possible updates.

On the other hand, research on adaptive conformance testing addresses a similar problem, testing a deterministic implementation from its NFSM specification [12], [6]. This is, in fact, also a special case of the more general learning problem of an unknown reduction of a specification machine, since only deterministic implementations are allowed.

To offer a unified solution for NFSM learning and adaptive conformance testing problems, we slightly adjust the above formulation as follows.

Given a complete NFSM $M = (S, s_0, I, O, T)$ and a black box that behaves like an FSM $N \in FD(n, I)$, learn N, if it is a reduction of M or determine a test that distinguishes them, otherwise. The remaining of this section aims at solving this problem.

To simplify the discussions, we focus on complete machines. At the same time, we define an FSM product for quasi-reduction, which becomes the reduction relation for complete FSMs.

Given FSMs $M = (S, s_0, I, O, T)$ and $M' = (S', s'_0, I, O, T')$, such that $\Theta(M) \subseteq \Theta(M')$, the FSM (P, p_0, I, O, H), where $p_0 = (s_0, s'_0)$ such that P and H are the smallest sets satisfying the following rule: if $(s, s') \in P$, $Out(s', a) \subseteq Out(s, a)$, $o \in Out(s', x)$, $(s, a, o, t) \in T$, $(s', a, o, t') \in T'$, then $(t, t') \in P$ and $((s, s'), a, o, (t, t')) \in H$, is called the *product for quasi-reduction*, denoted $M' \times_{\gtrsim} M$. The difference with the product for quasi-equivalence defined in Section 5 is as follows. The product for quasi-reduction has a transition for each common output and does not require that both machines have the same outputs for each common input, while the product for quasi-equivalence does require this.

Lemma 4. $M' \gtrsim M$ if and only if $\Theta(M) = \Theta(M' \times_{\gtrsim} M)$.

Proof. If $M' \gtrsim M$, then by definition, for each $\alpha \in \Theta(M)$, $Out(s_0, \alpha) \subseteq Out(s'_0, \alpha)$. Let $M \times_{\gtrsim} M' = (P, p_0, I, O, H)$. Then by construction, we know that $Out(p_0, \alpha) \subseteq Out(s'_0, \alpha)$. So $\Theta(M) = \Theta(M' \times_{\gtrsim} M)$.

Assume now that $\Theta(M) = \Theta(M \times_{\gtrsim} M')$. If $\alpha \in \Theta(M)$ then $\alpha \in \Theta(M \times_{\gtrsim} M')$ and by construction $Out(p_0, \alpha) = Out(s'_0, \alpha)$. So $M' \gtrsim M$. ■

The following procedure, similar to Algorithm 3, aims at determining distinguishable conjectures as reductions of the specification machine in order to formulate an output query. The latter are answered by the black box, whole traces are used to refine conjectures. It differs from Algorithm 3 in the following features

- it verifies whether a conjecture is a reduction of the specification machine
- it takes care of invalid outputs produced by a black box
- it employs both types of products for equivalence and reduction.

Algorithm 4. Learning a reduction of a known NFSM M

Input: A complete NFSM $M = (S, s_0, I, O, T)$ and black box that behaves like an FSM $N \in FD(n, I)$

Output: The FSM N if it is a reduction of M or a test that distinguishes them, otherwise

1. $\Omega := \varnothing$
2. $\Pi := \varnothing$
3. $\Psi := \varnothing$
4. $D :=$ Φ (the trivial FSM)
5. **while** a conjecture C is returned by *Infer_conjecture*(Ω, n, Π) **do**
6. **if** $C \times_{\gtrsim} M$ is complete **then**
7. **if** $C \times_{\simeq} D$ is complete **then**
8. $\Pi := \Pi \cup \{\pi_C\}$
9. **else**
10. Determine a shortest trace ω such that state $(C \times_{\simeq} D)$-after-ω has no transition for some input $a \in I$
11. **if** $Out(N\text{-after-}\omega, a) \setminus Out(M\text{-after-}\omega, a) \neq \varnothing$ **then**
12. **return** "$\omega\downarrow_I a$ distinguishes N and M"
13. **else**
14. $\Psi := \Psi \cup \{\omega\downarrow_I a\}$
15. $\Omega := \Omega \cup \{\alpha \in Tr(N) \mid \alpha\downarrow_I = \omega\downarrow_I a\}$
16. **if** $\Omega \not\subseteq Tr(D)$ **then**
17. $D :=$ *Infer_conjecture*(Ω, n, Π)
18. **end if**
19. **end if**
20. **end if**
21. **else**
22. Determine a shortest trace ω such that state $(C \times_{\gtrsim} M)$-after-ω has no transition for some input $a \in I$
23. **if** $Out(N\text{-after-}\omega, a) \setminus Out(M\text{-after-}\omega, a) \neq \varnothing$ **then**
24. **return** "$\omega\downarrow_I a$ distinguishes N and M"
25. **else**
26. $\Psi := \Psi \cup \{\omega\downarrow_I a\}$
27. $\Omega := \Omega \cup \{\alpha \in Tr(N) \mid \alpha\downarrow_I = \omega\downarrow_I a\}$
28. **if** $\Omega \not\subseteq Tr(D)$ **then**
29. $D :=$ *Infer_conjecture*(Ω, n, Π)
30. **end if**
31. **end if**
32. **end if**
33. **end if**
34. **return** D and Ψ

Theorem 5. Given FSMs $M = (S, s_0, I, O, T)$ and N, $N \in FD(n, I)$, Algorithm 4 learns the FSM N if it is a reduction of M and returns a test that distinguishes N and M, otherwise.

Proof. If Algorithm 4 returns a distinguishing test, then $Out(N\text{-after-}\omega, a) \setminus Out(M\text{-after-}\omega, a) \neq \varnothing$ for some ω and a and so, "$\omega\downarrow_I a$ distinguishes N and M". Assume now that Algorithm 4 returns an FSM N' such that $Tr(N') \neq Tr(N)$. If Algorithm 4 returns an FSM, this means that "*Infer_conjecture*(Ω, n, Π)" cannot find a conjecture. Since the set Ω contains only traces from $Tr(N)$, if "*Infer_conjecture*(Ω, n, Π)" cannot find a conjecture then Π contains a partition banning the conjecture

N. A partition is added to the set Π only if "$C \times_{\simeq} D$ is complete". So at one point, $N \times_{\simeq} D$ is complete, i.e., D is a reduction of M. When $N \times_{\simeq} D$ is complete, D will never change again because $\Omega \subseteq Tr(D)$. The termination of the algorithm is assured by the fact that in each loop, at least one conjecture is banned. ■

Corollary 6. If $N \gtrapprox M$ or M is a chaos machine then the set of input sequences returned by Algorithm 4 is a complete test suite for the fault model $<N, \simeq, FD(n, l)>$.

Example. We illustrate Algorithm 4 using a complete NFSM M shown in Fig. 3 (a) and a black box assuming that it behaves as a complete FSM N, shown in Fig. 3 (b), so we assume that $n = 2$.

The complete conjecture D_1 is obtained from the set of traces $\Omega = \{a0, b0\}$ using the set of input sequences $\Psi = \{a, b\}$. It is a single state machine with two transitions that is generated again as the conjecture C_1. The product $M \times_{\gtrapprox} C_1$ is a complete machine, so is the product $C_1 \times_{\simeq} D_1$ and the partition $\pi_{C1} = (\varepsilon, a0, b0\}$ is included into the set Π.

In the next iteration, *Infer_conjecture*(Ω, n, Π) returns the conjecture C_2 as a two-state machine, $a0$ labels a self-looping transition in the initial state, $b0$ labels another transition to the second state. The product has a state ($M \times_{\gtrapprox} C_2$)-after-$b0$, where input a is not defined. The input sequence ba is included into the set $\Psi = \{a, ba\}$.

The black box produces trace $b0a1$ when the sequence ba is applied, which is allowed by the specification machine M. $\Omega = \{a0, b0a1\}$, $\Omega \not\subset Tr(D_1)$ and D_1 is refined to D_2 that is a two-state machine, $a0$ labels a self-looping transition in the initial state, $b0$ labels a transition to the second state, $a1$ labels a transition back to the initial state. D_2 is generated again as the conjecture C_3. The product' state ($M \times_{\gtrapprox} C_3$)-after-$b0$ has b not defined. The input sequence bb is included into the set $\Psi = \{a, ba, bb\}$. The black box produces trace $b0b1$ in response to bb, which is allowed by the specification machine M. $\Omega = \{a0, b0a1, b0b1\}$, $\Omega \not\subset Tr(D_2)$ and D_2 is refined to D_3 shown in Fig. 3 (c). D_3 is generated again as the conjecture C_4. The product $M \times_{\gtrapprox} C_4$ is complete, so is the product $C_4 \times_{\simeq} D_3$ and the partition $\pi_{C4} = \{\varepsilon, a0, b0b1; b0, b0a1\}$ is included into the set Π. The next conjecture generated from the same set of traces is C_5 shown in Fig. 3 (d). The product $M \times_{\gtrapprox} C_5$ is complete, the product $C_5 \times_{\simeq} D_3$ is not, since the state ($C_5 \times_{\simeq} D_3$)-after-$a0$ has undefined input a. The sequence aa distinguishes C_5 and D_3. $Out(N\text{-after-}a0, a) = \{0\}$, determined by applying aa to the black box FSM N. $\Psi = \{aa, ba, bb\}$. $\Omega = \{a0a0, b0a1, b0b1\}$, $\Omega \not\subset Tr(D_3)$ and D_3 is refined to D_4 shown in Fig. 3 (e), regenerated then as C_6. The product $M \times_{\gtrapprox} C_6$ is complete, so is the product $C_6 \times_{\simeq} D_4$ and the partition $\pi_{C6} = \{\varepsilon, a0, a0a0; b0, b0a1, b0b1\}$ is included into the set Π.

The next conjecture generated from the same set of traces is C_7 shown in Fig. 3 (b) that is in fact the unknown FSM. The product $M \times_{\gtrapprox} C_7$ is complete, the product $C_7 \times_{\simeq} D_4$ is not, since the state ($C_7 \times_{\simeq} D_4$)-after-$b0b1$ has undefined input a. The sequence bba distinguishes C_7 and D_4. $Out(N\text{-after-}b0b1, a) = \{0\}$, determined by applying bba to the black box FSM N. $\Psi = \{aa, ba, bba\}$. $\Omega = \{a0a0, b0a1, b0b1a0\}$, $\Omega \not\subset Tr(D_4)$ and D_4 is refined to the conjecture D_5 shown in Fig. 3 (b) that is the unknown FSM. C_7 is generated again, the products $M \times_{\gtrapprox} C_7$ and $C_7 \times_{\simeq} D_5$ are complete and the partition $\pi_{C7} = \{\varepsilon, a0, a0a0, b0b1, b0b1a0; b0, b0a1\}$ is included into the set Π. The algorithm terminates, since the resulting constraints are not satisfiable. The FSM N is learned, and the set of tests $\Psi = \{aa, ba, bba\}$ is returned.

We compare this result with the test suite obtained by using the SC-method [24] for the specification FSM M in Fig. 3 (a) to generate a complete test suite for the fault model $<M, \gtrapprox, FD(n, l)>$. It contains 13 tests of the total length of 65. Adapting this test suite to the given implementation FSM N, a specification refinement method [26] yields a test suite of six tests of the total length of 18. Algorithm 4 returns three tests of the total length of 10. ■

The save could be explained by considering different goals of the two approaches. The adaptive testing approach needs to verify whether a given implementation is a reduction of the specification, while a complete test suite, i.e., a checking experiment has to verify whether each and every implementation in $FD(n, l)$ is a reduction, which requires more tests.

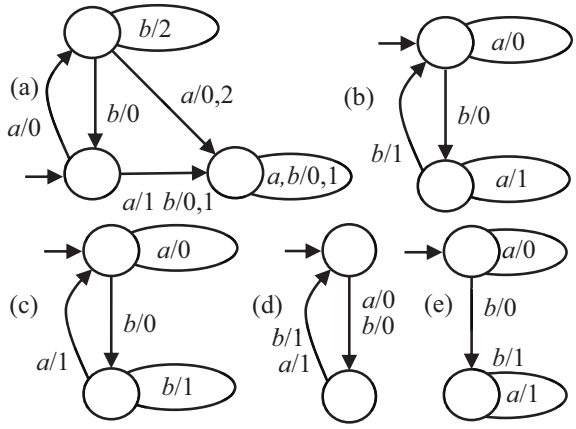

Fig. 3. Specification FSM M (a), black box FSM N (b) and intermediate conjectures (c), (d) and (e).

Compared to the existing methods for adaptive testing from NFSMs, the proposed method has the following advantages:

- It does not need to check whether the specification NFSM is reduced or has deterministically (or definitely) reachable states.
- It does not need any state identification, transfer and traversal sequences required by the existing methods.
- It does not only yield a conformance test suite but also identifies the implementation NFSM if it is a reduction of a given specification NFSM.

VII. Conclusion

We considered the problems of active learning and conformance testing of systems modeled by nondeterministic Mealy machines. We proposed a unified SAT-based approach addressing these problems which elevates the approach originally proposed for deterministic FSMs and checking sequences to partial nondeterministic machines and checking experiments. The learning approach neither needs a Teacher nor uses it a conformance tester to simulate/approximate equivalence queries substituting a Teacher, as opposed to all previous work. The idea behind this approach is to iteratively infer from a current set of traces not one, but two inequivalent (distinguishable) conjectures, use an input sequence distinguishing them in output query, and update a current trace set with an observed trace to obtain a new pair of distinguishable conjectures, if possible. We presented an extension of the encoding of the passive inference of FSMs to a SAT formulation that allows to deal with nondeterministic traces. The proposed approach demonstrates that active learning and conformance testing are in fact two sides of the same coin not only for deterministic but also for nondeterministic Mealy machines. We also generalized the classical active learning problem by adding a nondeterministic specification FSM, which constrains the solution space. The setup unifies the learning and adaptive testing problems for NFSMs and makes them equisolvable with the proposed approach.

Our current work is to complete the development of a prototype tool for learning and testing NFSMs. The tool should allow us to investigate how various encoding schemes effect the performance of the proposed approach.

Acknowledgment

This work was partially supported by MEI (Ministère de l'Économie et Innovation) of Gouvernement du Québec and NSERC of Canada.

References

[1] A. Abel and J. Reineke, "MeMin SAT-based exact minimization of incompletely specified Mealy machines," in IEEE/ACM International Conference on Computer-Aided Design, 2015, pp. 94-101.

[2] K. Ali, and A. Tacchella, "Learning nondeterministic Mealy machines," in International Conference on Grammatical Inference, 2014, pp. 109-123.

[3] D. Angluin, "Learning regular sets from queries and counterexamples." Information and Computation, 75(2), 1987. pp. 87–106.

[4] T. Berg et al. "On the correspondence between conformance testing and regular inference," in Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering, LNCS 3442, 2005, pp. 175-189.

[5] C. De la Higuera, Grammatical inference: learning automata and grammars. Cambridge University Press, 2010.

[6] M. Dorofeeva, A. Petrenko, M. Vetrova, N. Yevtushenko, "Adaptive test generation from a nondeterministic FSM," Radioelektronika i informatika. No. 3, 2004, pp. 91-95.

[7] K. El-Fakih, R. Groz, M. N. Irfan, M. Shahbaz, "Learning finite state models of observable nondeterministic systems in a testing context," In Proceedings of the 22nd IFIP International Conference on Testing Software and Systems (Short Papers), 2010. pp. 97-102.

[8] E. M. Gold. "Complexity of automaton identification from given data," Information and control, 37(3), 1978, pp. 302-320.

[9] M. J. Heule and S. Verwer. "Software model synthesis using satisfiability solvers. Empirical Software Engineering," 18(4), 2013, pp. 825–856.

[10] R. M. Hierons, "Adaptive testing of a deterministic implementation against a nondeterministic finite state machine," The Computer Journal, 41(5), 1998, pp. 349-355.

[11] R. M. Hierons, "Generating candidates when testing a deterministic implementation against a non-deterministic finite-state machine," The Computer Journal, 46(3), 2003, pp. 307-318.

[12] R. M. Hierons, "Testing from a non-deterministic finite state machine using adaptive state counting," IEEE Transactions on Computers, 53(10), 2004, pp. 1330-1342.

[13] T. Kam, T. Villa, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "Theory and algorithms for state minimization of nondeterministic FSMs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, no. 11, 1997, pp. 1311–1322.

[14] J. Kella, "Sequential machine identification," IEEE Transactions on Computers, 100(3), 1971, pp. 332-338.

[15] G. L. Luo, G. v. Bochmann, A. Petrenko, "Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method", IEEE Transactions on Software Engineering, 20(2), 1994, pp. 149–161.

[16] R. Milner, A calculus of communicating systems, Springer Verlag, 1980.

[17] O. Niese, An integrated approach to testing complex systems. Ph.D. thesis, Dort-mund University of Technology, 2003.

[18] J. Oncina and P. Garcia, "Identifying regular languages in polynomial time" In Advances in Structural and Syntactic Pattern Recognition. Series in Machine Perception and Artificial Intelligence, vol. 5, 1992, pp. 99–108.

[19] A. L. Oliveira and J. Silva, "Efficient algorithms for the inference of minimum size DFAs," Machine Learning 44, no.1, 2001, pp. 93-119.

[20] W. Pacharoen, T. Aoki, P. Bhattarakosol, A. Surarerks, "Active learning of nondeterministic finite state machines," Mathematical Problems in Engineering, vol. 2013, 2013, pp. 1-11.

[21] D. Peled, M. Y. Vardi, M. Yannakakis, "Black-box checking," In Formal Methods for Protocol Engineering and Distributed Systems. FORTE/PSTV, Kluwer, 1999, pp. 225‑240.

[22] A. Petrenko, F. Avellaneda, R. Groz, C. Oriat, "From passive to active FSM inference via checking sequence construction," In Proceedings of the IFIP International Conference on Testing Software and Systems. Springer, 2017, pp. 126-141.

[23] A. Petrenko, N. Yevtushenko, A. Lebedev, A. Das, "Nondeterministic State Machines in Protocol Conformance Testing," Protocol Test Systems, 1993, pp. 363-378.

[24] A. Petrenko and N. Yevtushenko, "Conformance tests as checking experiments for partial nondeterministic FSM," In Proceedings of the 5th International Workshop on Formal Approaches to Testing of Software, LNCS 3997, 2005, pp. 118-133.

[25] A. Petrenko and N. Yevtushenko, "Adaptive testing of deterministic implementations specified by nondeterministic FSMs," In Proceedings of the 23d IFIP International Conference on Testing Software and Systems, LNCS 7019, 2011, pp. 162-178.

[26] A. Petrenko and N. Yevtushenko, "Refining specifications in adaptive testing of nondeterministic finite state machines," In Vestnik Tomskogo gosudarstvennogo universiteta, 1(6), 2009, pp. 99-114.

[27] A. Petrenko and N. Yevtushenko, "Adaptive testing of nondeterministic systems with FSM," In Proceedings of the IEEE 15th International Symposium on High-Assurance Systems Engineering, 2014, pp. 224-228.

[28] A. Petrenko, N. Yevtushenko, G. v. Bochmann, "Testing deterministic implementations from their nondeterministic specifications," In Proceedings of the IFIP Ninth International Workshop on Testing of Communicating Systems, 1996, pp. 125-140.

[29] M. Shahbaz and R. Groz, "Inferring Mealy machines," In Proceedings of the 2nd World Congress on Formal Methods, Springer, 2009, pp. 207–222.

[30] B. Steffen, F. Howar, M. Merten, "Introduction to active automata learning from a practical perspective," In International School on Formal

Methods for the Design of Computer, Communication and Software Systems. Springer, (2011) pp. 256-296.

[31] N. Walkinshaw, B. Lambeau, C. Damas, K. Bogdanov, P. Dupont, "STAMINA: a competition to encourage the development and assessment of software model inference techniques," Empirical software engineering, 18(4), 2013, pp.791-824.

[32] F. Zhang and T. Cheung, "Optimal transfer trees and distinguishing trees for testing observable nondeterministic finite-state machines," IEEE Transactions on Software Engineering, 29(1), 2003, pp. 1-14.

[33] D. Neider, R. Smetsers, F. Vaandrager, and H. Kuppens, "Benchmarks for automata learning and conformance testing", In T. Margaria, K.G. Larsen and S. Graf, editors. Festschrift Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday. LNCS 11200, 2019. To appear.