On the Correspondence between Conformance Testing and Regular Inference

Therese Berg[1], Olga Grinchtein[1], Bengt Jonsson[1],
Martin Leucker[2], Harald Raffelt[3], Bernhard Steffen[3]

[1] Department of Computer Systems, Uppsala University, Sweden
{thereseb, olgag, bengt}@it.uu.se
[2] Institute of Informatics, TU Munich, Germany
leucker@in.tum.de
[3] LS V, Universität Dortmund, Germany
{raffelt, steffen}@cs.uni-dortmund.de

Abstract. Conformance testing for finite state machines and regular inference both aim at identifying the model structure underlying a *black box system* on the basis of a limited set of observations. Whereas the former technique *checks* for equivalence with a *given* conjecture model, the latter techniques addresses the corresponding *synthesis* problem by means of techniques adopted from automata learning. In this paper we establish a common framework to investigate the similarities of these techniques by showing how results in one area can be transferred to results in the other and to explain the reasons for their differences.

1 Introduction

The two areas of conformance testing for finite state machines and regular inference both share the same problem of deducing an unknown finite state machine from a limited set of observations. Whereas the former technique justifies a given conjecture, the latter techniques aims at constructing conjectures by observation. In this paper we establish a common framework to investigate the similarities of these techniques by showing how results in one area can be transferred to results in the other and to explain the reasons for their differences.

The area of testing reactive systems has witnessed significant advances in the last decades. A model problem in this area is that of black-box protocol testing, where one assumes given a finite-state machine specification of the intended behavior of a protocol, and would like to derive a test suite which checks that an implementation conforms to such a specification. There are several techniques for generating test suites that guarantee that an implementation under test (IUT) conforms to a specification, under certain hypotheses [Cho78,FvBK+91,SD88,VCI90].

A more recent line of development concerns checking whether an IUT satisfies certain correctness properties, in the absence of a model or specification. Recent work has employed techniques of automata learning, or regular inference [GPY02,HHNS02,HNS03,PVY99].

Both of the above approaches solve the problem of inferring a finite state machine from observations of its behavior, with different modalities. In conformance testing, the purpose is to check that it is equivalent to a given finite state specification. In automata learning, the purpose is to infer an unknown finite state machine. Both approaches must solve the problem of how to infer a finite state machine from a limited set of observations. Thus, techniques for conformance testing and automata learning must decide what is "enough information" to deduce that an IUT is equivalent to a certain finite state machine. In conformance testing, one goal is to minimize the cost (e.g., the number of observations or their total length) of the observations that are needed to infer that the IUT is equivalent to a specification. In automata learning, one goal has been to understand when "enough information" is obtained to make a conjecture about the structure of an IUT.

Let us make the preceding discussion a little more elaborate. In conformance testing we are given an FSM $\mathcal{M}$, playing the role of a specification, and we want to verify that the IUT is equivalent to $\mathcal{M}$. We construct a test suite with the property that any FSM $\mathcal{A}$ which passes the test suite is equivalent to $\mathcal{M}$. Of course, this can not be achieved unless some additional assumptions about $\mathcal{A}$ are introduced. A common such assumption is that $\mathcal{A}$ has at most as many states as $\mathcal{M}$. We will then say that the test suite is a conformance test suite for $\mathcal{M}$ under these assumptions.

In Automata Learning, we are given a set of observations generated by a test suite or a set of queries, and want to construct an FSM which is an "as good as possible explanation" of the observations, hopefully being close to the structure of the actual IUT. Since there are an infinite number of such FSMs, we should also here add assumptions. Typical assumptions are of the form to give an upper bound the number of states, or to ask for an automaton with a minimum number of states; note that there may be several such automata.

The problem of constructing conformant finite automata was studied by many people [Ang81,Ang87,BDG97,Gol67,Gol78,OG92] and others. Several of these works present conditions on observations that allow a unique minimum automaton to be constructed with modest effort (e.g., in polynomial time).

From the above discussion, it follows that in principle, we can relate conformance testing and automata learning in the following way:

- If the observations form a conformance test suite for an FSM $\mathcal{M}$, given some assumptions, then under the same assumptions we can infer the FSM $\mathcal{M}$ from these observations using automata learning techniques.
- If the FSM $\mathcal{A}$ is inferred from the observations under some assumptions, and furthermore $\mathcal{A}$ is the only such automaton, then under the same assumptions the observations form a conformance test suite for $\mathcal{A}$.

The above statements are rather general, and "kind of obvious". In this paper, we shall compare results in these two areas, and make the link between these two areas explicit. One goal is to relate existing techniques for conformance testing and automata learning by showing that they use very similar concepts of "enough information" in order to infer the structure of an IUT. We will also make a comparison of the difference in complexity between the two approaches in different settings.

From a different point of view, one can understand our contribution as clarifying the following question: What is the core information of an automaton in terms of observations/traces needed to identify it uniquely? We do this in the framework of conformance testing as well as in the framework of learning and show that both domains (nearly) identify the same type of information.

For our comparison, we must bridge several differences in the models typically used. In conformance testing, the most common model is the Mealy machine, which generates output on each transition. In automata learning, the most common model is deterministic finite automata (DFA), which merely accept or reject a given input string. We therefore define a unifying notion of finite state machine which has an abstract notion of "output" in response to a received sequence of input symbols. This notion can be instantiated to Mealy machines by letting the output be the sequence of symbols generated in response to the input, to DFAs outputting a verdict "accepted" or "not accepted".

An important vehicle in the comparison is a general theorem, which shows that under certain conditions on a set of observations, a small finite state machine that satisfies these observations must have a certain structure.

Related Work. The relationship between machine learning and conformance testing was observed by Lee and Yannakakis [LY96, p. 1118], who stated that Angluin's algorithm can be used for fault detection. Note that [LY96] employ learning techniques for conformance testing while we study their similarities. [LY96] also suggested as an interesting subject of study the relationship between conformance testing without reset (surveyed in [LY96]), and corresponding work on machine learning by Rivest and Schapire [RS93].

Organization of this Paper In the next section, we define our model of Finite State Machines, aiming to unify Mealy machines, DFAs, and some other models. In Section 3, we state a general theorem which shows how a set of observations limits the set of machines that may be inferred from it. Section 4 describes some existing techniques for deriving conformance test suites, and Section 5 describes some existing techniques for learning automata from observations. Results for these methods are shown to follow from the general theorem in Section 3. The techniques of these two sections are thereafter related in Section 6.

2 Preliminaries

We will first define two variants of finite state machines: Mealy machines, commonly used in the conformance testing literature, and finite automata, commonly used in automata learning literature. They differ in how they respond to input sequences: Mealy machines produce an output symbol in response to each received input symbol, whereas finite automata merely accept or reject a given input string. We will define a unifying more general model of finite state machines, that produce a more abstractly defined form of output, which can be specialized to both Mealy machines and finite automata.

We assume a finite set Σ of *input symbols*, usually denoted by $a, b, a_1, a_2, \ldots$. Elements of Σ^* are (input) *strings* or *words*. Given $u, v \in \Sigma^*$, u is said to be a *prefix* of v if $v = uw$ for some $w \in \Sigma^*$.

Mealy Machines A *Mealy machine* over Σ is a tuple $M = \langle O, Q, q_0, \delta, \lambda \rangle$ where O is a finite nonempty set of *output symbols*, Q is a finite nonempty set of *states*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to Q$ is the *state transition function*, and $\lambda : Q \times \Sigma \to O$ is the *output function*.

An intuitive interpretation of a Mealy machine is as follows. At any point in time, the machine is in one state $q \in Q$. It is possible to give inputs to the machine, by applying an input symbol a. The machine responds by producing an output symbol $\lambda(q, a)$ and transforming itself to the new state $\delta(q, a)$. We can depict Mealy machines as directed labeled graphs, where Q is the set of vertices. For each state $q \in Q$ and input symbol $a \in \Sigma$, there is an edge from q to $\delta(q, a)$ labeled by "a/b", where b is the output symbol $\lambda(q, a)$. See Figure 1 for an example of a Mealy machine. Note that the letters a and b are used in two ways. In the text they are metasymbols denoting arbitrary input and output symbols, whereas in examples they denote specific input or output symbols.

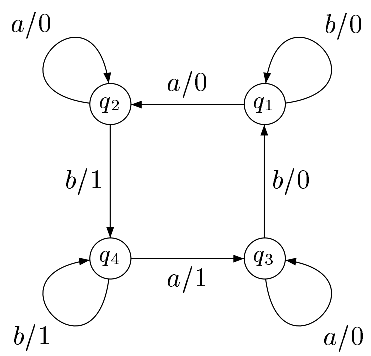

Fig. 1. A Mealy machine with states $Q = \{q_1, q_2, q_3, q_4\}$, input symbols $I = \Sigma = \{a, b\}$, and output symbols $O = \{0, 1\}$.

Applying an input sequence $u = a_1 a_2 \cdots a_k \in \Sigma^*$ starting in a state q_1 takes the machine successively to a sequence of states $q_2, q_3, \ldots, q_{k+1}$, denoted $\delta(q_i, u)$, where $q_{i+1} = \delta(q_i, a_i)$ for $i = 1, \cdots, k$, and produces a sequence of output symbols $b_1 b_2 \cdots b_k \in O^*$, where $b_i = \lambda(q_i, a_i)$ for $i = 1, \cdots, k$. We extend the transition and output functions from input symbols to sequences of input symbols, by defining $\delta(q_1, u) = q_{k+1}$ and $\lambda(q_1, u) = b_1 b_2 \cdots b_k$. A more precise recursive definition is as follows:

$$\begin{array}{ll} \delta(q, \varepsilon) \;\; = q & \qquad \lambda(q, \varepsilon) \;\; = \varepsilon \\ \delta(q, ua) = \delta(\delta(q, u), a) & \qquad \lambda(q, ua) = \lambda(q, u)\lambda(\delta(q, u), a) \end{array}$$

Finite Automata. A *deterministic finite automaton* (*DFA*) over Σ is a structure $\mathcal{A} = (Q, \delta, q_0, F)$ where Q is a non-empty finite set of *states*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to Q$ is the *transition function*, and $F \subseteq Q$ is the set of *accepting states*.

Just as for Mealy machines, we extend the transition function from input symbols to sequences of input symbols, by defining

$$\begin{aligned} \delta(q, \varepsilon) &= q \\ \delta(q, ua) &= \delta(\delta(q, u), a) \end{aligned}$$

An input string u is *accepted* iff $\delta(q_0, u) \in F$. The *language* accepted by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of accepted input strings.

Unifying Formalism. In order to unify the two above types of state machines, we define a more abstract notion of output produced by a finite state machine. Let an *output domain* be a semi-group $\mathcal{D}$ equipped with an associative binary operation, which we denote by juxtaposition. The intended intuition is that an FSM when inputing a sequence of inputs u outputs an element in $\mathcal{D}$. If the FSM outputs x after inputing u and thereafter outputs y in response to v, then the entire output in response to uv is the element xy.

Definition 1. *A* finite state machine *(FSM) over Σ is a structure $(\mathcal{D}, Q, \delta, q_0, \lambda)$ where $\mathcal{D}$ is an output domain, Q is a non-empty finite set of* states*, $q_0 \in Q$ is the* initial state*, $\delta : Q \times \Sigma \to Q$ is the* transition function*, and $\lambda : Q \times \Sigma^* \to \mathcal{D}$ is an* output function*, which satisfies the following* homomorphism *property:*

- $\lambda(q, uv) = \lambda(q, u)\lambda(\delta(q, u), v)$ *for any $q \in Q$ and $u, v \in \Sigma^*$.* □

By the homomorphism property, it is enough to define the output function for input sequences of length 0 and 1, i.e., to define $\lambda(q, \varepsilon)$ and $\lambda(q, a)$ for $a \in \Sigma$.

In this paper, we will consider only FSMs which are *suffix-observable*, meaning that from only the output $\lambda(q_0, uv)$ produced by applying the input sequence uv, we can uniquely extract the output generated by the suffix v, which we denote by $\lambda(q_0, uv)|_v$, so that $\lambda(q_0, uv)|_v = \lambda(\delta(q_0, u), v)$. For the Mealy machine and DFA models, this assumption trivially holds.

To see how our definition of finite state machines generalizes Mealy machines and finite automata, let us specialize it first to Mealy machines. Here, the output domain is the set O^* with the binary string concatenation operation. The output function is the same as that defined for Mealy machines, lifted to strings.

To specialize to finite automata, let $\mathcal{D}$ be the set $\{+, -\}$, where intuitively $+$ denotes "accept" and $-$ denotes "not accept". The semi-group operation maps a pair of arguments onto the second one, i.e., it can be defined by the following table.

$$\begin{array}{cccc} + + = + & & + - = - \\ - + = + & & - - = - \end{array}$$

The output function of a DFA is defined by

$$\begin{aligned} \lambda(q, \varepsilon) &= \begin{cases} + & \text{if} \quad q \in F \\ - & \text{if} \quad q \notin F \end{cases} \\ \lambda(q, u) &= \lambda(\delta(q, u), \varepsilon) \end{aligned}$$

where the last equality follows from the particular definition of the semi-group operation, which makes the left argument irrelevant.

Looking at the examples of Mealy machines and finite automata, we can identify two special subclasses of FSM, characterized by the forms of their output functions:

- FSMs that generate output only at transitions, where $\lambda(q, \varepsilon)$ is a unit element of $\mathcal{D}$ for any state q. Mealy machines are an example with ε as unit element.
- FSMs that generate output only at the last state, i.e., $\lambda(q, u) = \lambda(\delta(q, u), \varepsilon)$, implying that we only need to specify $\lambda(q, \varepsilon)$ for any state q. An example is DFAs.

3 Characterizing FSMs by Observations

In this section, we provide general definitions and results concerning how FSMs can be uniquely inferred from or characterized by observations or tests. Let $\mathcal{D}$ from now on be a fixed particular output domain.

Let us consider the process of observing or testing a black-box IUT, whose behavior can be represented as an FSM. This consists in applying a set of input sequences to the IUT, whereby the corresponding outputs are observed and recorded. The recorded observations can be represented as a partial *observation function* $\mathcal{O}$ from Σ^* to $\mathcal{D}$, whose domain $Dom(\mathcal{O})$ is the set of input sequences that have been applied to the IUT.

In conformance testing, the observation function should represent a test suite which is obtained from an FSM $\mathcal{M} = (\mathcal{D}, Q, \delta, q_0, \lambda)$ which plays the role of a specification, and a set $\mathcal{I} \subseteq \Sigma^*$ of input sequences. Define $\mathcal{M}|_{\mathcal{I}}$ as the observation function $\mathcal{O}$ with $Dom(\mathcal{O}) = \mathcal{I}$, such that $\mathcal{O}(u)$ is defined and equal to $\lambda(q_0, u)$ whenever $u \in \mathcal{I}$. We say that an FSM $\mathcal{A} = (\mathcal{D}, Q, \delta, q_0, \lambda)$ is *conformant* with an observation function $\mathcal{O}$, denoted $\mathcal{A} \models \mathcal{O}$, if $\mathcal{O}(u) = \lambda(q_0, u)$ whenever $u \in Dom(\mathcal{O})$. We trivially have $\mathcal{M} \models \mathcal{M}|_{\mathcal{I}}$ for any $\mathcal{M}$ and $\mathcal{I}$.

Definition 2. *$\mathcal{O}$ is a* conformance test suite *for $\mathcal{M}$ if any FSM $\mathcal{A}$ with at most as many states as $\mathcal{M}$, such that $\mathcal{A} \models \mathcal{O}$, is isomorphic to $\mathcal{M}$.*

In automata learning, we are given an observation function $\mathcal{O}$, and want to construct an FSM which is an "as good as possible explanation" of the observations, hopefully being close to the actual IUT. An obvious criterion is that the FSM should be conformant with $\mathcal{O}$.[4] Since there are an infinite number of such FSMs, we should also here add assumptions. A natural choice is to ask for an automaton with a minimum number of states. Note that there may be several such automata.

Definition 3. *Let $\mathcal{O}$ be an observation function. We say that the FSM $\mathcal{A}$ is* inferred from *$\mathcal{O}$ if $\mathcal{A} \models \mathcal{O}$ and any other $\mathcal{A}'$ with $\mathcal{A}' \models \mathcal{O}$ has at least as many*

[4] However, not all works on automata learning guarantee to generate conformant FSMs.

states as $\mathcal{A}$*. We say that* $\mathcal{A}$ *is* uniquely inferred from $\mathcal{O}$ *if* $\mathcal{A}$ *is the only such FSM.*

We observe the following propositions,

- if $\mathcal{O}$ is a conformance test suite for $\mathcal{M}$ then $\mathcal{M}$ is uniquely inferred from $\mathcal{O}$, and
- if $\mathcal{A}$ is uniquely inferred from $\mathcal{O}$ then $\mathcal{O}$ is a conformance test suite for $\mathcal{A}$.

By these propositions, both conformance testing and automata learning must in some sense prescribe how to transform an observation function $\mathcal{O}$ into an automaton which is conformant with $\mathcal{O}$. A natural approach is to define an equivalence relation on the prefixes of $Dom(\mathcal{O})$, and let each equivalence class be a state of an automaton. If the equivalence is properly constructed, the transition and output functions can be obtained from $\mathcal{O}$. In general, however, the number of possible equivalences is too large for this to be an efficient procedure. The problem of finding the minimal FSM (i.e., with the smallest number of states) which is conformant with a given observation function is NP-complete [Gol78]. But several works [Gol67,Gol78,Ang81,Ang87,OG92,BDG97] overcome this obstacle by presenting conditions on the observations that allow a unique minimum automaton to be constructed. The conditions exploit the property of suffix-observability, by regarding each input sequence as the concatenation of a prefix and a suffix (possibly in several ways).

So, let the set of observations be given by an *observation structure*, which is a partial function $\mathcal{T}$ from a set $Dom(\mathcal{T}) \subseteq \Sigma^*$ of prefixes, which must include ε. For each $u \in Dom(\mathcal{T})$, $\mathcal{T}(u)$ is a partial function from a set $Dom(\mathcal{T}(u)) \subseteq \Sigma^*$ of suffixes, which must include ε, to $\mathcal{D}$. Intuitively, $\mathcal{T}(u)(v)$, for $v \neq \varepsilon$, is the output produced in response to the suffix v in a situation where the input sequence uv is applied to the IUT. Note that this output can be uniquely extracted by the assumption of suffix-observability. In contrast, we let $\mathcal{T}(u)(\varepsilon)$ be the entire output produced by the IUT in response to the input sequence u. Note that $\mathcal{T}(u)(\varepsilon)$ has a meaning which differs from that of $\mathcal{T}(u)(v)$ for $v \neq \varepsilon$.

An observation structure $\mathcal{T}$ represents the observation function $\mathcal{O}_{\mathcal{T}}$ with

- $Dom(\mathcal{O}_{\mathcal{T}}) = \{uv \ : \ u \in Dom(\mathcal{T}) \text{ and } v \in Dom(\mathcal{T}(u))\}$, and
- $\mathcal{O}_{\mathcal{T}}(uv) = \mathcal{T}(u)(\varepsilon)\ \mathcal{T}(u)(v)$.

Conversely, an observation function $\mathcal{O}$ can, given a set $U \subseteq Dom(\mathcal{O})$, be represented by the observation structure $\mathcal{T}_{\mathcal{O},U}$ with

- $Dom(\mathcal{T}_{\mathcal{O},U}) = U \quad \text{and} \quad Dom(\mathcal{T}_{\mathcal{O},U}(u)) = \{v \ : \ uv \in Dom(\mathcal{O})\}$,
- $\mathcal{T}_{\mathcal{O},U}(u)(\varepsilon) = \mathcal{O}(u)$ for $u \in U$, and
- $\mathcal{T}_{\mathcal{O},U}(u)(v) = \mathcal{O}(uv)|_v$ for $v \neq \varepsilon$, $u \in U$, and $v \in Dom(\mathcal{T}_{\mathcal{O},U}(u))$,

where $\mathcal{O}(uv)|_v$ is the output produced in response to the suffix v, obtained from the result $\mathcal{O}(uv)$ of applying uv to the IUT.

When constructing an automaton from observations, the prefixes in $Dom(\mathcal{T})$ are candidates for representing states of the automaton, whereas the suffixes in the sets $Dom(\mathcal{T}(u))$ are used to determine which prefixes should represent the same state.

Let $\mathcal{T}(u) \approx \mathcal{T}(u')$ denote that for any $v \in (Dom(\mathcal{T}(u)) \cap Dom(\mathcal{T}(u')))$ we have $\mathcal{T}(u)(v) = \mathcal{T}(u')(v)$. Let $\mathcal{T}(u) \subseteq \mathcal{T}(u')$ denote that $Dom(\mathcal{T}(u)) \subseteq Dom(\mathcal{T}(u'))$ and $\mathcal{T}(u) \approx \mathcal{T}(u')$. Let $\mathcal{T}(u) = \mathcal{T}(u')$ denote that $\mathcal{T}(u) \subseteq \mathcal{T}(u')$ and $\mathcal{T}(u') \subseteq \mathcal{T}(u)$.

Define an *access string* of $\mathcal{T}$, to be an input sequence $u \in Dom(\mathcal{T})$ such that $ua \in Dom(\mathcal{T})$ for each $a \in \Sigma$. We say that an equivalence $\equiv$ on $Dom(\mathcal{T})$ is *U-closed* if each equivalence class contains a string in U. We say that an equivalence $\equiv$ on $Dom(\mathcal{T})$ is *U-consistent* if whenever $u \equiv u'$ for $u, u' \in U$ and $ua, u'a \in Dom(\mathcal{T})$ for any $a \in \Sigma$, then $ua \equiv u'a$ and $\mathcal{T}(ua)(\varepsilon)|_a = \mathcal{T}(u'a)(\varepsilon)|_a$.

Definition 4. *Let $\mathcal{T}$ be an observation structure, let U be a set of access strings of $\mathcal{T}$ containing ε. If $\equiv$ is a U-closed and U-consistent equivalence relation on $Dom(\mathcal{T})$, define the automaton $\langle \mathcal{T}, U\rangle / \equiv$ as $(\mathcal{D}, Q, \delta, q_0, \lambda)$, where*

- $Q = Dom(\mathcal{T}) / \equiv$,
- $\delta([u], a) = [ua]$ *for* $u \in U$,
- $q_0 = [\varepsilon]$,
- $\lambda([u], a) = \mathcal{O}_{\mathcal{T}}(u, a)|_a$ *for* $u \in U$. □

We are now ready to state a general theorem that gives constraints on any FSM that is conformant with an observation function.

Theorem 1 (Characterization Theorem). *Let $\mathcal{T}$ be an observation structure, and let U be a set of access strings of $\mathcal{T}$. If the relation $\approx$ on $Dom(\mathcal{T})$ contains a unique maximal equivalence relation $\equiv$, which is U-closed, then (letting n be the number of equivalence classes of $\equiv$)*

1. *any FSM which is conformant with $\mathcal{O}_{\mathcal{T}}$ has at least n states,*
2. *if $\mathcal{A} \models \mathcal{O}_{\mathcal{T}}$ and $\mathcal{A}$ has at most n states, then $\equiv$ is U-consistent, and*
   (a) $\mathcal{A}$ is isomorphic to $\langle \mathcal{T}, U\rangle / \equiv$,
   (b) $\mathcal{T}$ is a conformance test for $\mathcal{A}$
   (c) $\mathcal{A}$ is uniquely inferred from $\mathcal{T}$.

Proof. If $\mathcal{A} = (\mathcal{D}, Q', \delta', q_0', \lambda') \models \mathcal{O}_{\mathcal{T}}$, then each of its states can correspond to at most one equivalence class of $\equiv$, i.e., $u \equiv u'$ if $\delta'(q_0', u) = \delta'(q_0', u')$ for $u, u' \in Dom(\mathcal{T})$. If $\mathcal{A}$ has n states, this correspondence must be exact, and the theorem follows. □

Intuitively, Theorem 1 gives necessary constraints on an FSM that is conformant with the observations represented by an observation structure $\mathcal{T}$. If there is a unique maximal equivalence with n classes, then any conformant automaton has at least n states. In general, there is no guarantee that a conformant FSM with n states actually exists, but if it does, Theorem 1 states that it must be isomorphic to $\langle \mathcal{T}, U\rangle / \equiv$. Later, in Theorem 4 we shall give extra conditions on $\mathcal{T}$ which guarantee the existence of a conformant n-state automaton.

The condition "contains a unique maximal equivalence relation" in Theorem 1 is not so constructive. More concrete sufficient conditions on $\mathcal{T}$ are given by the following proposition.

Proposition 1. *Let $\mathcal{T}$ be an observation structure, and let U be a set of access strings of $\mathcal{T}$. If*

- *$\mathcal{T}(u) \not\approx \mathcal{T}(u')$ for $u, u' \in U$ with $u \neq u'$, and*
- *for each $u \in Dom(\mathcal{T})$ there is a $u' \in U$ with $\mathcal{T}(u) \subseteq \mathcal{T}(u')$,*

then $\approx$ is a unique maximal equivalence on $Dom(\mathcal{T})$, and is closed. □

4 Conformance Testing

In this section, we consider some standard techniques for constructing conformance test suites: the W-method by Vasilevski [Vas73] and Chow [Cho78], an optimization by Fujiwara et al. [FvBK+91] called the partial W-method (or Wp-method), and another optimization described by Lee and Yannakakis [LY96].

Definition 5. *Let $\mathcal{M} = (\mathcal{D}, Q, \delta, q_0, \lambda)$ be an FSM. A set U of input sequences containing ε is called*

- *a* state cover set *if for each state $q \in Q$ there is an input sequence $u \in U$ with $\delta(q_0, u) = q$, i.e., for each state of $\mathcal{M}$, some sequence in U leads to it*
- *a* transition cover set *if whenever $\delta(q, a) = q'$ for some $q, q' \in Q$ and $a \in \Sigma$, there is an input sequence u with $\delta(q_0, u) = q$ such that both $u \in U$ and $ua \in U$.* □

The literature has slight differences in how such sequences can be chosen. For instance, Lee and Yannakakis [LY96] consider state and transition cover sets that are generated by a spanning tree for $\mathcal{M}$.

Say that a sequence $w \in \Sigma^*$ *separates* the states q and q' if $\lambda(q, w) \neq \lambda(q, w')$.

Definition 6. *Let $\mathcal{M} = (\mathcal{D}, Q, \delta, q_0, \lambda)$ be an FSM.*

- *A set W of sequences is a* characterizing set *for $\mathcal{M}$ (or* separating set*) if for each pair $q, q' \in Q$ of states it contains a sequence $w \in W$ which separates q and q'.*
- *A collection $\{W_q\}_{q \in Q}$ of sets of sequences W_q, one for each $q \in Q$, is called*
  - *a* separating family *[LY96] for $\mathcal{M}$ if for each pair $q, q' \in Q$ of states there is a sequence $w \in W_q \cap W_{q'}$ which separates q and q',*
  - *a family of* identification sets *for $\mathcal{M}$ if for each pair $q, q' \in Q$ of states, the set W_q contains a sequence $w \in W_q$ that separates q from q',* □

A separating family is also a family of identification sets, but not vice versa. A family of identification sets can be transformed into a separating family by adding the necessary sequences to the sets. A characterizing set can be thought of as a separating family, where all sets are identical. A characterizing set (and hence also a separating family) exists for every machine that is minimized.

In the following, fix an FSM $\mathcal{M} = (\mathcal{D}, Q, \delta, q_0, \lambda)$. Let

- V be a transition cover set; we denote by $v_{q,a}$ the sequence leading to q such that both $v_{q,a} \in V$ and $v_{q,a}a \in V$,

- U be a state cover set included in V; we denote by u_q the sequence leading to q (i.e., $u_q = v_{q,a}$ for some a),
- W be a characterizing set,
- $\{Z_q\}_{q \in Q}$ be a separating family,
- $\{W_q\}_{q \in Q}$ be a family of identification sets.

Definition 7. *A set $\mathcal{I} \subseteq \Sigma^*$ is called*

- *A* W-set *if it is of form* $\quad VW$,
- *A* Wp-set *if it is of form*

$$U\left(\bigcup_{q \in Q} W_q\right) \quad \cup \quad \bigcup_{q \in Q, a \in \Sigma} v_{q,a} a W_{\delta(q,a)}$$

- *A* Z-set *if it is of form*

$$\bigcup_{q \in Q} v_{q,a} Z_q \quad \cup \quad \bigcup_{q \in Q, a \in \Sigma} v_{q,a} a Z_{\delta(q,a)}$$

□

Theorem 2 (Conformance Test Suites). *Let $\mathcal{M} = \langle Q, \delta, q_0, \lambda \rangle$ be an FSM and let $\mathcal{I} \subseteq \Sigma^*$ be a W-set, a Wp-set, or a Z-set. Then the observation function $\mathcal{M}|_{\mathcal{I}}$ is a conformance test suite for $\mathcal{M}$.*

Proof. We consider the case of Wp-set; the other cases are analogous. Let $\mathcal{T}$ be the observation structure defined by $Dom(\mathcal{T}) = U \cup \{v_{q,a}a \ : \ q \in Q, a \in \Sigma\}$, where $Dom(\mathcal{T}(u_q)) = \bigcup_{q \in Q} W_q$ for $u_q \in U$ and $Dom(\mathcal{T}(v_{q,a}a)) = W_{\delta(q,a)}$ for $v_{q,a} \in \{v_{q,a} \ : \ q \in Q, a \in \Sigma\} \setminus U$, such that $\mathcal{O}_{\mathcal{T}} = \mathcal{M}|_{\mathcal{I}}$. Since the observation structure is derived from $\mathcal{M}$, and by the properties of identification sets, it follows that U is a set of access strings such that the conditions in Proposition 1 are satisfied. Hence the conclusions of Theorem 1 hold, from which the result follows. □

The W-method by Vasilevski [Vas73] and Chow [Cho78] uses W-sets. The Wp-method by Fujiwara et al. [FvBK+91] optimizes by using (hopefully smaller) identifications sets to reduce the size of the test suite; another optimization, using separating families (here defined using what we call Z-sets) is described by Lee and Yannakakis [LY96]. Since in the worst case, each identification set W_q has the same cardinality as the characterizing set W, upper bounds on sizes of the test suite generated by the three methods are the same: $O(n^2 \, |\Sigma|)$.

5 Automata Learning

We here briefly review some techniques of Automata Learning. The techniques reviewed here work by making queries about the output of an IUT in response to a set of input sequences, and recording the results in what can be represented as an observation structure $\mathcal{T}$. When $\mathcal{T}$ has been developed so that it satisfies certain properties, then an automaton is conjectured from $\mathcal{T}$. This conjecture is then compared by other means (idealized by a so-called "equivalence query") with the IUT. If the conjecture is equivalent to the IUT, the learning process stops, otherwise the equivalence query returns an input sequence on which the conjecture and the IUT disagree, and the learning process continues. It is desirable that each hypothesis has strictly more states than the previous one, in order that the process monotonically converges to a correct conjecture in reasonable time. This can be ensured if the properties required for making a hypothesis ensure that only one automaton can be inferred from $\mathcal{T}$.

In this section, we present conditions on $\mathcal{T}$ that are defined by the L^* algorithm of Angluin [Ang87] using *observation tables*, and the *observation packs* defined by Balcázar et al. [BDG97].

Let $\mathcal{T}$ be an observation structure. Two situations are particularly interesting and separately well-studied in the literature

Definition 8. *Let $\mathcal{T}$ be an observation structure, where $Dom(\mathcal{T}) = U \cup U\Sigma$ for a set U of access strings. $\mathcal{T}$ is an*

- **observation table** *if $Dom(\mathcal{T})$ is prefix-closed, and all $Dom(\mathcal{T}(u))$ for $u \in Dom(\mathcal{T})$ are equal and suffix-closed.*
- **observation pack** *if $\varepsilon \in U$, and*
  - *$\mathcal{T}(u) \not\approx \mathcal{T}(u')$ for $u, u' \in U$ with $u \neq u'$, and*
  - *for each $u \in Dom(\mathcal{T})$ there is a $u' \in U$ with $\mathcal{T}(u) = \mathcal{T}(u')$.* □

Based on these definitions, we obtain:

Theorem 3 (Uniqueness Theorem). *Let $\mathcal{T}$ be an observation structure with U as in Definition 8. If $\mathcal{T}$ is either*

- *an observation table, where $\approx$ is U-closed and U-consistent, or*
- *an observation pack, where $\approx$ is U-closed,*

then the relation $\approx$ on $Dom(\mathcal{T})$ is an equivalence relation. Let n be the number of equivalence classes of $\equiv$. Then any automaton $\mathcal{A}$ with at most n states, which is conformant with $\mathcal{O}_{\mathcal{T}}$, is isomorphic to $\langle \mathcal{T}, U \rangle / \equiv$.

Proof. It follows from Definition 8 that $\approx$ is an equivalence relation. The rest follows immediately from the Characterization Theorem 1 □

Please note that the Uniqueness Theorem does not guarantee the existence of a conformant automaton with n states. However, for observation tables we can give such a guarantee.

Theorem 4 (Existence Theorem). *Let $\mathcal{T}$ be an observation table with U as in Definition 8, where $\approx$ is U-closed and U-consistent. Then $\langle \mathcal{T}, U \rangle / \approx \models \mathcal{O}_{\mathcal{T}}$.*

This theorem is proved in [Gol78,Ang87]. Our Existence Theorem is a straightforward generalization.

6 Relating Testing and Learning Techniques

Conformance testing and learning are both concerned with establishing a relationship between a formal model and a black box system. Both techniques work by constructing a particular set of tests serving for the observation of the black box system. These conceptual similarities should be clear from the previous sections.

In fact, this similarity even reaches down to the level of technical detail of observation structures:

From Automata Learning to Conformance Testing

- Let $\mathcal{T}$ be an observation table with U as in Definition 8, such that $\approx$ is U-closed and U-consistent. Let W denote $Dom(\mathcal{T}(u))$ for some $u \in U$ (the choice of u is irrelevant by Definition 8). If $\mathcal{M}$ is isomorphic to $\langle \mathcal{T}, U \rangle / \approx$, then the set $(U \cup U\Sigma)W$ is a W-set for $\mathcal{M}$.
- Let $\mathcal{T}$ be an observation pack with U as in Definition 8, such that $\approx$ is U-closed. If $\mathcal{M} = \langle \mathcal{T}, U \rangle / \approx$ is conformant with $\mathcal{O}_{\mathcal{T}}$, then the set

$$\bigcup_{u \in U} u \; Dom(\mathcal{T}(u)) \quad \cup \quad \bigcup_{u \in U, a \in \Sigma} ua \; Dom(\mathcal{T}(ua))$$

  is a Z-set for $\mathcal{M}$.

From Conformance Testing to Automata Learning Let $\mathcal{M} = \langle Q, \delta, q_0, \lambda \rangle$ be an FSM and let U be a state cover set of $\mathcal{M}$, and $U\Sigma$ the corresponding transition cover set.

- If U is prefix-closed, and W is a suffix-closed characterizing set, then the observation structure $\mathcal{T}$ defined by $Dom(\mathcal{T}) = U \cup U\Sigma$ and $Dom(\mathcal{T}(u)) = W$ for any $u \in U$, with $\mathcal{O}_{\mathcal{T}} = \mathcal{M}|_{(U \cup U\Sigma)W}$, is an observation table where $\approx$ is U-closed and U-consistent, such that $\mathcal{M}$ is isomorphic to $\langle \mathcal{T}, U \rangle / \approx$ and $\mathcal{M} \models \mathcal{O}_{\mathcal{T}}$.
- If $\{Z_q\}_{q \in Q}$ is a separating family and

$$\mathcal{I} \;=\; \bigcup_{q \in Q} u_q Z_q \quad \cup \quad \bigcup_{q \in Q, a \in \Sigma} u_q a Z_{\delta(q,a)}$$

  is a corresponding Z-set, then the observation structure $\mathcal{T}$ defined by $Dom(\mathcal{T}) = U \cup U\Sigma$ and $Dom(\mathcal{T}(u)) = Z_{\delta(q_0,u)}$ for $u \in U \cup U\Sigma$, with $\mathcal{O}_{\mathcal{T}} = \mathcal{M}|_I$, is an observation pack where $\approx$ is U-closed, such that $\mathcal{M}$ is isomorphic to $\langle \mathcal{T}, U \rangle / \approx$ and $\mathcal{M} \models \mathcal{O}_{\mathcal{T}}$.

Thus the observation table technique is strongly related to the W-method and the observation pack technique to the conformance testing technique described in [LY96].

However, there is also an intrinsic conceptual difference:

– conformance testing solves a *checking* problem: given a model and a black box system, it checks for conformance of the two. This allows us to systematically construct the tests from the given model, and
– learning solves a *synthesis* problem: given a black box, it synthesizes a model on the basis of a systematic experimentation process. The tests used here must be generated online in parallel with the model synthesis.

This conceptual difference becomes particularly clear under the often used assumption that the number of states of the black box system is known to be at most the number of states of the model n. In this case, we have:

The construction of a conformance test suite is a systematic and rather efficient process ($O\left(n^2\,|\Sigma|\right)$) that extracts sufficiently many tests from the model to characterize the model up to isomorphism.

The process of generating tests during the learning process is much more involved, as there is no model for orientation. Thus we are essentially left with a systematic search problem. Angluins assumption of an equivalence oracle, which provides a (minimal) counter example in case of failure, draws a nice dividing line between the efficient and expensive part:

– *Complexity relative to the equivalence oracle:* Angluins observation table only grows polynomially in the size of the resulting model. The original proof for $O\left(n^3\,|\Sigma|\right)$ can straightforwardly be extended to FSMs. Thus there is only an additional factor n in comparison to the conformance test suite generation. This factor is due to the fact that one must maintain many strings as potential state representatives as their redundancy can only be decided after the learning process has terminated.
– *Complexity for realizing/approximating the equivalence oracle:* In general it is impossible to implement an equivalence oracle, and even if the size of the black box system is known the problem is exponential in this size. Thus the equivalence oracle is the true bottleneck of automata learning. However, also here are similarities to conformance testing: a conformance test suite capturing IUTs which may have k states more than the model also grows exponentially in k. In fact, one could consider conformance testing of this more general kind as a good approximation of the equivalence oracle.[5]

7 Discussion

In this paper, we have established a common framework for investigating the similarities of conformance testing and automata learning by showing how results in one area can be transferred to results in the other and to explain the reasons for their differences. Both techniques aim at identifying the model structure underlying a *black box system* on the basis of a limited set of observations. Whereas the former technique aims at *checking* for equivalence with a *given*

[5] Note, this is usually the line where the interplay of learning and conformance testing is mentioned.

conjecture model, the latter techniques addresses the corresponding *synthesis* problem: given a system, it aims at inferring a corresponding model. Our unified framework makes it possible to directly transfer results between these two communities or, more concretely, to build tools that easily specialize to tools for conformance testing or automata learning, respectively.

Beyond this rather technical match, our contribution also directly addresses the following question: What is the essential information about an automaton in terms of observations/traces? The similarity of the corresponding characterizations in the two domains mark them as a 'natural' choice. And, in fact, the state of the art here does not seem to leave much room for further optimizations, at least for the general setting. In particular when considering automata learning this means that major performance gains, a necessary precondition for a significant practical impact of this technology, are only possible for restricted scenarios. In other words, learning will only scale to practically relevant system scenarios, if its is possible to steer the learning process on the basis of complementary knowledge, e.g. about the structure of the black box systems, its intended behavior or certain other behavioral characteristics like input enabledness or output determinism. Our first experiments [HNS03,SH03] indicate the power of exploiting such knowledge, which does not only reduce the learning effort, but also the size of the model representation. We are currently investigating, how similar considerations may also be used to minimize conformance test suites.

References

[Ang81] Dana Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51(1):76–87, 1981.

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

[BDG97] José L. Balcázar, Josep Díaz, and Ricard Gavaldá. Algorithms for learning finite automata from queries: A unified view. In *Advances in Algorithms, Languages, and Complexity*, pages 53–72. Kluwer, 1997.

[Cho78] Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans. on Software Engineering*, 4(3):178–187, May 1978. Special collection based on COMPSAC.

[FvBK+91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Trans. on Software Engineering*, 17(6):591–603, June 1991.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[Gol78] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[GPY02] A. Groce, D. Peled, and M. Yannakakis. Adaptive model checking. In J.-P. Katoen and P. Stevens, editors, *Proc. TACAS '02, 8^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 357–370. Springer Verlag, 2002.

[HHNS02] A. Hagerer, H. Hungar, O. Niese, and B. Steffen. Model generation by moderated regular extrapolation. In R.-D. Kutsche and H. Weber, editors, *Proc. FASE '02, 5^{th} Int. Conf. on Fundamental Approaches to Software Engineering*, volume 2306 of *Lecture Notes in Computer Science*, pages 80–95. Springer Verlag, 2002.

[HNS03] Hardi Hungar, Oliver Niese, and Bernhard Steffen. Domain-specific optimization in automata learning. In *Proc. 15^{th} Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 315–327, 2003.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines – a survey. *Proc. IEEE*, 84(8):1090–1126, 1996.

[OG92] J. Oncina and P. García. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, volume 1 of *Series in Machine Perception and Artificial Intelligence*, pages 49–61. World Scientific, 1992.

[PVY99] Doron Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. In Jianping Wu, Samuel T. Chanson, and Qiang Gao, editors, *Formal Methods for Protocol Engineering and Distributed Systems, FORTE/PSTV*, pages 225–240, Beijing, China, 1999. Kluwer.

[RS93] R.L. Rivest and R.E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103:299–347, 1993.

[SD88] Krishan Sabnani and Anton Dahbura. A protocol test generation procedure. *Computer Networks and ISDN Systems*, 15(4):285–297, September 1988.

[SH03] Bernhard Steffen and Hardi Hungar. Behavior-based model construction. In *VMCAI 2003: Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 2575 of *Lecture Notes in Computer Science*, pages 5–19. Springer-Verlag, 2003.

[Vas73] M. P. Vasilevski. Failure diagnosis of automata. *Cybernetic*, 9(4):653–665, 1973.

[VCI90] S.T. Vuong, W.Y.L. Chan, and M.R. Ito. The UIOv-method for protocol test sequence generation. In *Proc. 2nd Int. Workshop on Protocol Test Systems*, pages 161–176. North-Holland, 1990.