Testing and Model-Checking Techniques for Diagnosis

Maxim Gromov[1] and Tim A.C. Willemse[2]

[1] Institute for Computing and Information Sciences (ICIS)
Radboud University Nijmegen – The Netherlands, email: m.gromov@cs.ru.nl
[2] Design and Analysis of Systems Group,
Eindhoven University of Technology – The Netherlands, email: t.a.c.willemse@tue.nl

Abstract. Black-box testing is a popular technique for assessing the quality of a system. However, in case of a test failure, only little information is available to identify the root-cause of the test failure. In such cases, additional diagnostic tests may help. We present techniques and a methodology for efficiently conducting diagnostic tests based on explicit fault models. For this, we rely on *Model-Based Testing* techniques for *Labelled Transition Systems*. Our techniques rely on, and exploit differences in outputs (or inputs) in fault models, respectively. We characterise the underlying concepts for our techniques both in terms of mathematics and in terms of the modal μ-calculus, which is a powerful *temporal logic*. The latter characterisations permit the use of efficient, off-the-shelf model checking techniques, leading to provably correct algorithms and pseudo decision procedures for diagnostic testing.

1 Introduction

Testing has proved to be a much-used technique for validating a systems behaviour, but in itself it is a quite labour-intensive job. Formal approaches to testing, collectively known as *Model-Based Testing*, have been touted as effective means for reducing the required effort of testing by allowing for automation of many of its aspects. However, MBT provides only a partial answer to the validation problem, as in most cases its automation ceases at the point where a test failure has been detected; pinpointing the root-cause of the test failure remains a laborious and time-consuming task. Finding this root-cause is known as the *diagnosis* problem, and it is tightly linked to testing.

Formal approaches to the diagnosis problem rely on the use of models of the system-under-diagnosis, and are often referred to as *Model-Based Diagnosis* techniques. While MBD has been studied extensively in the formal domain of *Finite State Machines* (see e.g. [3, 4, 6, 11]), the topic is little studied in the setting of Labelled Transition Systems. The advantage of many LTS-based theories over FSM-based theories is that the assumptions under which they operate are more liberal, which makes them easier to apply in practice. In this paper, we advocate an LTS-based MBD approach for non-deterministic, reactive systems. The techniques that we put forward in this paper operate under the liberal LTS-based testing hypothesis of **ioco**-based testing [13]; our methods rely on explicit models describing the faulty behaviour, henceforth referred to as *fault models*.

The problem that we consider consists of identifying "correct" fault models among a given (but large) set of possible fault models. By "correct", we understand that no

evidence of a mismatch between the malfunctioning system and the fault model can be found. This can be asserted by e.g. testing. Note that even though this problem is readily solved by testing the malfunctioning system against each fault model separately, this is a daunting task which is quite expensive in terms of resources, even when fully automated. The main contributions of this paper are twofold:

1. inspired by classical FSM-based diagnosis approaches we present diagnostic concepts and techniques to make the fault model selection process more effective in an LTS-based setting. In particular, we adopt and modify the notion of *distinguishability* (see e.g. [11]) from FSMs to fit the framework of LTSs. Secondly, we introduce a novel notion, called *orthogonality* which helps to direct test efforts onto isolated aspects of fault models. Both notions are studied in the setting of **ioco**-based testing.
2. we link our diagnostic concepts and techniques to *model-checking* problems. This gives rise to effective and provably correct automation of our approach, and leads to a better understanding of all involved concepts.

Note that the problem of constructing the set of fault models is left outside the scope of this paper; in general, there are an infinite number of fault models per implementation. While this is indeed a very challenging problem, for the time being, we assume that these have been obtained by manually modifying e.g. a given specification, based on modifications of subcomponents of the specifications. Such modifications can be driven by the observed non-conformance between the specification and the implementation, but also fault injection is a good strategy.

Related work. In [8], Jéron *et al* paraphrase the diagnosis problem for discrete event systems (modelled by LTSs), as the problem of finding whether an observation of a system contains forbidden sequences of actions. Their approach takes a description of the structure of a system as input; the sequences of forbidden actions are specified using patterns. They subsequently propose algorithms for, a.o., synthesising a diagnoser which tells whether or not a pattern occurred in the system. A variation on this approach is given in [10], in which all actions are unobservable except for special "warning" actions. The problem that is solved is finding explanations for the observations of observed warning actions. Both works view the diagnosis problem as a supervisory problem.

Apart from the above mentioned works in the setting of LTSs, there is ample literature on diagnosis based on FSMs. Guo *et al*, in [6] focus on heuristics for fault diagnosis, which helps to reduce the cost of fault isolation and identification. El-Fakih *et al* [4] define a diagnostic algorithm for nets of FSMs, and in [3] these techniques are extended; the effectiveness of (a minor modification of) that algorithm is assessed in [5]. Most FSM-based approaches consist of two steps, the first step being the generation of a number of candidate fault models (often referred to as *candidates*), and the second step being a selection of appropriate candidates. The first step relies on strict assumptions, which in general are not met in an LTS-based setting.

In [12] the emphasis is on diagnosing non-reactive systems, mostly hardware, although their techniques have also been applied to software. Based on the topology of a system, explanations for a system's malfunctioning are computed and ranked according to likeliness. The techniques underlying the diagnosis are based on propositional logic and satisfiability solvers.

Structure of the paper. In Section 2 we repeat the **ioco**-based testing theory and the modal μ-calculus [2], the latter being our carrier for linking diagnosis to the problem of model-checking. The basic concepts for diagnosis, and their link to model-checking problems is established in Section 3. In Section 4, we provide an algorithm and a semi-decision procedure that implement the techniques and concepts of Section 3.

2 Background

In this section, we briefly recall the testing theory **ioco** as defined in [13]. The **ioco** framework and its associated testing hypotheses serve as the basic setting for our diagnosis techniques. Furthermore, we introduce the modal μ-calculus [2], which is a modal logic that we will use as a tool for characterising our diagnostic techniques.

Definition 1. *A* Labelled Transition System *(LTS) with inputs $\mathcal{A}ct_I$ and outputs $\mathcal{A}ct_U$ is a quintuple $\langle S, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow, \overline{s}\rangle$, where S is a non-empty set of states with initial state $\overline{s} \in S$; $\mathcal{A}ct_I$ and $\mathcal{A}ct_U$ are disjoint finite sets representing the set of input actions and output actions, respectively. We denote their union by $\mathcal{A}ct$. As usual, $\tau \notin \mathcal{A}ct$ denotes an internal non-observable action, and we write $\mathcal{A}ct_\tau$ for $\mathcal{A}ct \cup \{\tau\}$. The relation $\rightarrow \subseteq S \times \mathcal{A}ct_\tau \times S$ is the transition relation.*

Let $L = \langle S, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow, \overline{s}\rangle$ be a fixed LTS. Let $s, s', \ldots$ range over S. Throughout this paper, we use the following conventions: for all actions a, we write $s \xrightarrow{a} s'$ iff $(s, a, s') \in \rightarrow$, and $s \xrightarrow{a}$ iff for all $s' \in S$, *not* $s \xrightarrow{a} s'$.

Ioco-*based testing theory.* The notion of *quiescence* is added to an LTS as follows: a state s is quiescent — notation $\delta(s)$ — iff $s \xrightarrow{\tau}$ and for all $a \in \mathcal{A}ct_U$, $s \xrightarrow{a}$. Informally, a quiescent state is a state that is "stable" (it does not allow for internal activity) and it refuses to provide outputs. Let $\delta \notin \mathcal{A}ct_\tau$ be a fresh label representing the possibility to observe quiescence; $\mathcal{A}ct_\delta$ abbreviates $\mathcal{A}ct \cup \{\delta\}$. Let $\sigma, \sigma', \ldots$ range over $\mathcal{A}ct_\delta^*$, actions a range over $\mathcal{A}ct_\delta$, and $S', S'', \ldots \subseteq S$. We generalise the transition relation $\rightarrow$ to $\Longrightarrow \subseteq S \times \mathcal{A}ct_\delta^* \times S$, and write $s \stackrel{\sigma}{\Longrightarrow} s'$ iff $(s, \sigma, s') \in \Longrightarrow$. We define $\Longrightarrow$ as the smallest relation satisfying the following four rules:

$$\frac{}{s \stackrel{\epsilon}{\Longrightarrow} s} \qquad \frac{s \stackrel{\sigma}{\Longrightarrow} s' \quad s' \xrightarrow{\tau} s''}{s \stackrel{\sigma}{\Longrightarrow} s''} \qquad \frac{s \stackrel{\sigma}{\Longrightarrow} s' \quad s' \xrightarrow{a} s''}{s \stackrel{\sigma \cdot a}{\Longrightarrow} s''} \qquad \frac{s \stackrel{\sigma}{\Longrightarrow} s' \quad \delta(s')}{s \stackrel{\sigma \cdot \delta}{\Longrightarrow} s'}$$

Analogously to $\rightarrow$, we write $s \stackrel{\sigma}{\Longrightarrow}$ for $s \stackrel{\sigma}{\Longrightarrow} s'$ for some s'. For ease of use, we introduce the following functions and operators.

1. $[s]_\sigma \stackrel{\text{def}}{=} \{s' \in S \mid s \stackrel{\sigma}{\Longrightarrow} s'\}$; generalised: $[S']_\sigma \stackrel{\text{def}}{=} \bigcup_{s \in S'} [s]_\sigma$;
2. $\mathbf{out}(s) \stackrel{\text{def}}{=} \{a \in \mathcal{A}ct_U \mid s \xrightarrow{a}\} \cup \{\delta \mid \delta(s)\}$; generalised: $\mathbf{out}(S') \stackrel{\text{def}}{=} \bigcup_{s \in S'} \mathbf{out}(s)$,
3. *s-traces*$(s) \stackrel{\text{def}}{=} \{\sigma \in \mathcal{A}ct_\delta^* \mid s \stackrel{\sigma}{\Longrightarrow}\}$,
4. *traces*$(s) \stackrel{\text{def}}{=}$ *s-traces*$(s) \cap \mathcal{A}ct^*$,
5. $\mathbf{der}(s) \stackrel{\text{def}}{=} \bigcup_{\sigma \in \mathcal{A}ct^*} [s]_\sigma$; generalised: $\mathbf{der}(S') \stackrel{\text{def}}{=} \bigcup_{s \in S'} \mathbf{der}(s)$.

Note 1. Our notation $[S']_\sigma$ is a deviation from the standard **ioco**-notation, where $[S']_\sigma$ is written as S' **after** σ. While we are not in favour of changing common notation, our main motivation for using our notation is brevity in definitions, theorems and algorithms, in support of readability.

Definition 2. *We say that:*

- L *is* image finite *if for all* $\sigma \in \mathcal{A}ct^*$, $[\overline{s}]_\sigma$ *is finite,*
- L *is* deterministic *if for all* $s' \in S$ *and all* $\sigma \in \mathcal{A}ct^*$, $|[s']_\sigma| \leq 1$,
- L *is* strongly converging *if there is no infinite sequence of* τ *transitions,*
- *A state* $s \in S$ *is* input-enabled *if for all* $s' \in \mathbf{der}(s)$ *and all* $a \in \mathcal{A}ct_I$, *we have* $s' \stackrel{a}{\Longrightarrow}$. L *is input-enabled if* $\overline{s}$ *is input-enabled.*

Throughout this paper, we restrict to image finite, strongly converging LTSs. The *testing hypothesis* for **ioco** states that *implementations* can be modelled using *input-enabled* LTSs. Note that this does not imply that the theory requires that this LTS is known. The *conformance* relation **ioco** is defined as follows:

Definition 3. *Let* $L_i = \langle S_i, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow_i, \overline{s}_i \rangle$ *(for* $i = 1, 2$*) be two LTSs. Let* $s_1 \in S_1$ *and* $s_2 \in S_2$. *Then* s_1 *is* **ioco***-conforming to* s_2 *– notation* s_1 $\mathbf{ioco}$ s_2 *– when* s_1 *is input-enabled and*

$$\forall \sigma \in \textit{s-traces}(s_2) : \ \mathbf{out}([s_1]_\sigma) \subseteq \mathbf{out}([s_2]_\sigma)$$

We sometimes write L_1 $\mathbf{ioco}$ L_2 *instead of* $\overline{s}_1$ $\mathbf{ioco}$ $\overline{s}_2$.

Note that *proving* **ioco**-conformance is generally not feasible, as there is no guarantee that we have seen all the behaviours of an implementation (because of non-determinism). In practice, we settle for *confidence* in **ioco**-conformance, which is obtained by testing the implementation with a large set of successfully executed test-cases. A sound and complete algorithm for **ioco** for deriving test-cases from a specification is proved correct in [13]; it is implemented in e.g. TorX [1] and TGV [7].

Modal μ-calculus The modal μ-calculus is a powerful logic which can be used to express complex temporal properties over dynamic systems. Next to its modal operators $\langle a \rangle \phi$ and $[a]\phi$, it is equipped with least and greatest fixpoint operators. The grammar for the modal μ-calculus, given directly in positive form is as follows:

$$\phi ::= \mathsf{tt} \mid \mathsf{ff} \mid X \mid \phi \wedge \phi \mid [a]\phi \mid \langle a \rangle \phi \mid \phi \vee \phi \mid \mu X.\phi \mid \nu X.\phi$$

where $a \in \mathcal{A}ct_\tau$ is an action and X is a propositional variable from a set of propositional variables $\mathcal{X}$. A formula ϕ is said to be in *Positive Normal Form* (PNF) if all its propositional binding variables are distinct. We only consider formulae in PNF. A formula ϕ is interpreted relative to an LTS $L = \langle S, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow, \overline{s} \rangle$ and a *propositional environment* $\eta : \mathcal{X} \rightarrow 2^S$ that maps propositional variables to sets of states. The

semantics of ϕ is given by $[\![\phi]\!]^L_\eta$, which is defined as follows:

$$
\begin{array}{ll}
[\![\mathsf{tt}]\!]^L_\eta & = S \\
[\![\mathsf{ff}]\!]^L_\eta & = \emptyset \\
[\![\phi_1 \wedge \phi_2]\!]^L_\eta & = [\![\phi_1]\!]^L_\eta \cap [\![\phi_2]\!]^L_\eta \\
[\![\phi_1 \vee \phi_2]\!]^L_\eta & = [\![\phi_1]\!]^L_\eta \cup [\![\phi_2]\!]^L_\eta \\
[\![X]\!]^L_\eta & = \eta(X) \\
[\![[a]\phi]\!]^L_\eta & = \{s \in S \mid \forall s' \in S :\ s \xrightarrow{a} s' \Rightarrow s' \in [\![\phi]\!]^L_\eta\} \\
[\![\langle a\rangle\phi]\!]^L_\eta & = \{s \in S \mid \exists s' \in S :\ s \xrightarrow{a} s' \wedge s' \in [\![\phi]\!]^L_\eta\} \\
[\![\mu X.\phi]\!]^L_\eta & = \bigcap\{S' \subseteq S \mid [\![\phi]\!]^L_{\eta[S'/X]} \subseteq S'\} \\
[\![\nu X.\phi]\!]^L_\eta & = \bigcup\{S' \subseteq S \mid S' \subseteq [\![\phi]\!]^L_{\eta[S'/X]}\}
\end{array}
$$

where we write $\eta[S'/X]$ for the environment that coincides with η on all variables $Y \neq X$, and maps variable X to value S'. A state $s \in S$ satisfies a formula ϕ, written $s \models_L \phi$ when $s \in [\![\phi]\!]^L_\eta$. We write $L \models \phi$ when $\overline{s} \models_L \phi$.

The operator $\langle a\rangle\phi$ is used to express that there must exist an a transition from the current state to a state satisfying ϕ. Dually, the operator $[a]\phi$ is used to express that *all* states that can be reached by executing an a action satisfy property ϕ. Remark that when an a transition is impossible in a state s, the property $[a]\phi$ is trivially satisfied in state s. These operators are well-understood and can be found in early logics such as Hennessy-Milner Logic. In this paper, we use the following additional conventions: for sets of actions A we define:

$$[A]\phi \stackrel{\text{def}}{=} \bigwedge_{a\in A}[a]\,\phi \qquad\qquad \langle A\rangle\phi \stackrel{\text{def}}{=} \bigvee_{a\in A}\,\langle a\rangle\phi$$

Moreover, for a formula ϕ, we denote its dual by $\overline{\phi}$. Such a dual formula always exists and is readily obtained by simple transformations and renamings, see e.g. [2].

The major source for the expressive power of the modal μ-calculus is given by the fixpoint operators μ and its dual ν. Technically, a least fixpoint $\mu X.\phi$ is used to indicate the smallest solution of X in formula ϕ, whereas the greatest fixpoint $\nu X.\phi$ is used for the greatest solution of X in formula ϕ. These fixpoint expressions are generally understood as allowing one to express *finite looping* and *looping*, respectively.

Example 1. A system that can always perform at least one action is said to be *deadlock-free* (note that we do not require this to be a visible action). This can be expressed in the modal μ-calculus using a greatest fixpoint: $\nu X.\ [\mathcal{A}ct_\tau]X \wedge \langle \mathcal{A}ct_\tau\rangle\mathsf{tt}$. Informally, the formula expresses that we are interested in the largest set of states (say this would be $\overline{X}$) that satisfies the property that from each reachable state s ($s \in \overline{X}$), at least one action is enabled, and all enabled actions lead to states s' ($s' \in X$) that also have this property.

For a more detailed account we refer to [2], which provides an excellent treatment of the modal μ-calculus.

3 Techniques and Heuristics for Diagnostic Testing

Testing is a much used technique to validate whether an implementation conforms to its specification. Upon detection of a non-conformance, all that is available is a trace,

also known as a *symptom*, that led to this non-conformance. Such a symptom is often insufficient for locating the root-cause (or causes) of the non-conformance; for this, often additional tests are required. We refer to these additional tests as *diagnostic tests*.

In a Model-Based Testing setting, the basis for conducting diagnostic tests is given by a set of *fault models*. Each fault model provides a possible, formal explanation of the behaviour of the implementation; one may consider it a possible specification of the faulty implementation. Remark that we here appeal to the testing hypothesis of **ioco**, stating that there is an input enabled LTS model for every implementation. The different fault models describe different fault situations. The diagnostics problem thus consists of selecting one or more fault model(s) from the given set of fault models that best explain the behaviour of the implementation.

Formally, the diagnostics problem we are dealing with is the following: given a specification S, a non-conforming implementation I and a non-empty set of fault models $F = \{F_1, F_2, \ldots, F_n\}$. A *diagnosis* of I is given by the largest set $D \subseteq F$ satisfying I $\mathbf{ioco}$ F_i for all $F_i \in D$. The focus of this paper is on two techniques for obtaining D efficiently, viz. distinguishability and orthogonality. Note that given the partiality of the **ioco**-relation, the fault models in D are –generally– all unrelated.

In Sections 3.1 and 3.2, we introduce the notions of (strong and weak) *distinguishability* and (strong and weak) *orthogonality*, respectively. We provide alternative characterisations of all notions in terms of modal logic, which 1) provides a different perspective on the technique and, 2) enables the use of efficient commonplace tool support. The discussion on how exactly the theory and results described in this section can be utilised for diagnostic testing is deferred to Section 4.

3.1 Distinguishability

Given two fault models F_1 and F_2 and an implementation I. Chances are that during naive testing for I $\mathbf{ioco}$ F_1 and I $\mathbf{ioco}$ F_2, there is a large overlap between the test-cases for F_1 and F_2, as both try to model to a large extent the same implementation. This means that F_1 and F_2 often agree on the outcome of most test-cases. An effective technique for avoiding this redundancy is to exploit the *differences* between F_1 and F_2. In particular, when, after conducting an experiment σ on I, F_1 and F_2 predict different outputs, this provides the opportunity to remove at least one of the two fault models from further consideration. When one or more such experiments exist, we say that the fault models are *distinguishable*. Two types of distinguishability are studied: weakly and strongly distinguishable fault models.

We next formalise the above concepts. At the root of the distinguishability property is the notion of an *intersection* of fault models. Intuitively, the intersection of two fault models contains exactly those behaviours that are shared among the two fault models.

Definition 4. *Let $F_i = \langle S_i, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow_i, \overline{s}_i \rangle$, for $i = 1, 2$ be two LTSs. Assume $\Delta \notin \mathcal{A}ct$ is a fresh constant, and denote $\mathcal{A}ct_U \cup \{\Delta\}$ by $\mathcal{A}ct_U^{\Delta}$. Likewise, $\mathcal{A}ct^{\Delta}$. The* intersection *of F_1 and F_2, denoted $F_1 || F_2$, is again an LTS defined by $\langle (2^{S_1} \setminus \emptyset) \times (2^{S_2} \setminus \emptyset),\ \mathcal{A}ct_I, \mathcal{A}ct_U^{\Delta},\ \rightarrow,\ ([\overline{s}_1]_\epsilon, [\overline{s}_2]_\epsilon)\ \rangle$, where $\rightarrow$ is defined by the following rules:*

$$\frac{\emptyset \neq q_1 \subseteq S_1 \quad \emptyset \neq q_2 \subseteq S_2 \quad a \in \mathcal{A}ct}{(q_1, q_2) \xrightarrow{a} ([q_1]_a, [q_2]_a)} \qquad \frac{\emptyset \neq q_1 \subseteq S_1 \quad \emptyset \neq q_2 \subseteq S_2}{(q_1, q_2) \xrightarrow{\Delta} ([q_1]_\delta, [q_2]_\delta)}$$

Remark that no transitions lead to, or start in an element $(q, \emptyset)$ or $(\emptyset, q)$ since these are no elements of the state-space of the intersection of two LTSs.

The intersection of two LTSs extends the alphabet of output actions of both LTSs with the symbol Δ. This action captures the synchronisation of both LTSs over the observations of quiescence, which in the **ioco**-setting is treated as an output of the system. A "true" quiescent state in the intersection of two LTSs indicates that the output actions offered by both LTSs are strictly disjoint. In order to facilitate the mapping between the sets $\mathcal{A}ct_\delta$ and $\mathcal{A}ct^\Delta$, we use a *relabelling* function. Let $\mathfrak{R} : \mathcal{A}ct^\Delta \rightarrow \mathcal{A}ct_\delta$ be the following bijective function:

$$\mathfrak{R}(a) \stackrel{\text{def}}{=} a \text{ if } a \neq \Delta \text{ and } \delta \text{ otherwise}$$

We write $\mathfrak{R}^{-1}$ to denote the inverse of $\mathfrak{R}$. The mapping $\mathfrak{R}$ and its inverse extend readily over sets of actions. The extension of the mapping $\mathfrak{R}$ (and its inverse) over (sets of) traces, denoted by the mapping $\mathfrak{R}^*$ (resp. $\mathfrak{R}^{-1*}$), is defined in the obvious way.

Property 1. Let $F_1||F_2$ be the intersection of F_1 and F_2, and let s_1 be a state of F_1, s_2 be a state of F_2, (q_1, q_2) be a state of $F_1||F_2$ and $\sigma \in \mathcal{A}ct^*_\delta$. Then:

1. $F_1||F_2$ is deterministic,
2. $[([s_1]_\sigma, [s_2]_\sigma)]_a \neq \emptyset$ implies $([s_1]_{\sigma\mathfrak{R}(a)}, [s_2]_{\sigma\mathfrak{R}(a)}) \in [([s_1]_\sigma, [s_2]_\sigma)]_a$,
3. $\mathbf{out}(([q_1]_\epsilon, [q_2]_\epsilon)) \setminus \{\delta\} = \mathfrak{R}^{-1}(\mathbf{out}([q_1]_\epsilon) \cap \mathbf{out}([q_2]_\epsilon))$.

Some of the above properties should not come as a surprise: at the basis of the intersection operator is the *Suspension Automata* transformation of [13], which codes a non-deterministic specification into a deterministic LTS with explicit suspension transitions. That transformation is known to retain the exact same **ioco** testing power as the original specification, albeit on different domains of specification models.

Strong Distinguishability Recall that the intersection $F_1||F_2$ codes the behaviours that are shared among the LTSs F_1 and F_2. This means that in states of $F_1||F_2$ that have no output transitions, both LTSs disagree on the outputs that should occur, providing the opportunity to eliminate at least one of the two fault models. We say that such a state is *discriminating*. If a tester always has a finite "winning strategy" for steering an implementation to such a discriminating state, the fault models are *strongly distinguishable*. Recall that testing is sometimes portrayed as a (mathematical) game in which the tester is in control of the inputs and the system is in control of the outputs. We next formalise the notion of strong distinguishability.

Definition 5. *The intersection $F_1||F_2 = \langle S, \mathcal{A}ct_I, \mathcal{A}ct^\Delta_U, \rightarrow, \overline{s}\rangle$ is said to be* root-discriminating *if there exists a natural number k, such that $\overline{s} \in \mathcal{D}_{F_1||F_2}(k)$, where $\mathcal{D}_{F_1||F_2} : \mathbb{N} \rightarrow 2^S$ is inductively defined by:*

$$\begin{cases} \mathcal{D}_{F_1||F_2}(0) & = \{t \in S \mid \mathbf{out}([t]_\epsilon) = \{\delta\}\} \\ \mathcal{D}_{F_1||F_2}(n+1) & = \bigcap_{a \in \mathcal{A}ct^\Delta_U} \{t \in S \mid [t]_a \subseteq \mathcal{D}_{F_1||F_2}(n)\} \\ & \quad \cup \bigcup_{a \in \mathcal{A}ct_I} \{t \in S \mid \emptyset \neq [t]_a \subseteq \mathcal{D}_{F_1||F_2}(n)\} \end{cases}$$

A state $s \in \mathcal{D}_{F_1||F_2}(k)$ is referred to as a k-discriminating state. If it is clear from the context, we drop the subscript $F_1||F_2$ from the mapping $\mathcal{D}_{F_1||F_2}$. We say that fault models F_1 and F_2 are strongly distinguishable *iff $F_1||F_2$ is root-discriminating.*

Property 2. For all intersections $F_1||F_2$ and all $k \geq 0$, we have $\mathcal{D}(k+1) \supseteq \mathcal{D}(k)$.

Note that a state s is allowed to be $(k+1)$-discriminating if there is a strategy to move from state s to a state which is k-discriminating via some input, even though there are some outputs that would not lead to a k-discriminating state. This is justified by the fact that the implementations that we consider are input enabled. This means that they have to be able to accept inputs at all times, and input may therefore pre-empt possible output of a system. Strong distinguishability is preserved under **ioco**-conformance which means that if two fault models are strongly distinguishable, then also the implementations/refinements they model behave observably differently.

Property 3. Let F_1, F_2 be fault models, and let I_1, I_2 be implementations. If $I_1 \ \mathbf{ioco}\ F_1$ and $I_2 \ \mathbf{ioco}\ F_2$ and F_1 and F_2 are strongly distinguishable, then so are I_1 and I_2.

Strong distinguishability can be characterised by means of a modal μ-calculus formula. The formal connection is established by the following theorem.

Theorem 1. *Let F_1, F_2 be two fault models. Then F_1 and F_2 are strongly distinguishable iff $F_1||F_2 \models \phi_{sd}$, where*

$$\phi_{sd} \stackrel{\text{def}}{=} \mu X.\ [\mathcal{A}ct_U^{\Delta}]X \vee \langle \mathcal{A}ct_I \rangle X$$

Weak Distinguishability Strong distinguishability as a property is quite powerful, as it ensures that there is a testing strategy that inevitably leads to a verdict about one of the two fault models. However, it is often the case that there is no such fail-safe strategy, even though reachable discriminating states are present in the intersection. We therefore introduce the notion of *weak distinguishability*.

Definition 6. *Two fault models F_1, F_2 are said to be* weakly distinguishable *if and only if $\mathbf{der}(F_1||F_2) \cap \mathcal{D}(0) \neq \emptyset$.*

The problem of deciding whether two fault models are weakly distinguishable is a standard reachability property as testified by the following correspondence.

Theorem 2. *Let F_1, F_2 be two fault models. Then F_1 and F_2 are weakly distinguishable iff $F_1||F_2 \models \phi_{wd}$, where*

$$\phi_{wd} \stackrel{\text{def}}{=} \mu X.\ \langle \mathcal{A}ct^{\Delta} \rangle X \vee [\mathcal{A}ct_U^{\Delta}]\mathsf{ff}$$

Unlike strong distinguishability, weak distinguishability is not preserved under **ioco**. This is illustrated by the following example:

Example 2. Let F_1 and F_2 be two fault models and let I be an implementation (see Fig. 1). Clearly, $I \ \mathbf{ioco}\ F_1$ and $I \ \mathbf{ioco}\ F_2$. Moreover, F_1 and F_2 are weakly distinguishable, as illustrated by the trace $?b.!e$. However, I is clearly not weakly distinguishable from itself, as distinguishability is irreflexive.

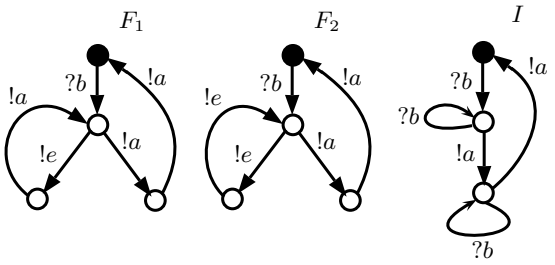

Fig. 1. Fault models F_1 and F_2 and implementation I.

3.2 Orthogonality

Whereas distinguishability focuses on the differences in output for two given fault models, it is equally well possible that there is a difference in the specified inputs. Note that this is allowed in **ioco**-testing theory: a specification does not have to be input complete; this partiality with respect to inputs supports a useful form of underspecification. In practice, a fault hypothesis can often be tested by focusing testing effort on particular aspects. Exploiting the differences in underspecifications of the fault models gives rise to a second heuristic, called *orthogonality*, which we describe in this section. We start by extending the intersection operator of Def. 4.

Definition 7. *Let $F_i = \langle S_i, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow_i, \overline{s}_i \rangle$, for $i = 1, 2$ be two fault models. Assume $\Theta = \{\Theta_a \mid a \in \mathcal{A}ct_I\}$ is a set of fresh constants disjoint from $\mathcal{A}ct^{\Delta}$. We denote $\mathcal{A}ct \cup \Theta$ by $\mathcal{A}ct^{\Delta}_{\Theta}$. The orthogonality-aware intersection of F_1 and F_2, denoted $F_1||_{\Theta}F_2$, is an LTS defined by $\langle (2^{S_1} \setminus \emptyset) \times (2^{S_2} \setminus \emptyset), \mathcal{A}ct^{\Theta}_I, \mathcal{A}ct^{\Delta}_U, \rightarrow, ([\overline{s}_1]_{\epsilon}, [\overline{s}_2]_{\epsilon}) \rangle$, where $\rightarrow$ is defined by the two rules of Def. 4 in addition to the following two rules:*

$$\frac{\emptyset \neq q_1 \subseteq S_1 \quad \emptyset \neq q_2 \subseteq S_2 \quad [q_1]_a \neq \emptyset \quad [q_2]_a = \emptyset \quad a \in \mathcal{A}ct_I}{(q_1, q_2) \xrightarrow{\Theta_a} (q_1, q_2)}$$

$$\frac{\emptyset \neq q_1 \subseteq S_1 \quad \emptyset \neq q_2 \subseteq S_2 \quad [q_2]_a \neq \emptyset \quad [q_1]_a = \emptyset \quad a \in \mathcal{A}ct_I}{(q_1, q_2) \xrightarrow{\Theta_a} (q_1, q_2)}$$

Property 4. Let $F_1||_{\Theta}F_2$ be the orthogonality-aware intersection of F_1 and F_2, and let (q_1, q_2) be a state of $F_1||_{\Theta}F_2$. Then:

1. $F_1||_{\Theta}F_2$ is deterministic,
2. For all inputs $a \in \mathcal{A}ct_I$, $(q_1, q_2) \xrightarrow{a}$ implies $(q_1, q_2) \not\xrightarrow{\Theta_a}$.

Note that the reverse of Property 4, item 2 does not hold exactly because of the input incompleteness of fault models in general. Intuitively, the occurrence of a label Θ_a in the orthogonality-aware intersection models the fact that input a is specified by only one of the two LTSs and is left unspecified by the other LTS. The presence of such labels in the orthogonality-aware intersection are therefore pointers to the orthogonality of two systems. Once an experiment arrives in a state with an orthogonality label Θ_a, testing can focus on one of the two fault models exclusively. Any test failure that is subsequently found is due to the incorrectness of the selected fault model. We next formalise the notions of strong and weak orthogonality, analogously to distinguishability.

Definition 8. *Let $F_1||_\Theta F_2 = \langle S, \mathcal{A}ct_I^\Theta, \mathcal{A}ct_U^\Delta, \rightarrow, \overline{s}\rangle$. F_1 and F_2 are said to be* strongly orthogonal *if there is a natural number k such that $\overline{s} \in \mathcal{O}_{F_1||_\Theta F_2}(k)$, where $\mathcal{O}_{F_1||_\Theta F_2} : \mathbb{N} \rightarrow 2^S$ is inductively defined by:*

$$\begin{cases} \mathcal{O}_{F_1||_\Theta F_2}(0) & = \{t \in S \mid \exists a \in \mathcal{A}ct_I : t \xrightarrow{\Theta_a}\} \\ \mathcal{O}_{F_1||_\Theta F_2}(n+1) & = \bigcap_{a \in \mathcal{A}ct_U^\Delta} \{t \mid [t]_a \subseteq \mathcal{O}_{F_1||_\Theta F_2}(n) \wedge \exists a' \in \mathcal{A}ct_U : [t]_{a'} \neq \emptyset\} \\ & \quad \cup \bigcup_{a \in \mathcal{A}ct_I} \{t \mid \emptyset \neq [t]_a \subseteq \mathcal{O}_{F_1||_\Theta F_2}(n) \vee t \xrightarrow{\Theta_a}\} \end{cases}$$

The following theorem recasts strong orthogonality as a modal property.

Theorem 3. *Fault models F_1 and F_2 are strongly orthogonal iff $F_1||_\Theta F_2 \models \phi_{so}$, where* $\phi_{so} \stackrel{\text{def}}{=} \mu X.\ (\langle \mathcal{A}ct_U^\Delta \rangle \mathtt{tt} \wedge [\mathcal{A}ct_U^\Delta] X) \vee \langle \mathcal{A}ct_I \rangle X \vee \langle \Theta \rangle \mathtt{tt}$

Analogously to distinguishability, we define a weak variation of strong orthogonality, which states that it is possible to reach a state in which an orthogonal label Θ_a for some a is enabled.

Definition 9. *Given $F_1||_\Theta F_2 = \langle S, \mathcal{A}ct_I^\Theta, \mathcal{A}ct_U^\Delta, \rightarrow, \overline{s}\rangle$. F_1 and F_2 are said to be* weakly orthogonal *iff $\mathbf{der}(F_1||_\Theta F_2) \cap \mathcal{O}(0) \neq \emptyset$.*

A recast of weak orthogonality into the μ-calculus is as follows.

Theorem 4. *Fault models F_1 and F_2 are weakly orthogonal iff $F_1||_\Theta F_2 \models \phi_{wo}$, where* $\phi_{wo} \stackrel{\text{def}}{=} \mu X.\ \langle \mathcal{A}ct^\Delta \rangle X \vee \langle \Theta \rangle \mathtt{tt}$

Orthogonality is not preserved under **ioco** conformance, which is illustrated by the following example.

Example 3. Let F_1 and F_2 be two fault models and let I_1 and I_2 be two implementations, depicted in Fig. 2. Clearly, I_1 **ioco** F_1 and I_2 **ioco** F_2. Moreover, F_1 and F_2 are (strongly and weakly) orthogonal, as illustrated by the trace $?b.?b$ which is applicable for F_1, but not applicable for F_2. However, I_1 and I_2 are not orthogonal. Note that by repeatedly executing experiment $?b.?b$ and subsequently observing output confidence in the correctness of (aspects of) F_1 can increase.

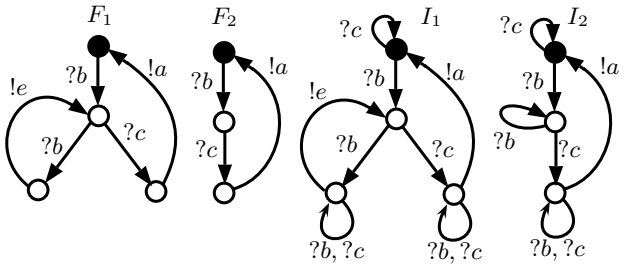

Fig. 2. Fault models F_1 and F_2 and implementations I_1 and I_2.

4 Automating Diagnostic Testing

In the previous section we formalised the notions of *distinguishability* and *orthogonality*, both in terms of set-theory and modal logic. In this section, we rely on the latter results for defining provably correct algorithms for eliminating fault models and for isolating behaviours of fault models for further scrutiny.

First, we introduce the basic tools that we rely on for defining our on-the-fly diagnostic testing algorithms and semi-decision procedures. Then, in Section 4.2 we define the algorithms for strong distinguishability and orthogonality, and in Section 4.3, the semi-decision procedures for weak distinguishability and orthogonality are given.

4.1 Preliminaries

For the remainder of these sections, we assume that I is an implementation that we wish to subject to diagnostic testing, and $F_i = \langle S_i, \mathcal{A}ct_I, \mathcal{A}ct_U, \rightarrow_i, \overline{s}_i \rangle$, for $i = 1, 2$ are two given fault models. $F_1 ||_{(\Theta)} F_2 = \langle S, \mathcal{A}ct_I^{(\Theta)}, \mathcal{A}ct_U^{\Delta}, \rightarrow, \overline{s} \rangle$ is the (orthogonality-aware) intersection of F_1 and F_2. From this time forth, we assume to have the following four methods at our disposal:

1. `Apply`(a): send input action $a \in \mathcal{A}ct_I$ to an implementation,
2. `Observe`(): observe some output $a \in \mathcal{A}ct_U \cup \{\delta\}$ from an implementation,
3. `Counterexample`(L, ϕ): returns an arbitrary counterexample for $L \models \phi$ if one exists, and returns $\perp$ otherwise.
4. `Counterexample`$_s(L, \phi)$: returns one among possibly many shortest counterexamples for $L \models \phi$ if a counterexample exists, and returns $\perp$ otherwise.

We refer to [9] for an explanation of the computation of counterexamples for the modal μ-calculus. In our ordeals we assume that $\perp$ is a special character that we can concatenate to sequences of actions.

4.2 Strong Distinguishability and Strong Orthogonality

Suppose F_1 and F_2 are strongly distinguishable or orthogonal. Algorithm 1 derives and executes (on-the-fly) an experiment that (see also Theorem 5), depending on the input:

- allows to eliminate at least one fault model from a set of fault models, or
- isolates a fault model for further testing.

Recall that $\overline{\phi}$ denotes the dual of ϕ (see Section 2). Informally, the algorithm works as follows for strongly distinguishable fault models F_1 and F_2 (likewise for strongly orthogonal fault models): η is the shortest counterexample for F_1 and F_2 *not* being strongly distinguishable. The algorithm tries to replay η on the implementation, and recomputes a new counterexample when an output produced by the system-under-test does not agree with the output specified in the counterexample. When the counterexample has length 0, we can be sure to have reached a discriminating state, and observing output in this state eliminates at least one of the two considered fault models.

Algorithm 1 Algorithm for exploiting strong distinguishability/orthogonality

Require: $P \subseteq S$, $|P| = 1$, η is a shortest counterexample for $P \models \overline{\phi}_x$, $\phi_x \in \{\phi_{sd}, \phi_{so}\}$
Ensure: Returns a sequence executed on I.

function $\mathcal{A}_1(P, \eta, \phi_x)$
  if $\eta = \epsilon$ **then**
    if $\phi_x = \phi_{sd}$ **then return** `Observe`();
    else choose a **from** $\{y \in \mathcal{A}ct_I \mid [P]_{\Theta y} \neq \emptyset\}$; **return** a;
    end if
  else ▷ Assume $\eta \equiv e\ \eta'$ for some action e and sequence η'
    if $e \in \mathcal{A}ct_I$ **then** `Apply`(e); **return** $e\ \mathcal{A}_1([P]_e, \eta', \phi_x)$;
    else $a :=$ `Observe`();
      if $a = e$ **then return** $e\ \mathcal{A}_1([P]_e, \eta', \phi_x)$;
      else if $\mathfrak{R}^{-1}(a) \in \mathbf{out}(P)$ **then**
        return $a\ \mathcal{A}_1([P]_a, \mathfrak{R}^*(\texttt{Counterexample}_s(\texttt{(}[P]_a, \overline{\phi}_x\texttt{)})), \phi_x)$;
      else return a;
      end if
    end if
  end if
end function

Theorem 5. *Let F_1 and F_2 be strongly orthogonal or strongly distinguishable fault models. Let $\phi = \phi_{sd}$ when F_1 and F_2 are distinguishable and let $\phi = \phi_{so}$ when F_1 and F_2 are orthogonal. Then algorithm $\mathcal{A}_1(\{\overline{s}\}, \texttt{Counterexample}_s\texttt{(}F_1||_\Theta F_2, \overline{\phi}\texttt{)}, \phi)$ terminates and the sequence $\sigma \equiv \sigma'\ a$ it returns satisfies:*

1. *$a \in \mathcal{A}ct_\delta \backslash \mathcal{A}ct_I$ implies $\mathbf{out}([I]_{\sigma'}) \not\subseteq \mathbf{out}([F_1]_{\sigma'})$ or $\mathbf{out}([I]_{\sigma'}) \not\subseteq \mathbf{out}([F_2]_{\sigma'})$,*
2. *$a \in \mathcal{A}ct_I$ implies $\phi = \phi_{so}$ and $[F_1]_\sigma = \emptyset$ or $[F_2]_\sigma = \emptyset$.*

The sequence that is returned by the algorithm can be used straightforwardly for checking which fault model(s) can be eliminated, or which fault model is selected for further scrutiny (see also Section 4.5). Such "verdicts" are easily added to our algorithms, but are left out for readability.

4.3 Weak distinguishability and Weak Orthogonality

In case F_1 and F_2 are not strongly but weakly distinguishable (resp. weakly orthogonal), there is no guarantee that a discriminating (resp. orthogonal) state is reached. By conducting sufficiently many tests, however, chances are that one of such states is eventually reached, unless the experiment has run off to a part of the state space in which no discriminating (resp. orthogonal) states are reachable. Semi-decision procedure 2 conducts experiments on implementation I, and terminates in the following three cases:

1. if a sequence has been executed that led to a discriminating/orthogonal state,
2. if an output was observed that conflicts at least one of the fault models,
3. if discriminating/orthogonal states are no longer reachable.

So long as neither of these cases are met, the procedure does not terminate. The semi-decision procedure works in roughly the same manner as the algorithm of the previous

section. The main differences are in the termination conditions (and the result it returns), and, secondly the use of arbitrary counterexamples, as shorter counterexamples are not necessarily more promising for reaching a discriminating/orthogonal state.

Algorithm 2 Procedure for exploiting weak distinguishability/orthogonality

Require: $P \subseteq S$, $|P| = 1$, η is any counterexample for $P \models \overline{\phi}_x$, $\phi_x \in \{\phi_{wo}, \phi_{wd}\}$
Ensure: Returns a sequence executed on I upon termination

function $\mathcal{A}_2(P, \eta, \phi_x)$
  if $\eta = \epsilon$ **then**
    if $\phi_x = \phi_{wd}$ **then return** `Observe()`;
    else choose a **from** $\{y \in \mathcal{A}ct_I \mid [P]_{\Theta_y} \neq \emptyset\}$; **return** a;
    end if
  else ▷ Assume $\eta \equiv e\,\eta'$ for some action e and sequence η'
    if $e \in \mathcal{A}ct_I$ **then** `Apply(e)`; **return** $e\,\mathcal{A}_2([P]_e, \eta', \phi_x)$;
    else $a :=$ `Observe()`;
      if $a = e$ **then return** $e\,\mathcal{A}_2([P]_e, \eta', \phi_x)$;
      else if $\mathfrak{R}^{-1}(a) \in \mathbf{out}(P) \wedge$ `Counterexample(`$[P]_a, \overline{\phi}_x$`)` $\neq \bot$ **then**
        return $a\,\mathcal{A}_2([P]_a, \mathfrak{R}^*($`Counterexample(`$[P]_a, \overline{\phi}_x$`)`$), \phi_x)$;
      else if $\mathfrak{R}^{-1}(a) \in \mathbf{out}(P) \wedge$ `Counterexample(`$[P]_a, \overline{\phi}_x$`)` $= \bot$ **then**
        return $\bot$;
      else return a;
      end if
    end if
  end if
end function

Theorem 6. *Let F_1 and F_2 be weakly orthogonal or weakly distinguishable fault models. Let $\phi = \phi_{wd}$ when F_1 and F_2 are distinguishable and let $\phi = \phi_{wo}$ when F_1 and F_2 are orthogonal. If algorithm $\mathcal{A}_2(\{\overline{s}\},$ `Counterexample(`$F_1||_\Theta F_2, \overline{\phi}$`)`$, \phi)$ terminates it returns a sequence $\sigma \equiv \sigma' a$ satisfying:*

1. *$a \in \mathcal{A}ct_\delta \backslash \mathcal{A}ct_I$ implies $\mathbf{out}([I]_{\sigma'}) \not\subseteq \mathbf{out}([F_1]_{\sigma'})$, or $\mathbf{out}([I]_{\sigma'}) \not\subseteq \mathbf{out}([F_2]_{\sigma'})$,*
2. *$a \in \mathcal{A}ct_I$ implies $\phi = \phi_{wo}$ and $[F_1]_\sigma = \emptyset$ or $[F_2]_\sigma = \emptyset$.*
3. *$a = \bot$ implies either $\phi = \phi_{wo}$ and $der([\overline{s}]_{\sigma'}) \cap \mathcal{O}(0) = \emptyset$, or $\phi = \phi_{so}$ and $\mathbf{der}([\overline{s}]_{\sigma'}) \cap \mathcal{D}(0) = \emptyset$.*

The following example illustrates that the semi-decision procedure does not necessarily terminate.

Example 4. Suppose the intersection of two fault models is given by $F_1||F_2$ and the malfunctioning implementation is given by I (see Fig. 3). Clearly, F_1 and F_2 are weakly distinguishable, which means semi-decision procedure 2 is applicable. A counterexample to non-weak distinguishability is e.g. $?b!e?b?b!a$, so the procedure might try to execute this sequence. However, termination is not guaranteed, as the implementation may never execute action $!a$, but output $!e$ instead, making the semi-decision procedure recompute new counterexamples.

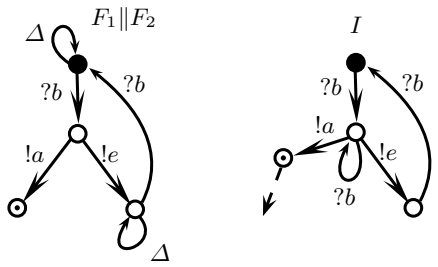

Fig. 3. No termination guaranteed for semi-decision procedure 2.

4.4 Optimisations

The algorithms for strong distinguishability (resp. strong orthogonality) in the previous section can be further optimised in a number of ways. First, one can include a minor addition to the standard model-checking algorithm, marking each k-discriminating (resp. k-orthogonal) state in the LTS that is checked with its depth k. While this has a negligible negative impact on the time complexity of the model checking algorithm, the state markings allow for replacing the method `Counterexample`$_s$`()` with a constant-time operation. Secondly, upon reaching a node in $\mathcal{D}(k)$ ($\mathcal{O}(k)$, respectively), the semi-decision procedure for weak distinguishability/orthogonality could continue to behave as algorithm 1. Furthermore, the orthogonality aware intersection is an extension of the plain intersection. Computing both is therefore unnecessary: only the former is needed; in that case, the formulae for strong and weak distinguishability need to be altered to take the extended set of input actions into account.

4.5 Diagnostic Testing Methodology

Distinguishability and orthogonality, and their associated algorithms, help in reducing the effort that is required for diagnostic testing. Thus far, we presented these techniques without addressing the issue of when a particular technique is worth investigating. In this section, we discuss a methodology for employing these techniques in diagnostic testing. For the remainder of this section, we assume a faulty implementation I and a given set of fault models $F = \{F_1, \ldots, F_n\}$.

We propose a stepwise refinement of the diagnostic testing problem using distinguishability and orthogonality. The first step in our methodology is to identify the largest non-symmetric set of pairs of strongly distinguishable fault models G. We next employ the following strategy: so long as $G \neq \emptyset$, select a pair $(F_i, F_j) \in G$ and provide this pair as input to algorithm 1. Upon termination of the algorithm, an experiment $\sigma \equiv \sigma' a$ is returned, eliminating F_k from F iff $a \notin \mathbf{out}([F_k]_{\sigma'})$ ($k = i, j$). Moreover, remove all fault models F_l for which $[F_l]_{\sigma'} \neq \emptyset$ and $a \notin \mathbf{out}([F_l]_{\sigma'})$ and recompute G. A worst case scenario requires at most $|G|$ iterations to reach $G = \emptyset$. The process can be further optimised by ordering fault models w.r.t. **ioco**-testing power, but it is beyond the scope of this paper to elaborate on this.

When G is empty, no strongly distinguishable pair can be found in F. The set of fault models can be further reduced using the weak distinguishability and strong orthogonality heuristics, in no particular order, as neither allows for a fail-safe strategy

to a conclusive verdict. As a last resort, weak orthogonality is used before conducting naive testing using the remaining fault models.

5 Example

As an illustration of some of the techniques that we presented in this paper, we consider a toy example concerning the prototypical coffee machine. The black-box behaviour of the coffee-machine is defined by specification S in Fig. 4, where action $?c$ and $!c$ represent a coffee request and production, $?t$ and $!t$ represent a tea request and production, and $?m$ and $!m$ represent a coffee-cream request and production. Among the set of fault

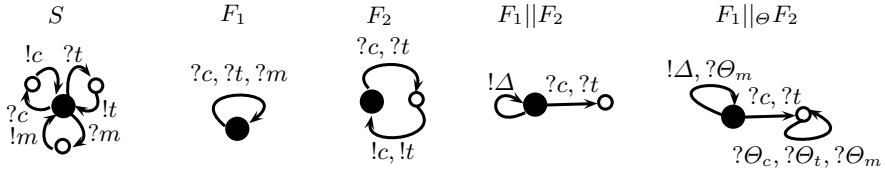

Fig. 4. Specification S and fault models F_1, F_2 and F_3 of a coffee machine

models for a misbehaving implementation of S are fault models F_1 (modelling e.g. a broken keypad in the machine) and F_2 (modelling e.g. a broken recipe book). Computing their intersection and their orthogonal-aware intersection, we find that F_1 and F_2 are strongly distinguishing and strongly orthogonal. The preferred choice here would be to run algorithm 1 with arguments setting it to check for strong distinguishability using e.g. $?t$ as input for the shortest counterexample. Algorithm 1 would first offer $?t$ to the implementation (which is accepted by assumption that implementations are *input-enabled*). Since then the shortest counterexample to non-strong distinguishability would be the empty string ϵ, the algorithm queries the output of the implementation and terminates. Any output the implementation produces either violates F_1 or F_2, or both. In case one would insist on using strong orthogonality, algorithm 1 would be used with the emtpy string ϵ as the shortest counterexample to non-strong orthogonality. The algorithm would return the sequence $?m$, indicating that isolated aspects of F_1 can be tested by experiments starting with input $?m$.

6 Concluding Remarks

We considered the problem of diagnosis for reactive systems, the problem of finding an explanation for a detected malfunction of a system. As an input to the diagnosis problem, we assumed a set of fault models. Each fault model provides a formal explanation of the behaviour of an implementation in terms of an LTS model. From this set of fault models, those models that do not correctly describe (aspects of) the implementation must be eliminated. As may be clear, this can be done naively by testing the implementation against each fault model separately, but this is quite costly. We have introduced several methods, based on model-based testing and model checking techniques, to make this selection process more effective.

Concerning issues for future research, we feel that the techniques that we have described in this paper can be further improved upon by casting our techniques in a

quantitative framework. By quantifying the differences and overlap between the outputs described by two fault models, a more effective strategy may be found. The resulting quantitative approach can be seen as a generalisation of our notion of weak distinguishability. Such a quantitative approach may very likely employ techniques developed in model checking with costs (or rewards). A second issue that we intend to investigate is the transfer of our results to the setting of real-time, in particular for fault models given by Timed Automata. In our discussions, we restricted our attention to the problem of selecting the right fault models from a set of explicit fault models by assuming this set was obtained manually, thereby side-stepping the problem of obtaining such fault models in the first place. Clearly, identifying techniques for automating this process is required for a full treatment of diagnosis for LTSs. Lastly, and most importantly, the efficacy of the techniques that we have developed in this paper must be assessed on real-life case-studies. There is already some compelling evidence of their effectiveness in [5] where a notion of distinguishability is successfully exploited in the setting of communicating FSM nets.

Acknowledgement The authors would like to thank Vlad Rusu, Jan Tretmans and René de Vries for stimulating discussions and advice on the subjects of diagnosis and testing.

References

1. A. Belinfante, J. Feenstra, R.G. de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *Testcom '99*, pages 179–196. Kluwer, 1999.
2. J.C. Bradfield and C.P. Stirling. Modal logics and mu-calculi: an introduction. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, chapter 4, pages 293–330. Elsevier, 2001.
3. K. El-Fakih, S. Prokopenko, N. Yevtushenko, and G. von Bochmann. Fault diagnosis in extended finite state machines. In D. Hogrefe and A. Wiles, editors, *Proc. TestCom 2003*, volume 2644 of *LNCS*, pages 197–210. Springer-Verlag, 2003.
4. K. El-Fakih, N. Yevtushenko, and G. von Bochmann. Diagnosing multiple faults in communicating finite state machines. In *Proc. FORTE'01*, pages 85–100. Kluwer, B.V., 2001.
5. M. Gromov, A. Kolomeetz, and N. Yevtushenko. Synthesis of diagnostic tests for fsm nets. *Vestnik of TSU*, 9(1):204–209, 2004.
6. Q. Guo, R.M. Hierons, M. Harman, and K. Derderian. Heuristics for fault diagnosis when testing from finite state machines. *Softw. Test. Verif. Reliab.*, 17:41–57, 2007.
7. C. Jard and T. Jéron. Tgv: theory, principles and algorithms. *STTT*, 7(4):297–315, 2005.
8. T. Jéron, H. Marchhand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Proc. WODES 2006*. IEEE, 2006.
9. A. Kick. *Generation of Counterexamples and Witnesses for Model Checking*. PhD thesis, Fakultät für Informatik, Universität Karlsruhe, Germany, July 1996.
10. G. Lamperti, M. Zanella, and P. Pogliano. Diagnosis of active systems by automata-based reasoning techniques. *Applied Intelligence*, 12(3):217–237, 2000.
11. A. Petrenko and N. Yevtushenko. Testing from partial deterministic fsm specifications. *IEEE Trans. Comput.*, 54(9):1154–1165, 2005.
12. J. Pietersma, A.J.C. van Gemund, and A. Bos. A model-based approach to sequential fault diagnosis. In *Proceedings IEEE AUTOTESTCON 2005*, 2005.
13. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 17(3):103–120, 1996.