RWTH Aachen

Department of Computer Science
Technical Report

Quantitative Testing

Henrik Bohnenkamp and Mariëlle Stoelinga

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2008-01

RWTH Aachen · Department of Computer Science · January 2008

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Quantitative Testing

Henrik Bohnenkamp‡ Mariëlle Stoelinga†

‡Software Modeling & Verification
Department of Computer Science
RWTH Aachen University
D-52056 Aachen, Germany
henrik@cs.rwth-aachen.de

†Formal Methods & Tools
Faculty EEMCS
University of Twente
NL-7500 AE Enschede
The Netherlands
marielle@cs.utwente.nl

Abstract. We investigate the problem of specification based testing with dense sets of inputs and outputs, in particular with imprecision as they might occur due to imprecise measurements, numerical instability or noisy channels. Using quantitative transition systems to describe implementations and specifications, we introduce implementation relations that capture a notion of correctness "up to ε", i.e. that allow deviation of the implementations behavior from that of the specification as long as it does not deviate more than ε. The deviations are described as Hausdorff distances between certain sets of traces. The implementation relations are conservative extensions of the well-known ioco relation. We develop a testing algorithm that we show to be sound and exhaustive with respect to the implementation relations introduced.

1 Introduction

Model-driven test theories have recently been developed, which surpass plain functional testing by taking also quantitative information of the system under test into account: $[2, 1, 6, 7]$ extend the classical model-driven test theories $[11, 5]$ with real-time and data respectively, and [13] to testing of hybrid systems. These works provide a solid formal underpinning of real-time, hybrid and data testing, together with methods for automatic test case generation, execution and evaluation for system with real-time and data. These theories, however, handle the values contained within the requirement specification and the implementation-under-test (IUT) with an infinite precision, not taking into account deviations due to measurement errors, numerical instability or noisy channels: if the specification requires a response within 1 second, but the IUT responds within 1.01 second, a fail verdict is generated, not allowing small deviations from the requirements for measurement errors etc.

For real-time testing, in [9] this problem is overcome by explicitly modeling the tester's time observation capabilities through a digital clock. In the area of verification, the related problem that real-time models are idealized mathematical abstractions from reality that may not be implementable or robust have gained attention in $[8, 10, 3]$.

This paper presents a model-driven test theory in the presence of imprecisions: the theory comprises definitions of implementation relations which take imprecision into account, and an on-the-fly testing algorithm that is shown to be sound and exhaustive w.r.t. the implementation relations. Rather than concentrating on one particular area like timed or hybrid testing, we take a more abstract view to testing when imprecisions are present; our results can can later be specialized to deal with the particularties of a concrete testing theory. We

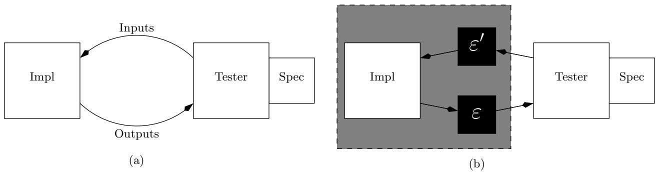

Fig. 1. Testing Scenarios

set our theory in the context of *quantitative transition systems.* These are an extension of input/output transition systems with continuous information: Each action in a QTS carries also a value $x \in [0, 1]$. Based on this model class, we define conformance relations $\mathbf{qioco}_{\varepsilon}$, conservative extensions of the well-known *ioco* relation [11], which are parameterized with a tolerance ε. An implementation conforms to a specification as long as it is functionally correct (i.e. delivers only outputs that are expected) and deviates in the quantitative part by at most ε. The presented theory relies on so-called *distance functions* [4], or distances. These distances, defined on the actions, traces and QTS, measure how far one action, trace or QTS lies apart from another. Our testing scenario finds out how far an IUT is from conforming to the specification: We show that, if every output generated by the IUT lies closely to a output one expects, then the distance (formalized by a quantitative notion of the conformance relation $\mathbf{qioco}_{\varepsilon}$ will be small, otherwise it will be large. The definition of $\mathbf{qioco}_{\varepsilon}$ relies on what trace distance is used. More than one reasonable choice seems to exist. In Figure 1 we see the classical testing framework for conformance testing, as it is for example formalized in the *ioco* theory [11]. The tester has access to the specification, and sends inputs derived from the specification to the implementation. The implementation responds with one or more outputs (or no output at all). The tester checks whether the received (lack of) outputs is correct according to specification. For our quantitative testing framework we assume that specification and implementation can be modeled by QTS, and that inputs are of the form $i?(x)$ and outputs of the form $o!(y)$; i and o indicate the type of input or output, and x, y the quantitative information assigned to it. In Figure 1 (b) we extend the previous scenario with two black boxes which represent the perturbances inputs and outputs can become subjected to. The sources of the perturbances are of no relevance, but we assume that inputs are perturbed by at most ε', outputs by at most ε ($\varepsilon, \varepsilon' \in [0, 1]$). The perturbances have an influence on the real and the perceived behavior of the implementation. An input $i?(x)$ sent to the implementation might actually be interpreted as an input $i?(x')$ with $x' \neq x$, where $|x' - x| \leq \varepsilon'$. The implementation might then produce an output $o!(y)$, which is then perturbed itself and arrives as $o!(y')$ at the tester, where $|y - y'| \leq \varepsilon$. This picture allows for different approaches to testing, which corresponds to different quality on the system to be tested. One view is to see the implementation together with the perturbances inside a black box, which makes it impossible to know how large ε and ε' are. However, the testing objective here is to find out if the complete black box conforms, i.e. if the deviations seen in output are within the tolerated limits. In this scenario the tester would send inputs that are correct according to the specification, observe outputs that are sent back, measure the deviation of the received to the expected outputs according to the specification, and base its verdict on these deviations. A second scenario is to assume that the tester has actually unperturbed access to the implementation itself. However, the implementation might be deployed in an environment in which inputs and outputs *are* perturbed by ε and ε', respectively. The testing objective might then be to find out how the implementation reacts to perturbations in the input. This would require that the tester sends inputs to the implementation that are deliberately perturbed and deviate from the inputs prescribed by the specification. By testing it could then, for example, be established that a perturbation of inputs by at most ε causes the implementation to produce outputs that are deviating by more than ε (which could be seen as a reason to fail the test).

In the reminder of this paper we show that both approaches to testing can be described in a single theory, and it is the choice of the *trace distance* which makes the difference. For that reason we keep the definition of $\mathbf{qioco}_\varepsilon$ parametric, i.e. define a $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$, where $\mathcal{D}$ is the trace distance used to measure deviations in quantitative information. In the scope of this paper, two trace distances are introduced that corresponds to the two scenarios sketched above. We also define an on-the-fly testing algorithm which again is parametric in the chosen trace distance. We show that the algorithm is sound and exhaustive w.r.t. to the $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ relations.

Structure of the paper. Section 2 introduces the necessary technical preliminaries on distances and QTS. In Section 3 we define the $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ relations and analyze some of their properties. In Section 4 we introduce the on-the-fly testing algorithm for $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ and prove its soundness and exhaustiveness. We conclude with Section 5.

2 Preliminaries

2.1 Distances and Hausdorff distances

Let X be a set. A distance[1] on X is a non-negative function $d : X \times X \to \mathbb{T}$, where $\mathbb{T}$ is a totally ordered dense set with $+$ containing 0 and for with $d(x, x) = 0$ and $d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality) holds. If d is a distance on X, then d can be lifted to sets by the Hausdorff distance h^d: for sets $Y, Z \subseteq X$, h^d is defined as $h^d(Y, Z) = \sup_{y \in Y} \inf_{z \in Z} d(y, z)$. Thus, for every $y \in Y$, the minimal distance to a $z \in Z$ is derived, and from these minimal distances the maximum is chosen. Note that the a Hausdorff distance h^d is in general not symmetric, even if d is so. To cover empty sets, we define for f a function, $\sup_{x \in \emptyset} f(x) = 0$ and $\inf_{x \in \emptyset} f(x) = \top$.

For $Y \subseteq X$, the ε-ball $B^d(Y, \varepsilon)$ around Y, containing all elements within distance ε from some element in Y, is defined as $B^d(Y, \varepsilon) = \{x \in X \mid \exists y \in Y : d(x, y) \leq \varepsilon\}$. For $Y, Z \subseteq X$, set inclusion can be expressed as $Y \subseteq Z = \forall y \in Y : \exists z \in Z : y = z$. If distance d is defined on X, then this can be rephrased as $Y \subseteq Z = \forall y \in Y : \exists z \in Z : d(y, z) = 0$. A natural generalization of set inclusion is then $Y \subseteq_\varepsilon^d Z \hat{=} \forall y \in Y : \exists z \in Z : d(y, z) \leq \varepsilon$. It is straightforward to show that $Y \subseteq_\varepsilon^d Z$ if and only if $h^d(Y, Z) \leq \varepsilon$. The following lemma gives a third characterization of $\subseteq_\varepsilon^d$ in terms of (ordinary) set inclusion.

[1] Sometimes also called *quasi-pseudo metrics*.

Lemma 1. *Let $d : X^2 \to \mathbb{R}$ be a distance and $Y, Z \subseteq X$. Then $Y \subseteq^d_\varepsilon Z$ if and only if $Y \subseteq B^d(Z, \varepsilon)$.*

Proof. See Appendix A.

2.2 Quantitative transition systems

This section introduces *quantitive transition systems* (QTS), which are labeled transition systems whose actions $a(x)$ consist of a label a and a value x in $[0, 1]$.

Definition 1. *A* quantitative transition system Q *is a tuple* $\langle S, S^0, L, \rightarrow\rangle$*, where*

- S *is a set of states (possibly uncountable);*
- $S^0 \subseteq S$ *is a set of initial states;*
- L *is a a set of action labels, which is partitioned into two sets* (L_I, L_O) *of* input *and* output *labels respectively. We write* $A = L \times [0, 1]$ *and pairs* $(a, x) \in A$ *as* $a(x)$*; We write* $A_I = L_I \times [0, 1]$ *and* $A_O = L_O \times [0, 1]$*;*
- $\rightarrow \subseteq S \times A \times S$ *is the transition relation.*

For states $s, s' \in S$*, we write* $s \xrightarrow{a} s'$ *for* $(s, a, s') \in \rightarrow$*.*

We use the following notational convention: if we refer to actions $a, b, c, \ldots$, then $a, b, c \in A$. If we refer to $a(x), b(y), c(z), \ldots$, then $a, b, c \in L$ and $x, y, z \in [0, 1]$. Note that we do not consider τ or hidden actions. For $s \in S$, we write $s \xrightarrow{a(x)}$, if $\exists s' \in S : s \xrightarrow{a(x)} s'$. We assume all QTS $\langle S, S^0, L, \rightarrow\rangle$ to be *non-blocking on outputs*, i.e. for every state $s \in S$, there is an action $a \in A_O$ such that $s \xrightarrow{a}$. Note that it is straightforward to transform any QTS into a non-blocking one: extend L_O with a fresh label δ (representing *quiescence*, or output-inactivity). For a state $s \in S$ which is blocking on outputs (i.e. without any outgoing output-transition), we add the transition $s \xrightarrow{\delta(0)} s$ to the transition relation. This construction is analogous to the suspension-automaton in (c.f. [11]).

Definition 2 (Determinism). *A QTS* $Q = \langle S, S^0, L, \rightarrow\rangle$ *is said to be* deterministic *iff for* $s, s', s'' \in S, a \in A$*:* $s \xrightarrow{a} s'$ *and* $s \xrightarrow{a} s''$ *implies* $s' = s''$*;* Q *is* input-enabled *iff for all* $s \in S$ *and all* $a \in A_I$ *we have* $s \xrightarrow{a}$*.*

Definition 3 (Traces). *An* execution fragment *of* Q *is a finite sequence* $\nu = s_0 a_1 s_1 a_2 s_2 \ldots s_n$ *such that* $(s_{i-1}, a_i, s_i) \in \rightarrow$ *for all* $1 \leq i \leq n$*. The* trace *of* ν *is obtained by removing all states in* ν*, i.e.* $trace(\nu) = a_1 a_2 \cdots a_n$*. We then write* $s_0 \xrightarrow{a_1 a_2 \cdots a_n} s_n$*. We denote by* $tr(Q) \subseteq A^*$ *the set of all traces* σ *of* Q *starting in a starting state. Given a trace* $\sigma = a_1 a_2 \cdots a_n$*, we denote its length* n *by* $|\sigma|$ *and write* σ_i *for* i^{th} *symbol in* σ *and* $\sigma^i = \sigma_i \sigma_{i+1} \ldots$ *for the suffix of* σ *starting from* σ_i*. We define* λ *to be the* empty trace.

2.3 Trace distances

In order to quantify how far an execution of an implementation is off from a specification, we introduce several distances on sets of traces [4]. For our purposes, distances take values $x \in [0, 1]_\top := [0, 1] \cup \top$. The $\top$ element is used to express incomparability between actions, and we define $\forall x \in [0, 1] : x < \top$. To define the trace distances, we define first distances on (sets of) actions and lift these on the set of traces. In general, the distance between sets that we use here are Hausdorff distances.

Definition 4. ***(Action Distances)*** *We define* action distances ad^I, ad^O, ad_c^I, *and* ad_c^O. *Let* $\dagger \in \{I, O\}$. *Then*

1. $ad^\dagger$ *is defined as*

$$ad^\dagger(a(x), b(y)) = \begin{cases} |x-y| & \text{if } a = b \text{ and } \{a,b\} \subseteq L_\dagger, \\ 0 & \text{if } a = b \text{ and } \{a,b\} \not\subseteq L_\dagger \\ \top & \text{otherwise.} \end{cases}$$

2. $ad_c^\dagger$ *(the* constrained *action distance), is defined as*

$$ad_c^\dagger(a(x), b(y)) = \begin{cases} |x-y| & \text{if } a = b \text{ and } \{a,b\} \subseteq L_\dagger, \\ 0 & \text{if } a(x) = b(y), \\ \top & \text{otherwise.} \end{cases}$$

   All distances derived from $ad_c^\dagger$ *are marked with subscript* $\cdot_c$.
3. *For* $d \in \{ad^\dagger, ad_c^\dagger\}$, $E, E' \subseteq A$: $d(E, E') = \sup_{a \in E} \inf_{b \in E'} d(a, b)$.

The action distance ad^O and ad_c^O measure the distances between output actions: for $o(x), o(y) \in A_O : ad^O(o(x), o(y)) = ad_c^O(o(x), o(y)) = |x - y|$. They differ in the way how input actions are compared: for $i(x), i(y) \in A_I$, $ad^O(i(x), i(y)) = 0$, regardless of the values of x, y. However, $ad_c^O(i(x), i(y)) = 0$ only if $x = y$, and $\top$ otherwise. The same holds dually for ad^I and ad_c^I. For all actions distances it holds that they result in $\top$ if the labels of the compared actions do not match. For $Y = \{o(x), i(y)\}, Z = \{o(x'), i(y')\}$ with $y \neq y'$ it holds that $ad^O(Y, Z) = |x - y|$, whereas $ad_c^O(Y, Z) = \top$.

Using the action distances, we define now distances on traces.

Definition 5. ***(Trace Distances)***

1. *For traces* $\sigma = a_1 \cdots a_n$ *and* $\rho = b_1 \cdots b_m$, *and* $\dagger \in \{I, O\}$, *we define* $td^\dagger(\sigma, \rho) = \max_{1 \le i \le n} ad^\dagger(a_i, b_i)$ *if* $n = m$, *and* $td^\dagger(\sigma, \rho) = \top$, *otherwise. Then* $td(\sigma, \rho) = \max\{td^I(\sigma, \rho), td^O(\sigma, \rho)\}$.
2. *For* $d \in \{td^I, td^O, td\}$, *and* P, Q *QTS*, $d(P, Q) = \sup_{\sigma \in tr(P)} \inf_{\rho \in tr(Q)} d(\sigma, \rho)$.
3. *The constrained trace distances,* $td_c^\dagger$ *and* td_c *, are defined as* $td^\dagger$ *and* td, *respectively, with* $ad_c^\dagger$ *taking the place of* $ad^\dagger$.

The trace distances which we will consider in this paper are td and td_c^O, and we will let the variable $\mathcal{D}$ range over $\{td, td_c^O\}$, if not indicated otherwise. The relation between these two distances is established in the following lemma.

Lemma 2. *Let* $\sigma, \rho \in A^*$. *Then*

$$td^O(\sigma, \rho) \le \varepsilon \text{ and } td^I(\sigma, \rho) \le 0 \qquad \text{if and only if} \qquad td_c^O(\sigma, \rho) \le \varepsilon.$$

Proof. See Appendix A.

Trace distances on traces with length 1 define also action distances, which are related to, but different from the ones defined above.

Lemma 3. *Let $a, b \in A$. Then*

$$\text{(i) } td_c^O(a,b) = ad_c^O(a,b) \quad \text{and} \quad \text{(ii) } td(a,b) = \begin{cases} ad^I(a,b) & \text{if } \{a,b\} \subseteq A_I \\ ad^O(a,b) & \text{if } \{a,b\} \subseteq A_O \\ \top & \text{otherwise.} \end{cases}$$

Note that $td_c^O(a,b) = td(a,b) = ad^O(a,b)$ for $\{a,b\} \subseteq A_O$.

We define now the set of states that can be reached from a starting state with a trace with maximal tolerance $\varepsilon \in [0,1]$. The definition is generic for $\mathcal{D} \in \{td, td_c^O\}$.

Definition 6. *Let $Q = \langle S, S^0, L, \rightarrow \rangle$ be a QTS, and $\mathcal{D} \in \{td, td_c^O\}$. Then, for $s \in S$, $\sigma \in A^*$ and $\varepsilon \in [0,1]$: $\underline{s \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma} = \{s' \mid \exists \rho \in A^* : s \xrightarrow{\rho} s' \wedge \mathcal{D}(\sigma, \rho) \leq \varepsilon\}$. For $S' \subseteq S$, $\underline{S' \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma} = \bigcup_{s \in S'} \underline{s \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma}$. We define $\underline{Q \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma} := \underline{S^0 \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma}$.*

The following definition defines the set of outputs thet can be executed from a set of states.

Definition 7 (Out-sets). *For QTS $Q = \langle S, S^0, L, \rightarrow \rangle$ and $S' \subseteq S$, the* out-set *of S' is defined as $out(S') = \{o \in A_O \mid \exists s \in S' : s \xrightarrow{o}\}$.*

3 Quantitative implementation relations

3.1 A quantitative generalization of the input-output refusal-relation

A frequently used formal correctness criterion for an implementation w.r.t. to a specification is to demand that every trace of the implementation is also a trace of the specification. Implementation relations for non-quantitative transition systems with inputs and outputs (a la *ioconf*, *ioco* and the I/O refusal relation) can all be formulated in terms trace inclusion. A natural adaption of this idea to quantitative systems is to replace strict set inclusion, $\subseteq$, with the quantitative version defined in Section 2.1. This idea leads us directly to the following definition.

Definition 8. *We assume a QTS $\mathcal{S}$ as specification and a QTS $\mathcal{I}$ as implementation. We assume both $\mathcal{I}, \mathcal{S}$ being input-enabled. For $0 \leq \varepsilon \leq 1$ and $\mathcal{D} \in \{td, td_c^O\}$, we define $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S} \iff \mathcal{D}(\mathcal{I}, \mathcal{S}) \leq \varepsilon$.*

Thus, we define $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S}$ as $tr(\mathcal{I}) \subseteq_\varepsilon^\mathcal{D} tr(\mathcal{S})$, and we obtain by Lemma 1 that $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S}$ iff $tr(\mathcal{I}) \subseteq B^\mathcal{D}(tr(\mathcal{S}), \varepsilon)$. If $\varepsilon = 0$, then $\sqsubseteq_\varepsilon^\mathcal{D}$ reduces to trace inclusion. Note that $\sqsubseteq_\varepsilon^\mathcal{D}$ for $\varepsilon \neq 0$ is not a preorder, since transitivity does not hold: from $\mathcal{P} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{Q}$ and $\mathcal{Q} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{R}$ we can not conclude that $\mathcal{P} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{R}$. However, the triangle inequality that holds for $\mathcal{D}$ allows us to conclude that $\mathcal{P} \sqsubseteq_{2\varepsilon}^\mathcal{D} \mathcal{R}$. The ordinary input-output refusal relation has a characterization in terms of output sets of implementation and specification. A similar characterization of $\sqsubseteq_\varepsilon^\mathcal{D}$ is also possible.

Lemma 4. *Let $\mathcal{S}, \mathcal{I}$ be two input-enabled QTS and $\mathcal{D} \in \{td, td_c^O\}$. Then*

$$\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S} \iff \forall \sigma \in A^* : out(\underline{\mathcal{I} \ \mathbf{after}_0^\mathcal{D} \ \sigma}) \subseteq_\varepsilon^\mathcal{D} out(\underline{\mathcal{S} \ \mathbf{after}_\varepsilon^\mathcal{D} \ \sigma})$$

Proof. See Appendix A.

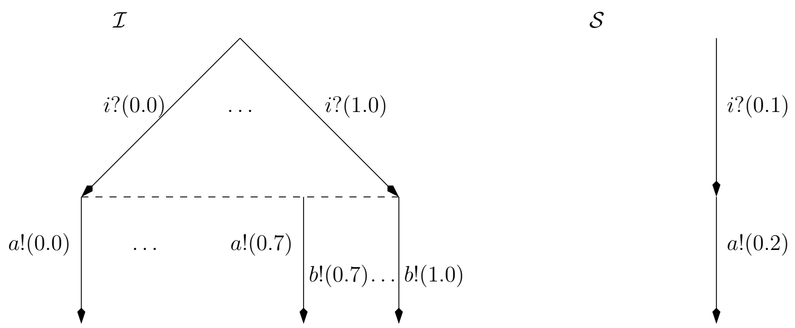

Fig. 2. Example 1

3.2 $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$

The formulation of $\sqsubseteq_\varepsilon^{\mathcal{D}}$ in terms of *out*-sets of implementation and specification allows us now to define a relation on QTS which corresponds to the *ioco* relation in the non-quantitative case. We assume again QTS $\mathcal{S}$ and $\mathcal{I}$, with $\mathcal{I}$ input-enabled.The classical way to define the non-quantitative *ioco* relation is to require inclusion of out sets not for all possible words $\sigma \in A^*$, but only for traces of the specification. In the quantitative case, this restriction is too sharp. Since the idea is to cut the implementation some slack (ε, to be exact), it is necessary to consider also traces that are at most ε off from the set of traces of the specification. The idea is that a tester sends inputs that are prescribed by the specification to the IUT, and receives outputs that may or may not be off from the expected output in the specification. We will therefore restrict the set of considered traces to $B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$, i.e. to the traces that are at most ε off from the trace-set of the specification.

Definition 9. *$\mathcal{I}$ $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ $\mathcal{S}$ iff $\forall \sigma \in B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$:*

$$out(\underline{\mathcal{I}\ \mathbf{after}_0^{\mathcal{D}}\ \sigma}) \subseteq_\varepsilon^{\mathcal{D}} out(\underline{\mathcal{S}\ \mathbf{after}_\varepsilon^{\mathcal{D}}\ \sigma}).$$

Clearly, the following property does hold:

Lemma 5. $\mathcal{I} \sqsubseteq_\varepsilon^{\mathcal{D}} \mathcal{S} \;\Rightarrow\; \mathcal{I}\, \mathbf{qioco}_\varepsilon^{\mathcal{D}}\, \mathcal{S}$,

by the fact that in case of $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ the *out*-set inclusion does only need to hold for a subset of A^*. Furthermore, the following holds:

Lemma 6. *Let $\mathcal{S}$ be input-enabled. Then $\mathcal{I}$ $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ $\mathcal{S}$ implies $\mathcal{I} \sqsubseteq_\varepsilon^{\mathcal{D}} \mathcal{S}$.*

Proof. See Appendix A.

Example 1. In Figure 2 we see two QTS $\mathcal{I}$ and $\mathcal{S}$, where $\mathcal{I}$ serves as implementation and $\mathcal{S}$ as specification[2]. We have $tr(\mathcal{I}) \supseteq \{i?(x) \cdot a!(x) \mid x \in [0, 0.7]\} \cup \{i?(x) \cdot b!(x) \mid x \in [0.7, 1.0]\}$. $\mathcal{I}$ implements an "echo process" which returns the input it has received, either with label $a!$ or $b!$. Specification $\mathcal{S}$ has only the trace $i?(0.1) \cdot a!(0.2)$. It turns out that $\mathcal{I}$ $\mathbf{qioco}_\varepsilon^{td_c^O}$ $\mathcal{S}$ for $\varepsilon = 0.1$, witnessed by

$$out(\underline{\mathcal{I}\ \mathbf{after}_0^{td_c^O}\ i?(0.1)}) = \{a!(0.1)\} \subseteq_{0.1}^{td_c^O} \{a!(0.2)\} = out(\underline{\mathcal{S}\ \mathbf{after}_{0.1}^{td_c^O}\ i?(0.1)}).$$

[2] Fot the sake of simplicity we do not bother to make $\mathcal{I}$ input-complete and non-blocking on outputs.

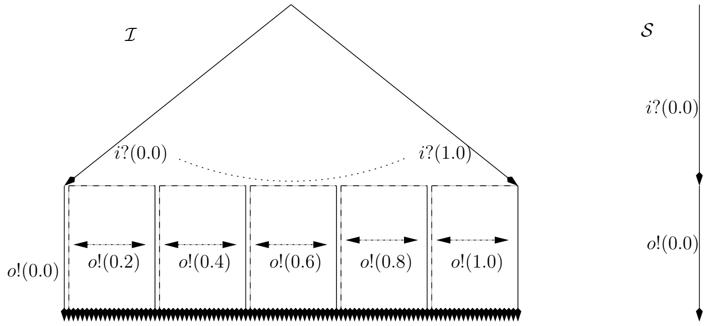

Fig. 3. Example 2

Moreover, $\mathcal{I}$ $\mathbf{qioco}_{\varepsilon}^{td_c^O}$ $\mathcal{S}$ for $0.1 < \varepsilon \leq 1$.

If we choose td as trace distance, things look differently. It is also true that $\mathcal{I}$ $\mathbf{qioco}_{\varepsilon}^{td}\,\mathcal{S}$, but only for $\varepsilon \in [0.2, 0.6]$. To demonstrate this, we first choose $\varepsilon = 0.1$. For $\sigma = i?(0.0) \in B^{td}(tr(\mathcal{S}), 0.1)$, we have $out(\underline{\mathcal{I}\ \mathbf{after}_0^{td}\ \sigma}) = \{a!(0.0)\}$ and $out(\underline{\mathcal{S}\ \mathbf{after}_{0.1}^{td}\ \sigma}) = \{a!(0.2)\}$ Certainly, $\{a!(0.0)\} \not\subseteq_{0.1}^{td} \{a!(0.2)\}$. For $\varepsilon = 0.2$, and $\sigma = i?(0.0)$, we get the same out-sets, but anyway $\{a!(0.0)\} \subseteq_{0.2}^{td} \{a!(0.2)\}$. For $\sigma = i?(0.3)$, we derive the respective out-sets $\{a!(0.3)\}$ and again $\{a!(0.2)\}$, and $\{a!(0.3)\} \subseteq_{0.2}^{td} \{a!(0.2)\}$. In case of $\varepsilon = 0.7$ and $\sigma = i?(0.8)$, we have the respective out-sets $\{b!(0.8)\}$ and $\{a!(0.2)\}$, and of course $\{b!(0.8)\} \not\subseteq_{0.7}^{td} \{a!(0.2)\}$, since the labels do not match.

The example shows that $\mathcal{I}$ $\mathbf{qioco}_{\varepsilon}^{td}\,\mathcal{S}$ does not imply that $\mathcal{I}$ $\mathbf{qioco}_{\varepsilon'}^{td}\,\mathcal{S}$ for $\varepsilon' > \varepsilon$. The underlying cause of this phenomenon is that the chosen tolerance value ε does also influence what inputs are applied to the implementation. The larger ε is, the more inputs to choose from. For a certain value ε on, inputs might be sent to the the implementation which might trigger behaviour that does not conform.

Example 2. In Figure 3,

$$\begin{aligned} tr(\mathcal{I}) \supseteq\ & \{i?(0.0)\cdot o!(0.0)\} \cup \{i?(x)\cdot o!(0.2) \mid x \in (0.0, 0.2]\} \\ & \cup \{i?(x)\cdot o!(0.4) \mid x \in (0.2, 0.4]\} \cup \{i?(x)\cdot o!(0.6) \mid x \in (0.4, 0.6]\} \\ & \cup \{i?(x)\cdot o!(0.8) \mid x \in (0.6, 0.8]\} \cup \{i?(x)\cdot o!(1.0) \mid x \in (0.8, 1.0]\}. \end{aligned}$$

and $tr(\mathcal{S}) = \{i?(0.0)\cdot o!(0.0)\}$. We have $\mathcal{I}\,\mathbf{qioco}_{\varepsilon}^{td_c^O}\ \mathcal{S}$ for $\varepsilon \in [0, 1]$, however, $\mathcal{I}\,\mathbf{qioco}_{\varepsilon}^{td}\,\mathcal{S}$ only for $\varepsilon \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

3.3 $\mathbf{qioco}^{\mathcal{D}}$ expressed as trace inclusion

Again we keep the definitions generic, assuming distance $\mathcal{D} \in \{td, td_c^O\}$. Our approach to express $\mathbf{qioco}^{\mathcal{D}}$ in terms of trace inclusion is based on demonic completion, similar to what is proposed in [12]. The idea is to manipulate the specifications such that they become input-enabled, yet retain basically all the information w.r.t. their underspecification. For this to work we must assume that the considered QTS have a certain structure (are "well-formed").

Definition 10 (well-formedness). *Let $Q = \langle S, S^0, L, \rightarrow\rangle$ be a QTS (not necessarily input-complete). We say that Q is well-formed, iff the following holds:*

$$\forall \sigma \in A^* : s, s' \in \underline{Q\ \mathbf{after}_0^{\mathcal{D}}\ \sigma} \textit{ implies } \forall a \in A_I : \quad s \xrightarrow{a} \textit{ iff } s' \xrightarrow{a}$$

Note that a well-formed QTS is not necessarily deterministic. However, obviously all deterministic QTS are well-formed.

Definition 11 (Γ-Closure). *Let $Q = \langle S, S^0, L, \rightarrow\rangle$ be a well-formed QTS. We define $\Gamma(Q) = \langle S', S^{0\prime}, L', \rightarrow'\rangle$, the* Γ-closure of Q, *as follows:*

1. $S' = S \cup \{s_\Gamma\}$, *where we require that* $s_\Gamma \notin S$.
2. $S^{0\prime} = S^0$
3. $L' = L$
4. $\rightarrow' = \quad \{(s, a, s_\Gamma) \mid s \in S, a \in A_I, s \overset{a}{\not\rightarrow}\}$
   $\cup \{(s_\Gamma, a, s_\Gamma) \mid a \in A\}$

We call $\Gamma(Q)$ the Γ-closure of Q, and call s_Γ the *garbage collector* (thus the Γ). Note that $\Gamma(Q)$ is input-enabled. The well-formedness of Q is crucial for the Γ-closure, which becomes apparent in the following lemma.

Lemma 7. *Let Q be a well-formed QTS. Then $\forall \sigma \in A^*$:*

$$s_\Gamma \in \underline{\Gamma(Q) \ \mathbf{after}_0^{\mathcal{D}}\ \sigma} \text{ implies } \underline{\Gamma(Q) \ \mathbf{after}_0^{\mathcal{D}}\ \sigma} = \{s_\Gamma\}.$$

Proof. See Appendix A.

The lemma states that, if a trace leads to the garbage collector, then the garbage collector is the only state to be reached with this trace. The other direction of the implication is of course vacuously true.

The definition of $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ makes use of the set $B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$. In order to express $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ in terms of trace inclusion, we must assume the existence of a QTS $B_\varepsilon^{\mathcal{D}}(\mathcal{S})$ such that $tr(B_\varepsilon^{\mathcal{D}}(\mathcal{S})) = B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$.

Definition 12. *Let $Q = \langle S, S^0, L, \rightarrow\rangle$ be a QTS. Then we denote by $B_\varepsilon^{\mathcal{D}}(Q)$ the QTS $(S', S^{0\prime}, L, \rightarrow')$, where*

1. $S' = S$,
2. $S^{0\prime} = S^0$,
3. $\rightarrow' \subseteq S' \times A \times S'$ *is the smallest set fulfilling the following property:* $s \xrightarrow{a} s'$ *implies* $s \xrightarrow{b}{}' s'$ *for all* $b \in A$ *with* $\mathcal{D}(a, b) \leq \varepsilon$.

Lemma 8. $tr(B_\varepsilon^{\mathcal{D}}(\mathcal{S})) = B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$.

Proof. Can be shown by induction in a straightforward manner.

The next lemma provides the characterization of $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ in terms of trace inclusion.

Lemma 9. *Let $\mathcal{I}$ be an input-enabled QTS and $\mathcal{S}$ a well-formed one. Then*

$$\mathcal{I} \ \mathbf{qioco}_\varepsilon^{\mathcal{D}}\ \mathcal{S} \quad \text{if and only if} \quad tr(\mathcal{I}) \subseteq tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S}))).$$

Proof. See Appendix A.

$\Gamma(\cdot)$ is introducing traces which are actually only needed for technical reasons, not because they have a special significance for testing: these are traces which would during actual testing never occur due to the underspecification of the specification. In the next step we will remove these traces therefore again.

Apparently, $B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon) \subseteq tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. We define $T_\varepsilon = tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S}))) \setminus B^{\mathcal{D}}(\mathcal{S}, \varepsilon)$. Then $tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S}))) \setminus T_\varepsilon = B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$.

Lemma 10. $\mathcal{I}$ $\mathbf{qioco}_\varepsilon^{\mathcal{D}}$ $\mathcal{S}$ *iff* $(tr(\mathcal{I}) \setminus T_\varepsilon) \subseteq_\varepsilon^{\mathcal{D}} tr(\mathcal{S})$.

Proof. Follows directly from Lemma 9, Lemma 1 and the definition of set T_ε.

3.4 The $\mathbf{qioco}^{\mathcal{D}}$ distance

The definition of the $\mathbf{qioco}_{\varepsilon}^{\mathcal{D}}$ relation in Section 3.2 is dissatisfactory in the sense that, for given $\mathcal{I}$ the implementation and $\mathcal{S}$ the specification, it lacks an indication of the minimal ε such that $\mathcal{I}\,\mathbf{qioco}_{\varepsilon}^{\mathcal{D}}\,\mathcal{S}$. It would be desirable to have a distance function $d_{qioco}^{\mathcal{D}}$ which actually measures the distance between $\mathcal{I}$ and $\mathcal{S}$. This function can be defined readily enough.

Definition 13 **($d_{qioco}^{\mathcal{D}}$).** *Let $\mathcal{I}$ be an input-enabled QTS and $\mathcal{S}$ a QTS. Then we define:*

$$d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S}) = \inf\{\varepsilon \in [0,1]_{\top} \mid \mathcal{I}\ \mathbf{qioco}_{\varepsilon}^{\mathcal{D}}\ \mathcal{S}\}.$$

It is desirable to express $d_{qioco}^{\mathcal{D}}$ also in terms of distances between sets of traces. This, however, is is in general not possible. Lemma 10 illustrates the problem: although $\mathbf{qioco}_{\varepsilon}^{\mathcal{D}}$ is defined in terms of trace inclusion (i.e. $(tr(\mathcal{I}) \setminus T_{\varepsilon}) \subseteq B^{\mathcal{D}}(tr(\mathcal{S}),\varepsilon)$) all we can say is that $\mathcal{D}((tr(\mathcal{I}) \setminus T_{\varepsilon}), tr(\mathcal{S})) = \varepsilon$. This is not satisfactory, since there still might be a $\varepsilon' < \varepsilon$ with $\mathcal{I}\ \mathbf{qioco}_{\varepsilon'}^{\mathcal{D}}\ \mathcal{S}$. The reason for this is that set $tr(\mathcal{I}) \setminus T_{\varepsilon}$ is also depending on ε.

Although a more illuminating characterisation of $d_{qioco}^{\mathcal{D}}$ seems to elude us, a different formulation of the above definition sheds light on how we can approximate $d_{qioco}^{\mathcal{D}}$ by means of *testing*. Another way to formulate $d_{qioco}^{\mathcal{D}}$ is as follows:

$$d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S}) = \sup\{\ \varepsilon\ \in [0,1]_{\top} \mid \forall\varepsilon' < \varepsilon : \mathcal{I}\ \mathbf{qio\not{c}o}_{\varepsilon'}^{\mathcal{D}}\ \mathcal{S}\}.$$

Using Lemma 9, this can be transformed to

$$d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S}) = \sup\{\ \varepsilon \in [0,1]_{\top} \mid \forall\varepsilon' < \varepsilon : tr(\mathcal{I}) \cap \overline{tr(\Gamma(B_{\varepsilon}^{\mathcal{D}}(\mathcal{S})))} \neq \emptyset\}.$$

Thus for all $\varepsilon < d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S})$, $tr(\mathcal{I}) \cap \overline{tr(\Gamma(B_{\varepsilon}^{\mathcal{D}}(\mathcal{S})))} \neq \emptyset$, i.e. $\exists\sigma \in tr(\mathcal{I})$ which is not element of $tr(\Gamma(B_{\varepsilon}^{\mathcal{D}}(\mathcal{S})))$. A testing approach to approximate $d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S})$ is then the following: we start with $\varepsilon = 0$ and begin to synthesize a trace of the implementation by exchanging inputs and outputs between tester and implementation. Whenever we encounter a trace $\sigma \in tr(\mathcal{I})$ with $\sigma \notin tr(\Gamma(B_{\varepsilon}^{\mathcal{D}}(\mathcal{S})))$ we can conclude that the chosen ε was too small. We must then derive an $\varepsilon' > \varepsilon$ from σ such that $\sigma \in tr(\Gamma(B_{\varepsilon'}^{\mathcal{D}}(\mathcal{S})))$. With this new ε' we start testing from the beginning and synthesize another trace σ', which gives us an ε'', and so on. In this way we approximate $d_{qioco}^{\mathcal{D}}(\mathcal{I},\mathcal{S})$. In the next section we will show how this general idea can be formulated in an on-the-fly testing algorithm.

4 On-the-fly testing with QTS

In this section we present a on-the-fly testing algorithm to approximate the $\mathbf{qioco}_{\varepsilon}^{\mathcal{D}}$ distance between an input-enabled QTS $\mathcal{I}$ and a QTS $\mathcal{S}$ by means of testing.

4.1 Tracefunctions and stepwise distance measuring

In order to make the behaviour of the implementation more accessible, we introduce the concept of *trace functions*.

Definition 14 (Trace function). *Let $\mathcal{I}$ be an input-enabled QTS. A* trace function *i of $\mathcal{I}$ is a partial function $i : A^* \rightharpoonup A_O$ with the properties:*

1. $i(\lambda)\downarrow$
2. $i(\sigma)\downarrow \;\Rightarrow\; i(\sigma \cdot a)\downarrow$ *for all* $a \in A_I$
3. $i(\sigma)=o \;\Rightarrow\; i(\sigma \cdot o)\downarrow$ *and for all* $o' \in A_O$ *with* $o \neq o'$ *and for all* $\rho \in A^*$: $i(\sigma \cdot o' \cdot \rho)\uparrow$
4. $i(\sigma)\downarrow \;\Rightarrow\; i(\sigma) \in out(\underline{Q\ \mathbf{after}_0^{\mathcal{D}}\ \sigma})$,

where $f(a)\downarrow$ means that f is defined for a, and $f(a)\uparrow$, if it is not.

We define $tr(i) = \{\sigma \mid i(\sigma)\downarrow\}$. The set of all trace functions of $\mathcal{I}$ is denoted as $TF(\mathcal{I})$.

There is a close relationship between the $i \in TF(\mathcal{I})$ and $tr(\mathcal{I})$.

Lemma 11. *Let $\mathcal{I}$ be an input-enabled QTS and i a* trace function *of $\mathcal{I}$. Then $tr(i) \subseteq tr(\mathcal{I})$ and $\bigcup_{i \in TF(\mathcal{I})} tr(i) = tr(\mathcal{I})$.*

Proof. See Appendix A.

In the following, we will use the trace functions $i \in TF(\mathcal{I})$ to represent the behaviour of $\mathcal{I}$. The following definition describes a way to express the distance of trace $\sigma = a_1 a_2 \cdots a_n$, $\mathcal{D}(\sigma, tr(\mathcal{S}))$, stepwise in terms of $a_1, a_2, \ldots, a_n$.

Definition 15. *Let $\mathcal{S} = \langle S, S^0, L, \rightarrow\rangle$ be a QTS, i a trace function of $\mathcal{I}$ and $\mathcal{D} \in \{td, td_c^O\}$. We define for $\mathcal{S}$ and i a family of functions, $\mathsf{curr\_dist}_\sigma^{\mathcal{D}} : S \rightarrow [0,1]_\top$ with $\sigma \in A^*, i(\sigma)\downarrow$ as follows:*

1. $\mathsf{curr\_dist}_\lambda^{\mathcal{D}}(s) = 0$ *if* $s \in S^0$, *and* $\top$ *otherwise;*
2. *for* $a = i(\sigma)$ *or* $a \in A_I$*:* $\mathsf{curr\_dist}_{\sigma\cdot a}^{\mathcal{D}}(s) = \inf_{s' \xrightarrow{b} s} \max\{\mathsf{curr\_dist}_\sigma^{\mathcal{D}}(s'), \mathcal{D}(a,b)\}$.

$\mathsf{curr\_dist}_\sigma^{\mathcal{D}}(s)$ is the minimal trace distance w.r.t. $\mathcal{D}$ of a trace σ from the set of traces $\{\rho \in A^* \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s\}$, as is stated in Theorem 1.

Theorem 1. $\mathsf{curr\_dist}_\sigma^{\mathcal{D}}(s) = \mathcal{D}(\sigma, \{\rho \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s\})$.

Proof. See Appendix A.

Corollary 1. $\inf_{s \in S} \mathsf{curr\_dist}_\sigma^{\mathcal{D}}(s) = \mathcal{D}(\sigma, tr(\mathcal{S}))$.

For more convenient construction of the $\mathsf{curr\_dist}$ functions we introduce the operator $\mathcal{C} : (S \rightarrow [0,1]_\top) \times A \times \{td, td_c^O\} \rightarrow (S \rightarrow [0,1]_\top)$ as follows:

$$\mathcal{C}(c, a, \mathcal{D}) = s \mapsto \inf_{s' \xrightarrow{b} s} \max\{c(s'), \mathcal{D}(a,b)\}).$$

Clearly, $\mathcal{C}(\mathsf{curr\_dist}_\sigma^{\mathcal{D}}, a, \mathcal{D}) = \mathsf{curr\_dist}_{\sigma\cdot a}^{\mathcal{D}}$.

4.2 The algorithm

The algorithm for on-the-fly testing of QTS has two parts. The first is the actual testing algorithm which synthesizes a trace of the implementation and measures the distance of this trace to the specification. The second algorihm uses the first to approximate $d_{qioco}^{\mathcal{D}}$. Again we assume that $\mathcal{I}$ is an input-enabled QTS

Algorithm 1 The distance measuring algorithm

Require: $\mathcal{S} = \langle S, S^0, L, \rightarrow\rangle$ is a QTS, i is a trace function of the IUT, $n \in \mathbb{N}$, $\mathcal{D} \in \{td, td_c^O\}$, $\varepsilon \in [0,1]$.

1: **procedure** MQOTF$(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$
2: $\quad\sigma \leftarrow \lambda$
3: $\quad cd \leftarrow \varepsilon$
4: $\quad curr\_dist = \mathsf{curr\_dist}_\lambda^{\mathcal{D}}$
5: $\quad M \leftarrow S^0$
6: $\quad n' \leftarrow 0$
7: $\quad$ **while** $cd < \top \wedge n' \leq n$ **do**
8: $\quad\quad$ $[\ \exists a \in A_I$ and $\underline{M\ \textbf{after}_{cd}^{\mathcal{D}}\ a} \neq \emptyset] \rightarrow$ **let** $a \in A_I$ with $\underline{M\ \textbf{after}_{cd}^{\mathcal{D}}\ a} \neq \emptyset$ **in**
9: $\quad\quad\quad curr\_dist \leftarrow \mathcal{C}(curr\_dist, a, \mathcal{D})$
10: $\quad\quad\quad M \leftarrow \{s \mid curr\_dist(s) \leq cd\}$
11: $\quad\quad\quad \sigma \leftarrow \sigma \cdot a$
12: $\quad\quad\quad n' \leftarrow n' + 1$
13: $\quad\quad$ **end**
14: $\quad\quad$ [**true**] $\rightarrow\ o \leftarrow i(\sigma)$
15: $\quad\quad\quad curr\_dist \leftarrow \mathcal{C}(curr\_dist, o, \mathcal{D})$
16: $\quad\quad\quad cd \leftarrow \max\{cd, \inf_{s \in S} curr\_dist(s)\}$
17: $\quad\quad\quad M \leftarrow \{s \mid curr\_dist(s) \leq cd\}$
18: $\quad\quad\quad \sigma \leftarrow \sigma \cdot o$
19: $\quad\quad\quad n' \leftarrow n' + 1$
20: $\quad\quad$ **end**
21: $\quad$ **end while**
22: $\quad$ **return**(cd, σ)
23: **end procedure**

representing the specification, and $\mathcal{S} = \langle S, S^0, L, \rightarrow\rangle$ is a QTS representing the specification.

The first algorithm is Algorithm 1. This depicts a non-deterministic procedure `mqotf`, which takes five parameters, $i, \mathcal{S}, n, \mathcal{D}, \varepsilon$. $i \in TF(\mathcal{I})$ is a trace function representing the behaviour of the implementation in this particular test run. n is the maximal number of test steps to be executed. $\mathcal{D} \in \{td, td_c^O\}$ is the distance function to be used. Finally, $\varepsilon \in [0,1]$ is a tolerance parameter which has influence on the inputs to be chosen to trigger the implementation. `mqotf` returns a tuple (cd, σ), where $\sigma \in tr(\mathcal{I})$ is the trace which was generated during testing, and $cd \in [0,1]_\top$. Later we will show that $cd = \max\{\varepsilon, \mathcal{D}(\sigma, tr(\mathcal{S}))\}$. The main purpose of `mqotf` is to construct the function $\mathsf{curr\_dist}_\sigma^{\mathcal{D}}$ step-by-step, where σ is the trace synthesized during testing and by using $\mathcal{S}$.
In lines 2–6, several local variables are initialized: σ is the trace observed so far, and is initialised with λ. cd keeps track of the lower bound of the distance of the observed trace to $tr(\mathcal{S})$ and is initialised with parameter ε. $curr\_dist$ is the current $\mathsf{curr\_dist}_\sigma$ function and is initialized with $\mathsf{curr\_dist}_\lambda$. M is the so-called *menu*, the set of states of $\mathcal{S}$ which can be reached with traces $\rho \in tr(\mathcal{S})$ such that $\mathcal{D}(\sigma, \rho) \leq cd$. M is initialized with the initial states. n' counts the number of test steps already performed, and is initialised with 0.

Lines 7 to 21 cover the main loop of `mqotf`, which is terminated if $cd = \top$ or $n' > n$. The body of the **while**-loop is a nondeterministic algorithm: execution starts either on line 8 or 14. On line 8, an input $a \in A_I$ is chosen such that $\underline{M\ \textbf{after}_{cd}^{\mathcal{D}}\ a} \neq \emptyset$. If such an a exists, $curr\_dist$ is updated, new menu M is defined, a is appended to σ, and n' increased by 1 (lines 9–12). Note that cd is not updated, since $\sigma \cdot a$ has the same trace distance to $tr(\mathcal{S})$ as σ. This is ensured

Algorithm 2 Approximating $d^{\mathcal{D}}_{qioco}(\mathcal{I},\mathcal{S})$

Require: $\mathcal{S} = \langle S, S^0, L, \rightarrow\rangle$ is a QTS, $\mathcal{I}$ an input-enabled QTS, $n \in \mathbb{N}$, $\mathcal{D} \in \{td, td^O_c\}$.
1: $n' \leftarrow 0$
2: $cd \leftarrow 0$
3: $\sigma \leftarrow \lambda$
4: **while** $n' \leq n$ and $cd < \top$ **do**
5:     [**true**] $\rightarrow$ **let** $i \in TF(\mathcal{I}), m \in \mathbb{N}$ **in**
6:         $(cd, \sigma) \leftarrow \texttt{mqotf}(i, \mathcal{S}, m, \mathcal{D}, cd)$
7:         $n' \leftarrow n' + 1$
8:     **end**
9: **end while**

by the condition on the choice of a on line 8. If execution continues with line 14, rather than 8, the output $i(\sigma)$ is used to update *curr_dist*, cd, M and σ. Note that cd is only increased if $\mathcal{D}(\sigma \cdot o, tr(\mathcal{S}))$ is larger than ε. Once the **while**-loop terminates, line 22 is reached. The computed distance cd, together with σ is then returned.

`mqotf` returns (cd, σ), i.e. the trace distance of one trace only. Assuming that $cd \geq \mathcal{D}(\sigma, tr(\mathcal{S}))$ (this is shown in Section 4.3), `mqotf` can be used to approximate $d^{\mathcal{D}}_{qioco}(\mathcal{I},\mathcal{S})$, as it has been sketched in Section 3.4 and is worked out in Algorithm 2. There, we have again an $n \in \mathbb{N}$, which bounds the number of test runs to be executed, and the usual $\mathcal{S}, \mathcal{I}$ and $\mathcal{D}$. The approximation takes place in the while-loop between lines 5 and 7. In each run through the loop, an $m \in \mathbb{N}$ is chosen, which is used to restrict the length of the test run. Moreover, a trace function $i \in TF(\mathcal{I})$ is chosen nondeterministically from $TF(\mathcal{I})$. This choice reflects the fact that in each test run the implementation $\mathcal{I}$ might actually behave differently from a previous test run, even if the same inputs are applied. `mqotf` is called with the current value of cd as tolerance parameter, initially 0. The value of cd is constantly updated with the distance computed by `mqotf`.

4.3 Soundness and exhaustiveness of `mqotf`

The procedure `mqotf` (Algorithm 1) is sound w.r.t. $\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}$, for $\mathcal{D} \in \{td, td^O_c\}$. Soundness means that, whenever $\mathcal{I}\, \mathbf{qioco}^{\mathcal{D}}_{\varepsilon}\, \mathcal{S}$ than for all $n \in \mathbb{N}$, $i \in TF(\mathcal{I})$ and possible return values (cd, σ) from $\texttt{mqotf}(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$, $cd = \varepsilon$ holds. Moreover, the algorithm is exhaustive, i.e. if $\mathcal{I}\, \not\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}\, \mathcal{S}$, then there is a trace function $i \in TF(\mathcal{I})$ and a run of $\texttt{mqotf}(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$ with return value (cd, σ) such that $cd > \varepsilon$.

Integral part of a soundness proof is to show that the following property of Algorithm 1 holds: whenever execution reaches line 7 (begin of while loop), it holds:

1. $curr\_dist = \mathsf{curr\_dist}^{\mathcal{D}}_{\sigma}$;
2. $cd = \max\{\mathcal{D}(\sigma, tr(\mathcal{S})), \varepsilon\}$;
3. $M = \{s \mid curr\_dist(s) \leq cd\}$;
4. $n' \leq n + 1$.

These conditions are easily verified when line 7 is entered for the first time. Then $\sigma = \lambda, cd = \varepsilon$ $(\mathcal{D}(\lambda, tr(\mathcal{S})) = 0)$, $curr\_dist = \mathsf{curr\_dist}^{\mathcal{D}}_{\lambda}, M = S^0 = \{s \mid curr\_dist(s) = 0\}$, and $n' = 0$.

If we assume that all four condition hold and additionally $M \neq \emptyset$ and $n' \neq n+1$, the loop body is entered, and a non-deterministic choice has to be made on either to continue with line 8 or line 14.

If the precondition of line 8 holds and the line is nondeterministically chosen, then action $a \in A_I$ is the input selected to be sent to the implementations (which is only implicitly done by appending a to σ). In line 9, *curr_dist* is updated. From the definition of $\mathcal{C}$ it is easy to see that then $curr\_dist = \mathsf{curr\_dist}^{\mathcal{D}}_{\sigma\cdot a}$ on line 10. Important to note is that in lines 9–12 the value of *cd* is not updated. The reason is that in fact $\inf_{s\in S} \mathsf{curr\_dist}^{\mathcal{D}}_{\sigma\cdot a}(s) = \inf_{s\in S} \mathsf{curr\_dist}^{\mathcal{D}}_{\sigma}(s)$, since the input a is chosen to not deviate more than *cd* from the specified inputs. The trace distance of $\sigma \cdot a$ to $\mathcal{S}$ is therefore equal to that of σ. When we return from line 12 to line 7, the four conditions are thus still satisfied.

If line 14 is chosen, output o is received from the implementation (symbolized by consulting the trace function). In line 15, *curr_dist* is updated from $\mathsf{curr\_dist}^{\mathcal{D}}_{\sigma}$ to $\mathsf{curr\_dist}^{\mathcal{D}}_{\sigma\cdot o}$. In line 15, *cd* is updated. By precondition and Theorem 1, then $cd = \max\{\max\{\varepsilon, \mathcal{D}(\sigma, tr(\mathcal{S})), \mathcal{D}(\sigma \cdot o, tr(\mathcal{S}))\}\} = \max\{\varepsilon, \mathcal{D}(\sigma \cdot o, tr(\mathcal{S}))\}$. In the remaining lines until line 19, the remaining variables are updated. Clearly, on return to line 7, the four conditions hold again.

The fact that these conditions hold also once line 22 is reached allows the conclusion that, once `mqotf` returns a result (cd, σ), then $cd = \max\{\varepsilon, \mathcal{D}(\sigma, tr(\mathcal{S}))\}$.

Proving soundness. To prove now the soundness of Algorithm 1, we assume that $\mathcal{I}\,\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}\,\mathcal{S}$, but that a run of $\mathtt{mqotf}(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$ for $i \in TF(\mathcal{I})$ returns (cd, σ) with $cd > \varepsilon$. We know then that $cd = \mathcal{D}(\sigma, tr(\mathcal{S}))$. Then there is also a prefix $\sigma' \cdot o$ of σ such that $\mathcal{D}(\sigma', tr(\mathcal{S})) \leq \varepsilon$, but $\mathcal{D}(\sigma' \cdot o, tr(\mathcal{S})) > \varepsilon$. Clearly, $o \in A_O$, since, as shown above, only outputs can increase the distance of the computed trace to $tr(\mathcal{S})$. Then $\sigma' \in B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$, and $o \in out(\underline{\mathcal{I}\ \mathbf{after}^{\mathcal{D}}_0\ \sigma'})$. But this implies also that $out(\underline{\mathcal{I}\ \mathbf{after}^{\mathcal{D}}_0\ \sigma'}) \not\subseteq^{\mathcal{D}}_{\varepsilon} out(\underline{\mathcal{S}\ \mathbf{after}^{\mathcal{D}}_0\ \sigma'})$, i.e. a contradiction to the assumption $\mathcal{I}\,\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}\,\mathcal{S}$.

Proving exhaustiveness. We have to show that, if $\mathcal{I}\,\mathbf{qio\not{c}o}^{\mathcal{D}}_{\varepsilon}\,\mathcal{S}$, then there is a trace function $i \in TF(\mathcal{I})$ and a run of $\mathtt{mqotf}(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$ with returns value (cd, σ) such that $cd > \varepsilon$. $\mathcal{I}\,\mathbf{qio\not{c}o}^{\mathcal{D}}_{\varepsilon}\,\mathcal{S}$ implies according to the definition of $\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}$ that $\exists \sigma \in B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$ such that $out(\underline{\mathcal{I}\ \mathbf{after}^{\mathcal{D}}_0\ \sigma}) \not\subseteq^{\mathcal{D}}_{\varepsilon} out(\underline{\mathcal{S}\ \mathbf{after}^{\mathcal{D}}_{\varepsilon}\ \sigma})$. There is thus an output $o \in out(\underline{\mathcal{I}\ \mathbf{after}^{\mathcal{D}}_0\ \sigma})$ such that $\mathcal{D}(\{o\}, out(\underline{\mathcal{S}\ \mathbf{after}^{\mathcal{D}}_{\varepsilon}\ \sigma})) > \varepsilon$, and moreover, $\mathcal{D}(\sigma \cdot o, tr(\mathcal{S})) > \varepsilon$.

This implies that $\{\sigma, \sigma \cdot o\} \subseteq tr(\mathcal{I})$, i.e. there is also a trace function $i \in TF(\mathcal{I})$ with $\sigma \in tr(i)$ and $i(\sigma) = o$. Let $n = |\sigma|$. Since $\sigma \in B^{\mathcal{D}}(tr(\mathcal{S}), \varepsilon)$, we can assume that there is a run through $\mathtt{mqotf}(i, \mathcal{S}, n, \mathcal{D}, \varepsilon)$ such that we enter line 8 of Algorithm 1 with the following conditions fulfilled:

1. $curr\_dist = \mathsf{curr\_dist}^{\mathcal{D}}_{\sigma}$;
2. $cd = \varepsilon \geq \mathcal{D}(\sigma, tr(\mathcal{S}))$;
3. $M = \{s \mid curr\_dist(s) \leq \varepsilon\}$;
4. $n' = n$.

If the algorithm proceeds then to line 14, trace function i will return output o, *curr_dist* will be updated to $\mathsf{curr\_dist}^{\mathcal{D}}_{\sigma\cdot o}$ and *cd* to $\max\{\varepsilon, \inf_{s\in S} curr\_dist(s)\} =$

$\max\{\varepsilon, \mathcal{D}(\sigma \cdot o, tr(\mathcal{S})\} = \mathcal{D}(\sigma \cdot o, tr(\mathcal{S}))$. Thus $cd > \varepsilon$. Since n' will be updated to $n+1$, the algorithm will terminate and return with $(cd, \sigma \cdot o)$, where $cd > \varepsilon$. This was to be shown.

Approximating $d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$. If Algorithm 2 is executed with parameter $n \in \mathbb{N}$, then for the value of variable cd it holds $cd < d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$. Increasing $n \to \infty$ does not guarantee that $cd \to d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$. The reason is that in general the set $TF(\mathcal{I})$ is uncountable (since $[0,1]$ is uncountable) and Algorithm 2 can even in infinite time only test against a countable number of trace functions. The test runs of `mqotf` might then well give results that converge against an $\varepsilon < d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$. It is however possible to show that at least there *exists* always a series of trace functions and results of `mqotf` which converge to $d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$ in the limit. We show this in the following.

Let $d = d^{\mathcal{D}}_{qioco}(\mathcal{I}, \mathcal{S})$. If $d = 0$, then there is nothing to show: since `mqotf` is sound, for all runs of Algorithm 2, in the end $cd = 0$. If we assume that $d > 0$, then $\mathcal{I}\, \mathbf{qioco}^{\mathcal{D}}_{d}\, \mathcal{S}$, but $\forall \varepsilon < d$: $\mathcal{I}\, \not\mathbf{qioco}^{\mathcal{D}}_{\varepsilon}\ \mathcal{S}$.

Then there is an increasing series $\langle d_j \rangle_{j=0,\ldots\infty}$ with $d_0 = 0$ and $\lim_{j\to\infty} d_j = d$. Then $\mathcal{I}\, \not\mathbf{qioco}^{\mathcal{D}}_{d_j}\ \mathcal{S}$ for all $d_j < d, j = 1, 2, 3, \ldots$. Since `mqotf` is sound and exhaustive, we can find for each d_j a trace function i_j and a run of $\mathtt{mqotf}(i_j, \mathcal{S}, m, \mathcal{D}, d_j)$ with outcome (cd_j, σ_j) such that $d_j \leq cd_j \leq d$. Then $\lim_{j\to\infty} cd_j = d$, and $\langle cd_j \rangle_{j=0,\ldots,\infty}$ corresponds to the intermediate values of variable cd of a run of Algorithm 2, which approximates d.

5 Conclusions and further work

We introduced two *ioco*-like implementation relations for QTS, based on the two different trace distances td and td^{O}_{c}. We have introduced a algorithm for on-the-fly conformance testing of QTS and shown that this algorithm is sound w.r.t. to the two respective conformance relations.

The algorithm is unusual in the sense that it just measures the distance of the implmentation to the specification, rather than reaching a verdict. However, a *judging* testing algorithm for QTS can be easily defined by an straightforward adaption of the measuring algorithm: whenever the distance of the implementation is larger than tolerated, testing is stopped and a fail verdict returned.

The algorithm is only effective if the QTS serving as specification fullfils certain structural restrictions. In particular, the QTS state space must be finitley representable. Otherwise the function *curr_dist* is not representable. Moreover, $\cdot$ **after**$^{\mathcal{D}}$ $\cdot$ must be computable, and the choice of an input must be effectively possible. One such class of QTS is that representabla as finite *interval automata*.

The developed theory is currently abstract from any concrete application area. A most important topic to be adressed is therefore how it can be integrated into existing testing theories with concrete quantitative elements, like timed testing [1, 2] or hybrid testing [13].

References

1. H. Bohnenkamp and A. Belinfante. Timed testing with TorX. In *Proc. FME 2005*, volume 3582 of *LNCS*, pages 173–188. Springer-Verlag, 2005.

2. L. Brandán Briones and H. Brinksma. A test generation framework for quiescent real-time systems. In *Proc. FATES '04*, volume 3395 of *LNCS*, pages 64–78. Springer-Verlag, 2005.
3. C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In *FORMATS*, volume 4202 of *LNCS*, pages 143–155. Springer-Verlag, 2006.
4. L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantatative transition systems. In *Proc. ICALP'04*, volume 3142 of *LNCS*, pages 97–109. Springer–Verlag, 2004.
5. J-C. Fernandez, C. Jard, T. Jeron, and C. Viho. Using on-the-fly verification techniques for the generation of test suites. In *Proc. CAV '96*, volume 1102 of *LNCS*, pages 348–359. Springer-Verlag, 1996.
6. L. Frantzen, J. Tretmans, and T.A.C. Willemse. Test generation based on symbolic specifications. In *FATES 2004*, number 3395 in LNCS, pages 1–15. Springer-Verlag, 2005.
7. L. Frantzen, J. Tretmans, and T.A.C. Willemse. A symbolic framework for model-based testing. In *FATES/RV 2006*, number 4262 in LNCS, pages 40–54. Springer-Verlag, 2006.
8. Vineet Gupta, Thomas A. Henzinger, and Radha Jagadeesan. Robust timed automata. In *HART '97: Proceedings of the International Workshop on Hybrid and Real-Time Systems*, pages 331–345. Springer-Verlag, 1997.
9. M. Krichen and S. Tripakis. An expressive and implementable formal framework for testing real-time systems. In *Proc. TESTCOM'05*, number 3502 in LNCS. Springer-Verlag, 2005.
10. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
11. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.
12. M. v. d. Bijl, A. Rensink, and J. Tretmans. Compositional testing with ioco. In *Proc FATES '03*, volume 2931 of *LNCS*, pages 89–103. Springer-Verlag, 2004.
13. M. v. Osch. Hybrid input-output conformance and test generation. In *Proc. FATES/RV '06*, volume 4262 of *LNCS*, pages 70–84. Springer-Verlag, 2006.

A Proofs

Lemma 1. Let $d : X^2 \to \mathbb{R}$ be a distance and $Y, Z \subseteq X$. Then

$$Y \subseteq_\varepsilon^d Z \iff Y \subseteq B^d(Z, \varepsilon).$$

Proof. Let $y \in Y$. Then $\exists z \in Z : d(y, z) \leq \varepsilon$. But then $y \in B^d(Z, \varepsilon)$. Since this holds for all $y \in Y$, we have immediately $Y \subseteq B^d(Z, \varepsilon)$.

Lemma 2. Let $\sigma, \rho \in A^*$. Then

$$td^O(\sigma, \rho) \leq \varepsilon \text{ and } td^I(\sigma, \rho) \leq 0 \qquad \text{if and only if} \qquad td_c^O(\sigma, \rho) \leq \varepsilon.$$

Proof.

"⇐" : $td_c^O(\sigma, \rho) \leq \varepsilon$ implies $|\sigma| = |\rho|$ and $\forall i : 1 \leq i \leq |\sigma| : ad_c^O(\sigma_i, \rho_i) \leq \varepsilon$. Moreover, if $\sigma_i \in A_I$, then $\sigma_i = \rho_i$. We can conclude that $td^I(\sigma, \rho) = 0$ and $td^O(\sigma, \rho) \leq \varepsilon$.

"⇒": $td^I(\sigma, \rho) = 0$ and $td^O(\sigma, \rho) \leq \varepsilon$ implies $|\sigma| = |\rho|$. Then the reasoning of the "⇐" direction can be easily reversed.

Lemma 4. Let $\mathcal{S}, \mathcal{I}$ be two input-enabled QTS and $\mathcal{D} \in \{td, td_c^O\}$. Then

$$\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S} \iff \forall \sigma \in A^* : out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma}) \subseteq_\varepsilon^\mathcal{D} out(\underline{\mathcal{S}\ \mathbf{after}_\varepsilon^\mathcal{D}\ \sigma})$$

Proof. **"⇒":** Let $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S}$ and $\sigma \in A^*$. We have two cases to consider:

1. $\sigma \in tr(\mathcal{I})$. Then $out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma}) \neq \emptyset$, and for all $o \in out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma})$, $\sigma{\cdot}o \in tr(\mathcal{I})$. Therefore, there exists $\rho{\cdot}o' \in tr(\mathcal{S})$ with $\mathcal{D}(\sigma{\cdot}o, \rho{\cdot}o') \leq \varepsilon$. Then $\mathcal{D}(\sigma, \rho) \leq \varepsilon$ and $\mathcal{D}(o, o') \leq \varepsilon$. Thus, $o' \in out(\underline{\mathcal{S}\ \mathbf{after}_\varepsilon^\mathcal{D}\ \sigma})$ and therefore $out(\underline{i\ \mathbf{after}_0^\mathcal{D}\ \sigma}) \subseteq_\varepsilon^\mathcal{D} out(\underline{s\ \mathbf{after}_\varepsilon^\mathcal{D}\ \sigma})$.
2. $\sigma \notin tr(\mathcal{I})$. Then $out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma}) = \emptyset$, thus the statement is vacuously true.

"⇐": We assume the right part of the lemma to be true. We have to show that for all $\sigma \in tr(\mathcal{I})$ there exists a $\rho \in tr(\mathcal{S})$ with $\mathcal{D}(\sigma, \rho) \leq \varepsilon$, i.e. $\mathcal{D}(\sigma, \mathcal{S}) \leq \varepsilon$. We do this with induction over the length of $\sigma \in tr(\mathcal{I})$. For $\sigma = \lambda$, there is nothing to show. Assume we have shown that for $\sigma' \in tr(\mathcal{I})$, $\mathcal{D}(\sigma', \mathcal{S}) \leq \varepsilon$. We now consider $\sigma = \sigma'{\cdot}a$. If a is an input, there is nothing to show, since we assume $\mathcal{S}$ to be input enabled, i.e. $\mathcal{D}(\sigma, \mathcal{S}) = \mathcal{D}(\sigma', \mathcal{S}) \leq \varepsilon$. If $a \in out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma'})$, then there is a $o \in out(\underline{\mathcal{S}\ \mathbf{after}_\varepsilon^\mathcal{D}\ \sigma'})$ with $\mathcal{D}(a, o) \leq \varepsilon$. Also there is a $\rho' \in A^*$ such that $\rho = \rho' \cdot o \in tr(\mathcal{S})$ and $\mathcal{D}(\sigma', \rho') \leq \varepsilon$. Then $\mathcal{D}(\sigma, \rho) \leq \varepsilon$.

Lemma 6. Let $\mathcal{S}$ be input-enabled. Then $\mathcal{I}\ \mathbf{qioco}_\varepsilon^\mathcal{D}\ \mathcal{S}$ implies $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S}$.

Proof. We assume $\mathcal{I}\ \mathbf{qioco}_\varepsilon^\mathcal{D}\ \mathcal{S}$. Let $T = B^\mathcal{D}(tr(\mathcal{S}), \varepsilon)$ and $\sigma \in tr(\mathcal{I})$. We prove by induction over the length of σ that $\mathcal{D}(\sigma', tr(\mathcal{S})) \leq \varepsilon$ and $\sigma' \in T$.

For $\sigma = \lambda$ this is trivial. Let $\sigma = \sigma' \cdot a$, assuming that $\mathcal{D}(\sigma', tr(\mathcal{S})) \leq \varepsilon$ and $\sigma' \in T$ has been shown already. If $a \in A_I$, nothing has to be shown due to the input-enabledness of $\mathcal{S}$. If $a \in A_O$, then we know that $a \in out(\underline{\mathcal{I}\ \mathbf{after}_0^\mathcal{D}\ \sigma'})$. Since $\mathcal{I}\ \mathbf{qioco}_\varepsilon^\mathcal{D}\ \mathcal{S}$, and $\sigma' \in T$, $\exists a' \in out(\underline{\mathcal{S}\ \mathbf{after}_\varepsilon^\mathcal{D}\ \sigma'})$ with $\mathcal{D}(a, a') \leq \varepsilon$. Thus, there is a trace $\rho' \in tr(\mathcal{S})$ such that $\mathcal{D}(\sigma', \rho') \leq \varepsilon$ and $\rho' \cdot a' \in tr(\mathcal{S})$. Also $\mathcal{D}(\sigma, \rho' \cdot a') \leq \varepsilon$, thus also $\mathcal{D}(\sigma, tr(\mathcal{S})) \leq \varepsilon$. Also $\sigma \in T$ then. We can conclude that $\sup_{\sigma \in tr(\mathcal{I})} \mathcal{D}(\sigma, tr(\mathcal{S})) = \mathcal{D}(\mathcal{I}, \mathcal{S}) \leq \varepsilon$, i.e. $\mathcal{I} \sqsubseteq_\varepsilon^\mathcal{D} \mathcal{S}$.

Lemma 7. Let Q be a well-formed QTS. Then $\forall \sigma \in A^*$:

$$s_\Gamma \in \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma} \text{ implies } \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma} = \{s_\Gamma\}.$$

Proof. We denote by $\rightarrow$ the transition relation of Q, and by $\rightarrow'$ that of $\Gamma(Q)$. If $s_\Gamma \in \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma}$, then there is a unique prefix $\sigma' \cdot a$ of σ such that $s_\Gamma \in \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma' \cdot a}$, but $s_\Gamma \notin \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma'}$. Since s_Γ can only initially be reached by transitions labeled with inputs, a must be an input. Thus there is a $s \in \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma'}$ with $s \xrightarrow{a}' s_\Gamma$. Then $s \not\xrightarrow{a}$ in Q. Since Q is well-formed, it holds then for all $s' \in \underline{Q \textbf{ after}_0^{\mathcal{D}} \sigma'} : s' \not\xrightarrow{a}$. But then, for all $s' \in \underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma'} : s' \xrightarrow{a}' s_\Gamma$. Therefore $\underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma' \cdot a} = \{s_\Gamma\}$, and for all $\sigma'' \in A^*$, also $\underline{\Gamma(Q) \textbf{ after}_0^{\mathcal{D}} \sigma' \cdot a \cdot \sigma''} = \{s_\Gamma\}$.

Lemma 9. Let $\mathcal{I}$ be an input-enabled QTS and $\mathcal{S}$ a well-formed one. Then

$$\mathcal{I} \textbf{ qioco}_\varepsilon^{\mathcal{D}} \mathcal{S} \quad \text{if and only if} \quad tr(\mathcal{I}) \subseteq tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S}))).$$

Proof.

"If": We assume $tr(\mathcal{I}) \subseteq tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. Let $\sigma \in B^{\mathcal{D}}(\mathcal{S}, \varepsilon)$. We have to show that $out(\underline{\mathcal{I} \textbf{ after}_0^{\mathcal{D}} \sigma}) \subseteq_\varepsilon^{\mathcal{D}} out(\underline{\mathcal{S} \textbf{ after}_\varepsilon^{\mathcal{D}} \sigma})$. If $\sigma \notin tr(\mathcal{I})$, this is vacuously true. Let us thus assume that $\sigma \in tr(\mathcal{I})$. Then $\exists o \in out(\underline{\mathcal{I} \textbf{ after}_0^{\mathcal{D}} \sigma})$, and $\sigma \cdot o \in tr(\mathcal{I})$ and of course also $\{\sigma, \sigma \cdot o\} \subseteq tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. There is a $\rho \in tr(\mathcal{S})$ such that $\mathcal{D}(\sigma, \rho) \leq \varepsilon$, but also an $o' \in A_O$ such that $\rho \cdot o' \in tr(\mathcal{S})$ and $\mathcal{D}(\sigma \cdot o, \rho \cdot o') \leq \varepsilon$. Then $o' \in out(\underline{\mathcal{S} \textbf{ after}_\varepsilon^{\mathcal{D}} \sigma})$ and $\mathcal{D}(o, o') \leq \varepsilon$. Since σ was chosen arbitrarily from $B^{\mathcal{D}}(\mathcal{S}, \varepsilon)$ (and $o \in out(\underline{\mathcal{I} \textbf{ after}_0^{\mathcal{D}} \sigma})$ as well), we have $\mathcal{I} \textbf{ qioco}_\varepsilon^{\mathcal{D}} \mathcal{S}$.

"Only if": We assume $\mathcal{I} \textbf{ qioco}_\varepsilon^{\mathcal{D}} \mathcal{S}$, and let $\sigma \in tr(\mathcal{I})$. We prove by induction over the length of σ that $\sigma \in tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. For $\sigma = \lambda$ this is vacuously true. Assume now that $\sigma = \sigma \cdot a \in tr(\mathcal{I})$ (the $\notin tr(\mathcal{I})$ case is also vacuously true), and assume that we have shown already that $\sigma' \in tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. If $a \in A_I$, then $\sigma \in tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$ as well, since $\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S}))$ is input-enabled. If $a \in A_O$, then $a \in out(\underline{\mathcal{I} \textbf{ after}_0^{\mathcal{D}} \sigma'})$. We have now to distinguish two cases. First, $\sigma' \notin B^{\mathcal{D}}(\mathcal{S}, \varepsilon)$. Then $\underline{\mathcal{S} \textbf{ after}_\varepsilon^{\mathcal{D}} \sigma} = \emptyset$, which implies that $\underline{\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})) \textbf{ after}_0^{\mathcal{D}} \sigma} = \{s_\Gamma\}$. Due to the definition of the Γ-closure, $s_\Gamma \xrightarrow{o} s_\Gamma$, i.e. $\sigma \in tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$. Second, $\sigma' \in B^{\mathcal{D}}(\mathcal{S}, \varepsilon)$. Then $\exists \rho' \in tr(\mathcal{S})$ and $o' \in out(\underline{\mathcal{S} \textbf{ after}_\varepsilon^{\mathcal{D}} \sigma'})$ such that $\mathcal{D}(\sigma' \cdot o, \rho' \cdot o') \leq \varepsilon$. Then also $\sigma' \cdot o \in tr(\Gamma(B_\varepsilon^{\mathcal{D}}(\mathcal{S})))$.

Lemma 11. Let $\mathcal{I}$ be an input-enabled QTS and i a *trace function* of $\mathcal{I}$. Then $tr(i) \subseteq tr(\mathcal{I})$ and $\bigcup_{i \in TF(\mathcal{I})} tr(i) = tr(\mathcal{I})$.

Proof. The first part follows directly from the definition and the fact that $\mathcal{I}$ is input enabled. The second part follows from the fact that from each $\sigma \in tr(\mathcal{I})$ we can construct a $i \in TF(\mathcal{I})$ such that $\sigma \in tr(i)$.

Theorem 1. $\mathsf{curr\_dist}_\sigma^{\mathcal{D}}(s) = \mathcal{D}(\sigma, \{\rho \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s\})$.

Proof. Proof by induction over the length of σ:

$\sigma = \lambda$**:** For $s \in S$,

$$\mathcal{D}(\lambda, \{\rho \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s\}) = \inf_{\substack{\rho : s_0 \xrightarrow{\rho} s \\ s_0 \in S^0}} \mathcal{D}(\lambda, \rho)$$
$$= \begin{cases} \top \text{ if } s \notin S^0 \\ 0 \ \text{ otherwise,} \end{cases}$$

which coincides precisely with the definition of $\mathsf{curr\_dist}^{\mathcal{D}}_{\lambda}$.

$\sigma = \sigma' \cdot a$**:**

$$\begin{aligned}
\mathsf{curr_dist}^{\mathcal{D}}_{\sigma' \cdot a}(s) &= \inf_{s' \xrightarrow{b} s} \max\{\mathsf{curr_dist}^{\mathcal{D}}_{\sigma'}(s'), \mathcal{D}(a, b)\} \\
&= \inf_{s' \xrightarrow{b} s} \max\{\mathcal{D}(\sigma', \{\rho' \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho'} s'\}), \mathcal{D}(a, b)\} \\
&= \inf_{s' \xrightarrow{b} s} \max\{\inf_{\rho' : \exists s_0 \in S^0 : s_0 \xrightarrow{\rho'} s'} \mathcal{D}(\sigma', \rho'), \mathcal{D}(a, b)\} \\
&= \inf_{s' \xrightarrow{b} s} \inf_{\rho' : \exists s_0 \in S^0 : s_0 \xrightarrow{\rho'} s'} \max\{\mathcal{D}(\sigma', \rho'), \mathcal{D}(a, b)\} \\
&= \inf_{\rho : \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s} \mathcal{D}(\sigma, \rho) \\
&= \mathcal{D}(\sigma, \{\rho \mid \exists s_0 \in S^0 : s_0 \xrightarrow{\rho} s\})
\end{aligned}$$

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2001-01 *	Jahresbericht 2000
2001-02	Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03	Thierry Cachat: The power of one-letter rational languages
2001-04	Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05	Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03	Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04	Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05	Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06	Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07	Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08	Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09	Markus Mohnen: Interfaces with Default Implementations in Java
2002-10	Martin Leucker: Logics for Mazurkiewicz traces
2002-11	Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08	Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09	Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12	Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning
2006-13	Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-14	Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"
2006-15	Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16	Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17	Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01 *	Fachgruppe Informatik: Jahresbericht 2006
2007-02	Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations
2007-03	Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase
2007-04	Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation
2007-05	Uwe Naumann: On Optimal DAG Reversal
2007-06	Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking
2007-07	Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications
2007-08	Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches
2007-09	Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption
2007-10	Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes
2007-11	Klaus Wehrle: 6. Fachgespräch Sensornetzwerke
2007-12	Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and current programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.