

# Comparing Disjunctive Modal Transition Systems with an One-Selecting Variant<sup>\*</sup>

H. Schmidt<sup>\*</sup>, H. Fecher<sup>\*\*</sup>

*Institut für Informatik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany*

---

## Abstract

Abstract models, used for specification, analysis and verification, usually describe sets of implementations by means of a refinement relation. In the branching time setting, implementations are commonly modeled as labeled transition systems (LTS). An expressive class of abstractions for LTSs is that of disjunctive modal transition systems (DMTS), featuring may- and must transitions as well as disjunctive hypertransitions (OR). In order to describe exclusive choice adequately, we develop a variant of DMTSs called 1-selecting modal transition systems (1MTS) that, roughly speaking, interprets hypertransitions exclusively (XOR). These abstract models, DMTSs and 1MTSs, are compared with respect to their expressive power. By giving transformations or showing their non-existence, we show that the two settings can express the same sets of implementations, but 1-selecting modal transition systems have a richer refinement preorder.

*Key words:* underspecification, abstraction, refinement, expressiveness, branching time

---

## 1 Introduction

Refinement and abstraction are common concepts for the specification, verification and analysis of programs. The two notions are dual to each other: Whenever a system  $\mathcal{U}$  refines another system  $\hat{\mathcal{U}}$ , we call  $\hat{\mathcal{U}}$  an abstraction of  $\mathcal{U}$ .

---

<sup>\*</sup> This work is in part financially supported by the DFG project *Refism* (FE 942/1-1)

<sup>\*</sup> Tel.: +49-431-880-4467; fax: +49-431-880-7617

<sup>\*\*</sup> Tel.: +49-431-880-3733; fax: +49-431-880-7617

*Email addresses:* hsc@informatik.uni-kiel.de (H. Schmidt), hf@informatik.uni-kiel.de (H. Fecher).

Abstraction is a common technique, e.g., in model checking to fight the state explosion problem [1]. Refinement, e.g., is frequently used in model-driven software development, where the process of development starts with an abstract model, which is then refined in later design phases [2].

Abstractions basically stand for sets of implementations. An implementation setting consists of a class of models and an equivalence relation that defines which implementations should be identified. An abstraction setting is based on an implementation setting and consists of a class of models, a subclass of relevant models (e.g., the finite ones), a refinement preorder and an embedding of implementations into the abstract models. The models corresponding to implementations via this embedding are called *concrete*. The expressive power of abstraction settings can be characterized by the set of implementations they can describe or by taking the whole refinement preorder into account.

### 1.1 Abstraction settings

In this paper, we concentrate on branching time, which is relevant whenever nondeterminism occurs from external sources (e.g., user input) or random behavior. Therefore, labeled transition systems, together with bisimulation equivalence [3], are considered.

In order to model check liveness (resp. safety) properties, labeled transition systems, with forward (resp. backward) simulation [4,5], are used as abstractions. Since forward (resp. backward) simulation only allows to add (resp. remove) transitions, both simulation notions are not in general sound for combinations of liveness and safety properties, i.e., when such a property holds for an abstraction, it is not guaranteed that it also holds for the abstracted system. In order to obtain sound abstraction also for arbitrary branching time formulae, different abstraction settings have been introduced:

A modification of transition systems features two kinds of transitions, one transition relation to denote the steps that are mandatory for the implementation, called *must transitions*, and the other to indicate those steps which may occur, but are not necessary for the implementation, called *may transitions*. This approach was followed by Larsen and Thomsen, who introduced *modal transition systems* [6], and Dams, who called his model *mixed transition systems* [7,8]. For example, in Figure 1, the labeled transition system (a) is abstracted by combining its two states on the left, leading to the modal transition system (b): From the combined state, actions *b* and *c* are possible, but not required.

In the previous abstraction example, the transition system having only an *a*-loop is also a concrete refinement of the abstraction, although every orig-

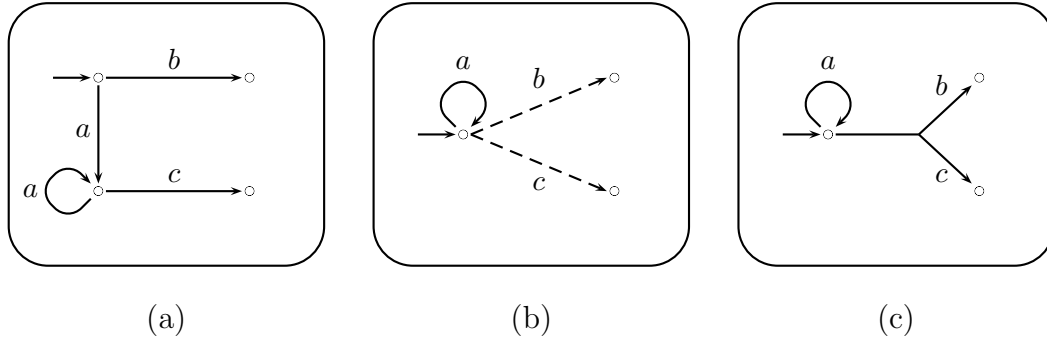


Fig. 1. A labeled transition system (a), a modal transition system (b) and a disjunctive modal transition system (c). Illustration (c) can also be interpreted as an 1-selecting modal transition system. In illustrations (b) and (c), must transitions are drawn as solid arrows, whereas may transitions are dashed. Every must transition implies a corresponding may transition, which is not drawn. Hypertransitions are graphically represented by dividing the head of an arrow such that it points to all target states.

inal state had further outgoing transitions besides  $a$ . If this refinement is unwanted, it can be excluded in the setting of disjunctive modal transition systems (DMTS), which allows *hypertransitions*. A hypertransition is a special transition starting in a single state, but ending in a set of states. For different targets of a single hypertransition different labels are allowed. Hypertransitions are interpreted disjunctively (OR interpretation), i.e., *at least* one of the targets must appear in an implementation. Combining the two states on the left of Figure 1(a) using DMTS as abstraction setting leads to the DMTS shown in Figure 1(c). Any refinement must have the  $b$  transition *or* the  $c$  transition and consequently, the system allowing neither  $b$  nor  $c$  is no refinement any more. However, the disjunctive interpretation still allows that both  $b$  and  $c$  are possible in a concrete refinement. There is no way to exclude this possibility in DMTSs without changing the state space. However, for modeling purposes it is often desirable to be able to model an exclusive choice (XOR interpretation), as illustrated by the following example: Consider a model for a soda machine, in which it is not yet specified whether it dispenses 0.4 liters or 0.5 liters of soda. A concrete machine however should always give 0.4 liters or always give 0.5 liters and not allow randomly both possibilities, which could lead to overflowing or not completely filled glasses.

## 1.2 Contribution

We develop a variant of DMTSs called *1-selecting modal transition systems* (1MTS) that interprets hypertransitions *exclusively* (XOR), i.e., *exactly* one of the targets must appear in an implementation.

With the aim of comparing the expressive power of 1MTSs and DMTSs, we formalize two different types of transformations between abstraction settings. One preserves the sets of implementations, whereas the other also takes the refinement preorders into account.

We give such transformations between the two considered abstraction settings, or show their non-existence, establishing transformations and expressiveness results. In particular, we present two transformations preserving the sets of implementations, from DMTSs to 1MTSs and vice versa. This yields that DMTSs and 1MTSs are equally expressive with respect to the describable sets of implementations.

Taking the refinement preorder into account, we show, by presenting a transformation, that DMTSs can be embedded into 1MTS. Furthermore, we prove that such a transformation from 1MTS to DMTS does not exist. Thus, with respect to the refinement preorder, 1MTSs are strictly more expressive. This indicates a richer refinement structure, which can be useful for defining sound and less approximative compositional satisfaction relations, which is however out of the scope of this paper.

### 1.3 Outline

A short presentation of required preliminaries is given in Section 2. In Section 3 a general introduction to implementation and abstraction settings is given and two types of transformations between abstraction settings are developed. Section 4 introduces the abstraction setting of DMTSs and presents a complete characterization of all refinements of a simple DMTSs. Following the same structure, Section 5 introduces the new abstraction setting of 1MTSs and presents a complete characterization of all refinements of a simple 1MTSs. In section 6, DMTSs and 1MTSs are compared by giving transformations between the settings or showing their non-existence. Section 7 addresses related work. Section 8 concludes and discusses future work. Proofs that are not straightforward are given in the Appendix.

## 2 Preliminaries

For a set  $S$ , let  $|S|$  denote the cardinality of  $S$  and let  $\mathcal{P}(S)$  denote its power set. Let  $\circ$  denote relational composition. For a binary relation  $R \subseteq S_1 \times S_2$ , we usually make use of infix notation, i.e., we write  $s_1 R s_2$  instead of  $(s_1, s_2) \in R$ . For a ternary relation  $\longrightarrow \subseteq S_1 \times L \times S_2$ , we usually write  $s_1 \xrightarrow{a} s_2$  instead of  $(s_1, a, s_2) \in \longrightarrow$ . For  $s_1 \in S_1$ , we define  $(s_1 \xrightarrow{a}) \stackrel{\text{def}}{=} \{s_2 \in S_2 \mid s_1 \xrightarrow{a} s_2\}$  and

$(s_1 \longrightarrow) \stackrel{\text{def}}{=} \{(a, s_2) \in L \times S_2 \mid s_1 \xrightarrow{a} s_2\}$ . Given a relation  $R \subseteq S \times S$  and a set  $L$ ,  $R$  is extended to  $(L \times S) \times (L \times S)$  as follows: For  $\vartheta_1 = (a_1, s'_1) \in L \times S$  and  $\vartheta_2 = (a_2, s'_2) \in L \times S$ , define  $\vartheta_1 R \vartheta_2 \stackrel{\text{def}}{=} a_1 = a_2 \wedge s'_1 R s'_2$ . For an equivalence relation  $\sim \subseteq S \times S$  and  $s \in S$ , let  $[s]_\sim$  denote the equivalence class for which  $s$  is a representative, i.e.,  $[s]_\sim \stackrel{\text{def}}{=} \{\tilde{s} \in S \mid \tilde{s} \sim s\}$ . For the definition of refinement on 1MTSs, we need the concept of choice functions:

**Definition 1 (Choice function)** *Let  $A$  be a set,  $\mathcal{B} \subseteq \mathcal{P}(A)$  and  $\gamma : \mathcal{B} \rightarrow A$ . Then  $\gamma$  is a choice function if  $\forall B \in \mathcal{B} : \gamma(B) \in B$ . We denote the set of all choice functions on  $\mathcal{B}$  by  $\text{choice}(\mathcal{B})$ .*

### 3 Implementation and abstraction settings

This section gives a formal introduction to implementation and abstraction settings. Examples of abstraction settings will follow in Sections 4 and 5, namely disjunctive and 1-selecting modal transition systems.

#### 3.1 Implementation settings

Before we start reasoning about abstractions, a precise definition of implementations is given. Implementations are described as equivalence classes on a set of models:

**Definition 2** *An implementation setting is a tuple  $(I, \equiv)$  consisting of a set of possible implementations  $I$  together with an equivalence relation  $\equiv$  on  $I$ .*

As already mentioned in the introduction, we consider the branching time setting of labeled transition systems, which will shortly be called transition systems in this paper, and bisimulation equivalence. The implementation setting considered is consequently  $(\mathbb{T}\mathcal{S}, \sim)$ , with  $\mathbb{T}\mathcal{S}$  and  $\sim$  defined as follows:

**Definition 3 (TS)** *A transition system (TS) is a tuple  $(S, L, \longrightarrow, s^0)$ , where  $S$  is a set of states,  $L$  is a set of labels,  $\longrightarrow \subseteq S \times L \times S$  is the transition relation and  $s^0 \in S$  is the root state. We denote the set<sup>1</sup> of all TSs by  $\mathbb{T}\mathcal{S}$ .*

**Definition 4 (Bisimulation)** *Let  $\mathcal{T}_1 = (S_1, L, \longrightarrow_1, s_1^0), \mathcal{T}_2 = (S_2, L, \longrightarrow_2, s_2^0) \in \mathbb{T}\mathcal{S}$ . A bisimulation between  $\mathcal{T}_1$  and  $\mathcal{T}_2$  is a relation  $R \subseteq S_1 \times S_2$  such that  $s_1^0 R s_2^0$  and for all  $(s_1, s_2) \in R$  we have the following:<sup>2</sup>*

<sup>1</sup> Strictly speaking, this is a class, but a skeletal set may be chosen. This remark also applies to DMTSs and 1MTSs, which are defined later.

<sup>2</sup> Note that in the definition of bisimulation  $\vartheta_1$  and  $\vartheta_2$  are pairs of label and suc-

- (1)  $\forall \vartheta_1 \in (s_1 \longrightarrow_1) : \exists \vartheta_2 \in (s_2 \longrightarrow_2) : \vartheta_1 R \vartheta_2$ , and  
(2)  $\forall \vartheta_2 \in (s_2 \longrightarrow_2) : \exists \vartheta_1 \in (s_1 \longrightarrow_1) : \vartheta_1 R \vartheta_2$ .

$\mathcal{T}_1$  and  $\mathcal{T}_2$  are called bisimilar if there exists a bisimulation between them. In that case, we write  $\mathcal{T}_1 \sim \mathcal{T}_2$ .

### 3.2 Abstraction settings

Abstractions basically stand for sets of implementations. Consequently, we need an embedding of the implementations into the abstraction formalism together with a refinement notion relating different levels of abstraction. Furthermore, we distinguish a subset of abstractions that are of increased practical relevance, e.g., those for which verification tools can be applied. This subset will often be the set of finite abstractions. Formally:

**Definition 5** An abstraction setting  $\mathfrak{A}$  for the implementation setting  $(I, \equiv)$  is a tuple  $(A, A', \preceq, h)$ , where

- $A$  is a set of possible abstractions,
- $A' \subseteq A$  is a set of relevant abstractions (e.g., the finite ones),
- $\preceq$  is a preorder, which we call refinement, on  $A$ , and
- $h : I \rightarrow A$  is a function which embeds  $(I, \equiv)$  in  $(A, \preceq)$ , i.e.,  $\forall i_1, i_2 \in I : i_1 \equiv i_2 \Leftrightarrow h(i_1) \preceq h(i_2)$ <sup>3</sup>. We call  $h(I) = \{h(i) \mid i \in I\}$  the set of concrete elements of  $\mathfrak{A}$ .

For the definition of the refinement preorder, two general approaches can be distinguished. The first approach uses a refinement notion based on implementations: As abstractions generally stand for sets of implementations, it is straightforward to define that one abstraction  $\alpha_1$  refines another abstraction  $\alpha_2$ , if and only if the set of implementations of  $\alpha_1$  is a subset of the set of implementations of  $\alpha_2$ . This is called *thorough refinement* [9]. Thorough refinement has the drawback of expensive refinement and satisfaction checks (the sets of implementations are usually infinite). For this reason, another approach is usually taken, defining a refinement preorder conductively that approximates the thorough refinement. Then the refinement relation has less pairs, but due to its inductive definition, efficient refinement checks are possible.

---

cessor state, say  $\vartheta_1 = (a_1, s'_1)$  and  $\vartheta_2 = (a_2, s'_2)$ . Thus  $\vartheta_1 R \vartheta_2$  is an abbreviation for  $a_1 = a_2 \wedge s'_1 R s'_2$ . This notation is used due to technical reasons, it will be useful in following definitions.

<sup>3</sup> Note that, if the condition is satisfied, we also have  $h(i_2) \preceq h(i_1)$  due to symmetry of  $\equiv$ .

### 3.3 Comparison of abstraction settings

For this subsection, let  $\mathfrak{A} = (A, A', \preceq, h)$  and  $\tilde{\mathfrak{A}} = (\tilde{A}, \tilde{A}', \tilde{\preceq}, \tilde{h})$  be two abstraction settings for the same implementation setting  $(I, \equiv)$ . We consider two approaches to compare abstraction settings. The first just compares the sets of concrete refinements (which directly correspond to the implementations) by giving transformations of the following kind:

**Definition 6 (Implementation-based embedding)** *A function  $f : A \rightarrow \tilde{A}$  is an implementation-based embedding from  $\mathfrak{A}$  to  $\tilde{\mathfrak{A}}$ , if*

- (1)  *$f$  preserves the sets of concrete refinements, i.e.,  $\forall \alpha \in A, i \in I : h(i) \preceq \alpha \Leftrightarrow \tilde{h}(i) \tilde{\preceq} f(\alpha)$ , and*
- (2)  *$f$  maps relevant abstractions to relevant abstractions, i.e.,  $f(A') \subseteq \tilde{A}'$ .*

An implementation-based embedding enables the reuse of tools and algorithms as described in the following: Suppose we have an implementation-based embedding  $f$  from  $\mathfrak{A}_1$  to  $\mathfrak{A}_2$  and we have, e.g., a sound model checking algorithm  $\text{check}_2$  for  $\mathfrak{A}_2$ , where soundness means that if the property holds for an abstraction from  $A_2$  then it holds for all its concrete refinements (if equivalence holds then the algorithm is additionally called precise). Now, suppose we cannot verify the validity of a property  $\phi$  for an element  $\alpha_1$  in  $A_1$  although  $\phi$  holds for all its concrete refinements. This can occur when (i) a sound model checking algorithm is applied that is not precise or (ii) there are only too expensive model checking algorithms available. Then it is possible that the validity of  $\phi$  can be soundly model checked for  $\alpha_1$  by checking  $f(\alpha_1)$  using algorithm  $\text{check}_2$ . The same technique can be applied for checking *thorough refinement*, as introduced after Definition 5. Consequently, there is a special interest in finding implementation-based embeddings that do not introduce too much complexity.

The above approach is especially promising if a result has already been established that the abstraction setting we map to has a “richer” refinement preorder, because this increases the probability to determine further thorough refinement pairs. How can such expressiveness results be accomplished? Checking existence of implementation-based embeddings is not suitable, since they only need to preserve the sets of implementations. Additionally, we have to take the refinement preorders into account, which is accomplished by finding or proving non-existence of so-called *preorder-based embeddings* that preserve implementations and furthermore imply an order embedding on the equivalence classes of abstractions.

**Definition 7 (Preorder-based homomorphism/embedding)** *A function  $f : A \rightarrow \tilde{A}$  is a preorder-based homomorphism from  $\mathfrak{A}$  to  $\tilde{\mathfrak{A}}$ , if*

- (1)  $f$  is monotonic, i.e.,  $\forall \alpha_1, \alpha_2 \in A : \alpha_1 \preceq \alpha_2 \Rightarrow f(\alpha_1) \tilde{\preceq} f(\alpha_2)$ ,
- (2)  $f$  keeps implementations fixed, i.e.,  $\forall i \in I : f(h(i)) \tilde{\preceq} \tilde{h}(i) \wedge \tilde{h}(i) \tilde{\preceq} f(h(i))$ ,  
and
- (3)  $f$  maps relevant abstractions to relevant abstractions, i.e.,  $f(A') \subseteq \tilde{A}'$ .

A preorder-based homomorphism  $f$  from  $\mathfrak{A}$  to  $\tilde{\mathfrak{A}}$  is a preorder-based embedding from  $\mathfrak{A}$  to  $\tilde{\mathfrak{A}}$ , if in condition (1) even equivalence is satisfied, i.e.,  $\forall \alpha_1, \alpha_2 \in A : \alpha_1 \preceq \alpha_2 \Leftrightarrow f(\alpha_1) \tilde{\preceq} f(\alpha_2)$ .

Note that a preorder-based homomorphism (resp. embedding) induces an order homomorphism (resp. order embedding) on the refinement equivalence classes, where we call two abstractions refinement equivalent if they refine each other in both directions.

**Proposition 8** (1) Every preorder-based embedding is an implementation-based embedding.

- (2) Not every preorder-based homomorphism is an implementation-based embedding.

**PROOF.** Each statement is proven separately: (1) Let  $\alpha \in A, i \in I$ . Then  $h(i) \preceq \alpha \Leftrightarrow f(h(i)) \tilde{\preceq} f(\alpha) \Leftrightarrow \tilde{h}(i) \tilde{\preceq} f(\alpha)$ . (2) A counter-example is as follows:  $I = (\{i\}, =)$ ,  $\mathfrak{A} = (\{1, 2\}, \{1, 2\}, \emptyset, \{(i, 1)\})$ ,  $\tilde{\mathfrak{A}} = (\{\tilde{1}, \tilde{2}\}, \{\tilde{1}, \tilde{2}\}, \{(\tilde{1}, \tilde{2})\}, \{(i, \tilde{1})\})$ . Then  $f = \{(1, \tilde{1}), (2, \tilde{2})\}$  is a preorder-based homomorphism, but no implementation-based embedding.  $\square$

## 4 Disjunctive modal transition systems

### 4.1 Syntax

We re-introduce disjunctive modal transition systems (DMTSs) [10], an abstraction setting for (TS,  $\sim$ ). We first describe the model: Instead of a single root state, as for TSs, DMTSs allow a set of root states. They feature two types of transitions: must and may transitions. In contrast to the transitions in TSs, a transition in a DMTS has a set of targets instead of a single target. We require this set to be non-empty, and, in the case of may transitions, the set has to have exactly one element. Every target in the target set of a must transition is required to appear also as the target of a may transition. This requirement seems reasonable, since transitions required (“must”) need to be allowed (“may”). The formal definition is as follows:

**Definition 9 (DMTS)** A disjunctive modal transition system (DMTS) is a tuple  $(U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0)$ , where  $U_D$  is a set of states,  $L$  is a set of labels,

$\mapsto_{\mathbb{D}} \subseteq U_{\mathbb{D}} \times (\mathcal{P}(L \times U_{\mathbb{D}}) \setminus \{\emptyset\})$  is the must transition relation,  $\dashrightarrow_{\mathbb{D}} \subseteq U_{\mathbb{D}} \times (\mathcal{P}(L \times U_{\mathbb{D}}) \setminus \{\emptyset\})$  is the may transition relation, and  $\emptyset \neq U_{\mathbb{D}}^0 \subseteq U_{\mathbb{D}}$  is the set of root states, such that for all  $u \in U_{\mathbb{D}}$  we have  $\forall \Theta_{\mathbb{D}} \in (u \dashrightarrow_{\mathbb{D}}) : |\Theta_{\mathbb{D}}| = 1$  and the so-called implicit-may condition

$$\forall \Theta_{\mathbb{D}} \in (u \mapsto_{\mathbb{D}}), \vartheta \in \Theta_{\mathbb{D}} : \{\vartheta\} \in (u \dashrightarrow_{\mathbb{D}}) \quad (1)$$

holds. We denote the set of all DMTSs by  $\mathbb{D}\text{MTS}$ .

A transition, i.e., an element of the must or may transition relation, is called *hypertransition*, if its target set contains more than one element. In graphical representations, hypertransitions are drawn as arrows having several heads, with every head having its own label. Must transitions are represented by solid arrows, whereas may transitions are drawn as dashed arrows. We do not draw a may transition from a state  $u$ , if there is a must (hyper-)transition starting in  $u$  that has a target set including the target of the may transition. Due to the implicit-may condition (1) of DMTSs, such a may transition always exists implicitly.

Note that DMTSs may have must hypertransitions, but are not allowed to have may hypertransitions because of the requirement  $\forall \Theta_{\mathbb{D}} \in (u \dashrightarrow_{\mathbb{D}}) : |\Theta_{\mathbb{D}}| = 1$ . The original definition of DMTS as, e.g., defined in [10], uses a common TS-style transition relation for the may transitions (without target sets). We allow target sets in the may transition relation, and require them to be singleton sets, in order to enable easier comparison to 1MTSs which allow multiple targets in the may transition relation. There, may hypertransitions imply a straightforward increase of expressibility, whereas for DMTSs the set of expressible sets of concrete refinements would remain unchanged, if may hypertransitions were allowed.

The embedding of the implementations (TSs) into the abstractions (DMTSs) is defined as follows:

**Definition 10** Define  $\pi_{\text{TS}, \text{DMTS}} : \text{TS} \rightarrow \mathbb{D}\text{MTS}$ ;  $(S, L, \longrightarrow, s^0) \mapsto (S, L, \mapsto, \dashrightarrow, \{s^0\})$ , where  $\dashrightarrow \stackrel{\text{def}}{=} \{(s, \{(a, s')\}) \mid s \xrightarrow{a} s'\}$ .

This is in fact an embedding, which will be shown in Proposition 14 after we have defined the refinement notion on DMTSs. As usual, the images of the embedding are called *concrete*, i.e., the concrete DMTSs are exactly those having one root state, no hypertransitions and only such may transitions that also appear as must transitions. The function  $\pi_{\text{TS}, \text{DMTS}}$  is obviously injective. We define  $\pi_{\text{DMTS}, \text{TS}} : \pi_{\text{TS}, \text{DMTS}}(\text{TS}) \rightarrow \text{TS}; \mathcal{U}_{\mathbb{D}} \mapsto \mathcal{T}$ , with  $\mathcal{T}$  being the unique TS with  $\pi_{\text{TS}, \text{DMTS}}(\mathcal{T}) = \mathcal{U}_{\mathbb{D}}$ .

## 4.2 Disjunctive refinement

Now we define the refinement notion for DMTSs. A disjunctive refinement is a relation relating states of one DMTS with states of another DMTS such that the root states are related and for every concrete may transition a corresponding abstract may transition can be found, as well as for every abstract must transition a corresponding concrete must transition can be found, such that the targets of the transitions match in a certain sense, requiring that successor states are again in the refinement relation. Then one DMTS disjunctively refines another DMTS, if there is a disjunctive refinement between the two. The exact definition follows:

**Definition 11 (Disjunctive refinement)** *Let  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashrightarrow_D, U_D^0)$ ,  $\hat{\mathcal{U}}_D = (\hat{U}_D, L, \hat{\mapsto}_D, \hat{\dashrightarrow}_D, \hat{U}_D^0) \in \mathbb{DMTS}$ . A disjunctive refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$  is a relation  $Q \subseteq U_D \times \hat{U}_D$  such that  $\forall u \in U_D^0 : \exists \hat{u} \in \hat{U}_D^0 : uQ\hat{u}$  and for all  $(u, \hat{u}) \in Q$  we have the following:<sup>4</sup>*

- (1)  $\forall \{\vartheta\} \in (u \dashrightarrow_D) : \exists \{\hat{\vartheta}\} \in (\hat{u} \hat{\dashrightarrow}_D) : \vartheta Q \hat{\vartheta}$ , and
- (2)  $\forall \Theta \in (\hat{u} \hat{\mapsto}_D) : \exists \Theta \in (u \mapsto_D) : \forall \vartheta \in \Theta : \exists \hat{\vartheta} \in \hat{\Theta} : \vartheta Q \hat{\vartheta}$ .

$\mathcal{U}_D$  (disjunctively) refines  $\hat{\mathcal{U}}_D$ , written  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ , if there exists a disjunctive refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .

The formal introduction of disjunctive refinement motivates, why we allow root state sets in DMTSs instead of single root states. Having more than one root state is similar to having more than one target in a hypertransition. In both cases, for each concrete root state/target an abstract root state/target needs to be found in order to have a refinement relation. Thus in some sense, a root state set corresponds to an imaginary (unlabeled) hypertransition preceding the root states, i.e., corresponds to an OR-decision.

**Proposition 12**  $\triangleleft_D$  is a preorder, i.e., the relation is reflexive and transitive.

**PROOF.** Reflexivity: Any DMTS disjunctively refines itself via equality as disjunctive refinement relation. Transitivity: Given a disjunctive refinement  $Q_{12}$  between  $\mathcal{U}_D^1$  and  $\mathcal{U}_D^2$  and a disjunctive refinement  $Q_{23}$  between  $\mathcal{U}_D^2$  and  $\mathcal{U}_D^3$ , it is straightforwardly checked that  $Q_{12} \circ Q_{23}$  is a disjunctive refinement between  $\mathcal{U}_D^1$  and  $\mathcal{U}_D^3$ .  $\square$

<sup>4</sup> Note that in the definition of disjunctive refinement  $\vartheta$  and  $\hat{\vartheta}$  are pairs of label and successor state, say  $\vartheta = (a, u')$  and  $\hat{\vartheta} = (\hat{a}, \hat{u}')$ . Thus  $\vartheta R \hat{\vartheta}$  is an abbreviation for  $a = \hat{a} \wedge u' R \hat{u}'$ .

The preorder induces an equivalence relation on  $\mathbb{DMTS}$  as follows:

**Definition 13 (DR-equivalence)** Let  $\mathcal{U}_D^1, \mathcal{U}_D^2 \in \mathbb{DMTS}$ . We call  $\mathcal{U}_D^1$  and  $\mathcal{U}_D^2$  disjunctive refinement equivalent (or shortly DR-equivalent), if  $\mathcal{U}_D^1 \triangleleft_D \mathcal{U}_D^2$  and  $\mathcal{U}_D^2 \triangleleft_D \mathcal{U}_D^1$ . In that case, we write  $\mathcal{U}_D^1 \approx_D \mathcal{U}_D^2$ .

In order to get a proper abstraction setting, we need to make sure that  $\pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}}$  is an embedding, i.e., in the case of two concrete  $\mathbb{DMTS}$ s, disjunctive refinement should coincide with bisimulation:

**Proposition 14** For all  $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{T}\mathbb{S}$ , we have  $\mathcal{T}_1 \sim \mathcal{T}_2 \Leftrightarrow \pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}}(\mathcal{T}_1) \triangleleft_D \pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}}(\mathcal{T}_2)$ .

**PROOF.** Consider the definition of disjunctive refinement in the special case that both  $\mathbb{DMTS}$ s are concrete. Then the proposition is obvious.  $\square$

We now have established everything needed to fix the abstraction setting of  $\mathbb{DMTS}$ :

**Definition 15** The abstraction setting  $\mathbb{DMTS}$  is defined to be  $(\mathbb{DMTS}, \mathbb{DMTS}^{\text{fin}}, \triangleleft_D, \pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}})$ , where  $\mathbb{DMTS}^{\text{fin}} = \{(U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0) \mid |U_D| + |\dashv\rightarrow_D| < \infty\}$ .

Note that due to the implicit-may condition of  $\mathbb{DMTS}$ s, finiteness of  $\dashv\rightarrow_D$  implies finiteness of  $\mapsto_D$  as well as a finite branching property of hypertransitions, i.e., every hypertransition has only finitely many targets.

The following proposition states that refinement does not coincide with inclusion of sets of concrete refinements. As a consequence, the two comparison approaches presented in Section 3.3 do not coincide for  $\mathbb{DMTS}$ s.

**Proposition 16** (1) Refinement implies inclusion of the sets of concrete refinements, i.e., for all  $\mathcal{U}_D, \hat{\mathcal{U}}_D \in \mathbb{DMTS}$  we have  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D \Rightarrow \{\tilde{\mathcal{U}}_D \in \pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}}(\mathbb{T}\mathbb{S}) \mid \tilde{\mathcal{U}}_D \triangleleft_D \mathcal{U}_D\} \subseteq \{\tilde{\mathcal{U}}_D \in \pi_{\mathbb{T}\mathbb{S}, \mathbb{DMTS}}(\mathbb{T}\mathbb{S}) \mid \tilde{\mathcal{U}}_D \triangleleft_D \hat{\mathcal{U}}_D\}$ .  
(2) The other implication (“ $\Leftarrow$ ”) does not hold.

**PROOF.** (1) follows directly from the transitivity of  $\mathcal{U}_D$ . (2) is proven by the counter-example shown in Figure 2.  $\square$

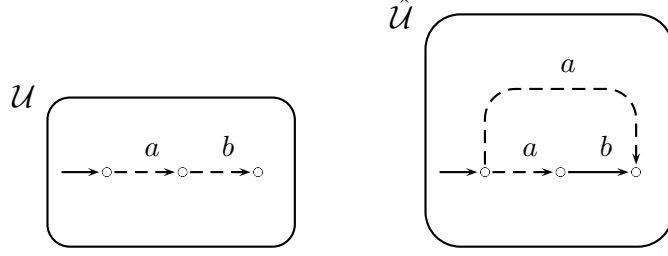


Fig. 2.  $\mathcal{U}$  does not refine  $\hat{\mathcal{U}}$ , although both have the same concrete refinements.

### 4.3 All refinements of a simple DMTS

We consider the simple DMTS  $\hat{\mathcal{U}}_{\mathbb{D}} \stackrel{\text{def}}{=} (\{0, 1\}, \{a, b\}, \{(0, \{(a, 1), (b, 1)\})\}, \{(0, \{(a, 1)\}), (0, \{(b, 1)\})\}, \{0\})$ , which is illustrated at the top of Figure 3. It consists of a single (must) hypertransition. By the implicit-may condition of DMTSs, it has two “implicit” may transitions, which we do not draw in illustrations. We want to give a complete overview over all refinements of  $\hat{\mathcal{U}}_{\mathbb{D}}$ , up to refinement equivalence. We show that Figure 3 gives such a complete overview. Here, DMTSs below disjunctively refine DMTSs above. Consequently, Figure 3 forms a Hasse diagram of all equivalence classes of disjunctive refinements of  $\hat{\mathcal{U}}_{\mathbb{D}}$ , ordered by  $\triangleleft_{\mathbb{D}}$ .

**Theorem 17** (1) *In Figure 3, every line represents disjunctive refinement, i.e., if there is a line from  $\mathcal{U}_{\mathbb{D}}$  (below) to  $\tilde{\mathcal{U}}_{\mathbb{D}}$  (above), then  $\mathcal{U}_{\mathbb{D}} \triangleleft_{\mathbb{D}} \tilde{\mathcal{U}}_{\mathbb{D}}$ . Furthermore, if there is no ascending path of lines from  $\mathcal{U}_{\mathbb{D}}$  (below) to  $\tilde{\mathcal{U}}_{\mathbb{D}}$  (above), then  $\mathcal{U}_{\mathbb{D}} \not\triangleleft_{\mathbb{D}} \tilde{\mathcal{U}}_{\mathbb{D}}$ .*  
(2) *Each refinement of  $\hat{\mathcal{U}}_{\mathbb{D}}$ , shown at the top of Figure 3, is DR-equivalent to one of the DMTSs shown in Figure 3.*

**PROOF.** (1) is straightforwardly checked. The proof of (2) is given in Appendix A.1.  $\square$

## 5 1-selecting modal transition systems

### 5.1 Syntax

Having defined DMTSs together with their refinement notion, we now continue with the definition of the newly introduced 1MTSs. The approach is similar, since the two notions essentially only differ in the interpretation of hypertransitions. As a result of this difference, may hypertransitions, that were not featured in DMTSs, make sense here and for this reason are allowed. As a consequence, the requirement that must transitions have to appear as may

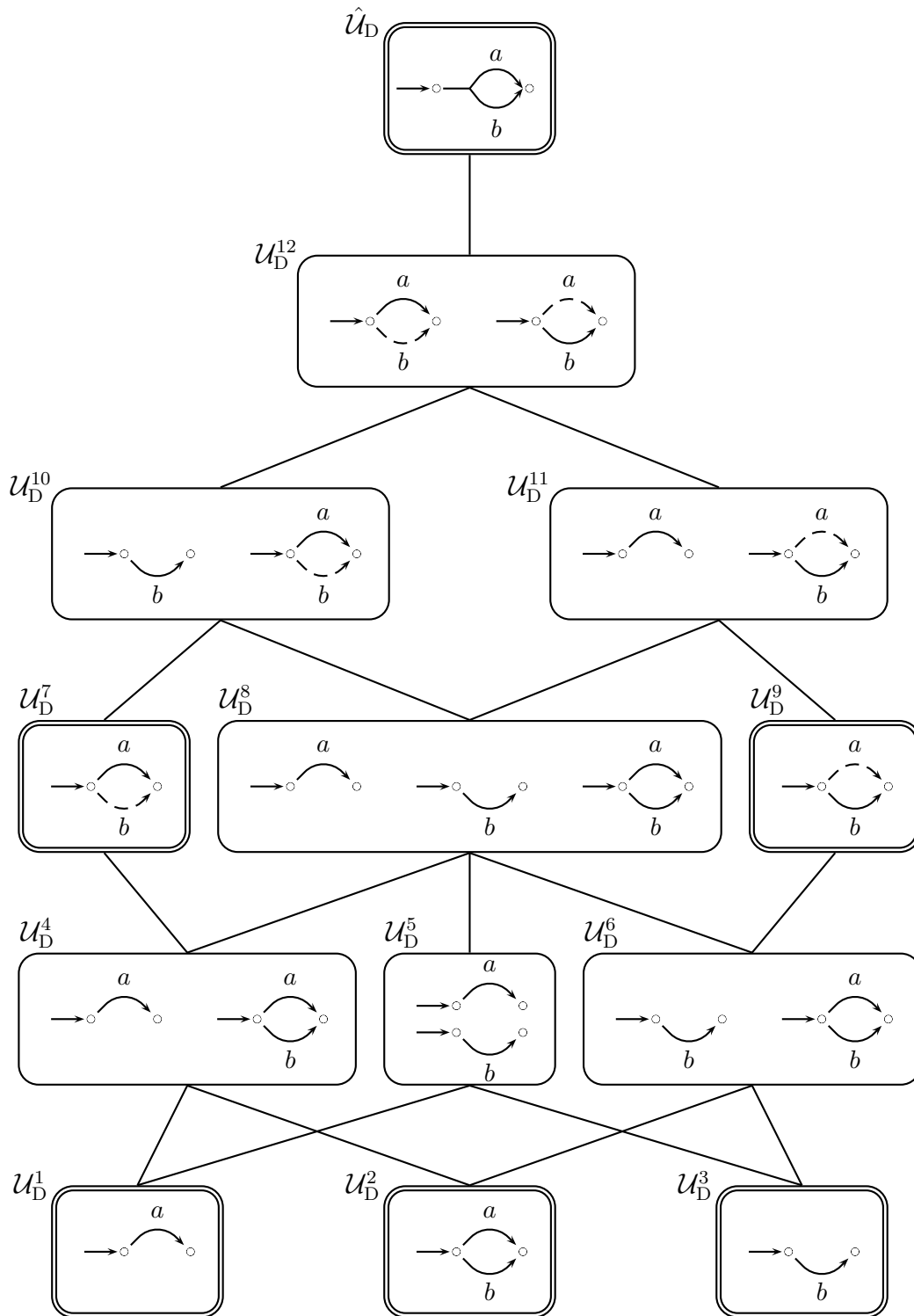


Fig. 3.  $[\hat{\mathcal{U}}_D]_{\approx_D}$  and all its disjunctive refinements in a Hasse diagram.

transitions (implicit-may condition in the case of DMTSs) then simplifies to  $\mapsto_1 \subseteq \dashv\rightarrow_1$  for 1MTSs. The formal definition follows:

**Definition 18 (1MTS)** *A 1-selecting modal transition system (1MTS) is a tuple  $(U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0)$ , where  $U_1$  is a set of states,  $L$  is a set of labels,  $\mapsto_1 \subseteq U_1 \times (\mathcal{P}(L \times U_1) \setminus \emptyset)$  is the must transition relation,  $\dashv\rightarrow_1 \subseteq U_1 \times (\mathcal{P}(L \times U_1) \setminus \emptyset)$  is the may transition relation, and  $\emptyset \neq U_1^0 \subseteq U_1$  is the set of root states, such that  $\mapsto_1 \subseteq \dashv\rightarrow_1$  holds. We denote the set of all 1MTSs by  $\mathbb{1MTS}$ .*

The embedding of the implementations (TSs) into the abstractions (1MTSs) is defined as follows:

**Definition 19** *Define  $\pi_{\text{TS}, \mathbb{1MTS}} : \mathbb{TS} \rightarrow \mathbb{1MTS}$ ;  $(S, L, \longrightarrow, s^0) \mapsto (S, L, \mapsto, \mapsto, \{s^0\})$ , where  $\mapsto \stackrel{\text{def}}{=} \{(s, \{(a, s')\}) \mid s \xrightarrow{a} s'\}$ .*

This is in fact an embedding, which will be shown in Proposition 23 after we have defined the refinement notion on 1MTSs. As usual, the images of the embedding are called *concrete*, i.e., the concrete 1MTSs are exactly those having one root state, no hypertransitions and only such may transitions that also appear as must transitions. The function  $\pi_{\text{TS}, \mathbb{1MTS}}$  is obviously injective. We define  $\pi_{\mathbb{1MTS}, \text{TS}} : \pi_{\text{TS}, \mathbb{1MTS}}(\mathbb{TS}) \rightarrow \mathbb{TS}; \mathcal{U}_1 \mapsto \mathcal{T}$ , with  $\mathcal{T}$  being the unique TS with  $\pi_{\text{TS}, \mathbb{1MTS}}(\mathcal{T}) = \mathcal{U}_1$ .

## 5.2 1-selecting refinement

Now we define the refinement notion for the newly introduced 1MTSs. A 1-selecting refinement is a relation relating states of one 1MTS with states of another 1MTS in a way similar to DMTSs. However, here we require for every choice function that chooses targets in the concrete system a choice function that chooses targets in the abstract system, such that the chosen targets match. One 1MTS 1-selecting refines another 1MTS, if there is an 1-selecting refinement between them. The exact definition follows:

**Definition 20 (1-selecting refinement)** *Let  $\mathcal{U}_1 = (U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0)$ ,  $\hat{\mathcal{U}}_1 = (\hat{U}_1, L, \hat{\mapsto}_1, \hat{\dashv\rightarrow}_1, \hat{U}_1^0) \in \mathbb{1MTS}$ . An 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$  is a relation  $Q \subseteq U_1 \times \hat{U}_1$  such that  $\forall u \in U_1^0 : \exists \hat{u} \in \hat{U}_1^0 : uQ\hat{u}$  and for all  $(u, \hat{u}) \in Q$  and all  $\gamma \in \text{choice}(u \dashv\rightarrow_1)$  there exists  $\hat{\gamma} \in \text{choice}(\hat{u} \hat{\dashv\rightarrow}_1)$  such that the following holds:<sup>5</sup>*

$$(1) \quad \forall \Theta \in (u \dashv\rightarrow_1) : \exists \hat{\Theta} \in (\hat{u} \hat{\dashv\rightarrow}_1) : \gamma(\Theta) Q \hat{\gamma}(\hat{\Theta}), \text{ and}$$

<sup>5</sup> Note that in the definition of 1-selecting refinement  $\gamma(\Theta)$  and  $\hat{\gamma}(\hat{\Theta})$  are pairs of label and successor state, say  $\gamma(\Theta) = (a, u')$  and  $\hat{\gamma}(\hat{\Theta}) = (\hat{a}, \hat{u}')$ . Thus  $\gamma(\Theta)R\hat{\gamma}(\hat{\Theta})$  is an abbreviation for  $a = \hat{a} \wedge u'R\hat{u}'$ .

(2)  $\forall \hat{\Theta} \in (\hat{u} \hat{\mapsto}_1) : \exists \Theta \in (u \mapsto_1) : \gamma(\Theta) Q \hat{\gamma}(\hat{\Theta})$ .

$\mathcal{U}_1$  (1-selecting) refines  $\hat{\mathcal{U}}_1$ , written  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1$ , if there exists a 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ .

**Proposition 21**  $\triangleleft_1$  is a preorder, i.e., the relation is reflexive and transitive.

**PROOF.** By the same arguments as for disjunctive refinement (Proposition 12).  $\square$

1-selecting refinement induces an equivalence notion on  $\mathbb{1MTS}$  as follows:

**Definition 22 (1R-equivalence)** Let  $\mathcal{U}_1^1, \mathcal{U}_1^2 \in \mathbb{1MTS}$ . We call  $\mathcal{U}_1^1$  and  $\mathcal{U}_1^2$  1-selecting refinement equivalent (or shortly 1R-equivalent), if  $\mathcal{U}_1^1 \triangleleft_1 \mathcal{U}_1^2$  and  $\mathcal{U}_1^2 \triangleleft_1 \mathcal{U}_1^1$ . In that case, we write  $\mathcal{U}_1^1 \approx_1 \mathcal{U}_1^2$ .

In order to get a proper abstraction setting, we need to make sure that  $\pi_{\mathbb{T}\$, \mathbb{1MTS}}$  is an embedding, i.e., in the case of two concrete  $\mathbb{1MTS}$ s, 1-selecting refinement should coincide with bisimulation:

**Proposition 23** For all  $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{T}\$, we have  $\mathcal{T}_1 \sim \mathcal{T}_2 \Leftrightarrow \pi_{\mathbb{T}\$, \mathbb{1MTS}}(\mathcal{T}_1) \triangleleft_1 \pi_{\mathbb{T}\$, \mathbb{1MTS}}(\mathcal{T}_2)$ .$

**PROOF.** By the same arguments as for disjunctive refinement (Proposition 14).  $\square$

We now have established everything needed to fix the abstraction setting of  $\mathbb{1MTS}$ :

**Definition 24** The abstraction setting  $\mathbb{1MTS}$  is defined to be  $(\mathbb{1MTS}, \mathbb{1MTS}^{\text{fin}}, \triangleleft_1, \pi_{\mathbb{T}\$, \mathbb{1MTS}})$ , where  $\mathbb{1MTS}^{\text{fin}} = \{(U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0) \mid |U_1| + |\dashv\rightarrow_1| < \infty \wedge \forall u \in U_1, \Theta_1 \in (u \dashv\rightarrow_1) : |\Theta_1| < \infty\}$ .

Note that due to the required condition  $\mapsto_1 \subseteq \dashv\rightarrow_1$  of any  $\mathbb{1MTS}$ , finiteness of  $\dashv\rightarrow_D$  implies finiteness of  $\mapsto_D$ .

The following proposition states that refinement does not coincide with inclusion of sets of concrete refinements. As a consequence, the two comparison approaches presented in Section 3.3 do not coincide for  $\mathbb{1MTS}$ s.

**Proposition 25** (1) Refinement implies inclusion of the sets of concrete refinements, i.e., for all  $\mathcal{U}_1, \hat{\mathcal{U}}_1 \in \mathbb{1MTS}$  we have  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1 \Rightarrow \{\tilde{\mathcal{U}}_1 \in \pi_{\mathbb{T}\$, \mathbb{1MTS}}(\mathbb{T}\$) \mid \tilde{\mathcal{U}}_1 \triangleleft_1 \mathcal{U}_1\} \subseteq \{\tilde{\mathcal{U}}_1 \in \pi_{\mathbb{T}\$, \mathbb{1MTS}}(\mathbb{T}\$) \mid \tilde{\mathcal{U}}_1 \triangleleft_1 \hat{\mathcal{U}}_1\}$ .

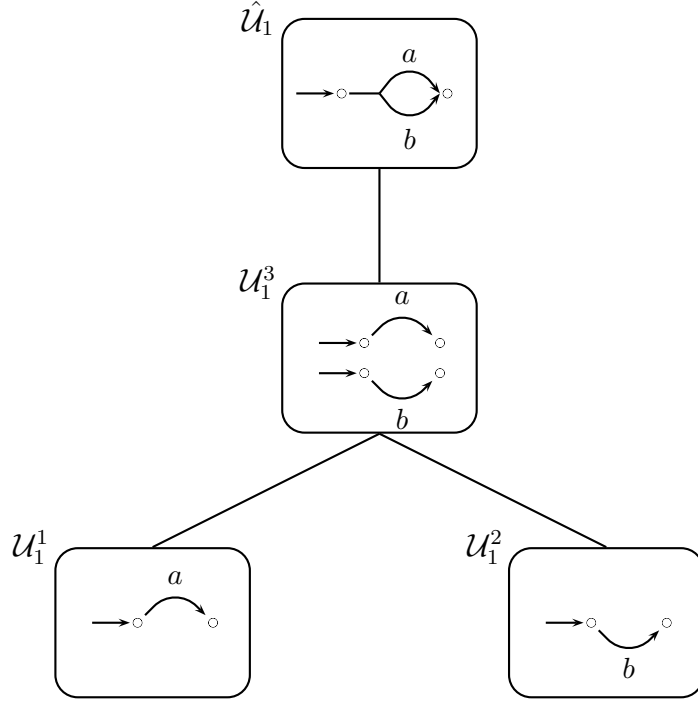


Fig. 4.  $[\hat{\mathcal{U}}_1]_{\approx_1}$  and all its 1-selecting refinements in a Hasse diagram.

(2) The other implication (“ $\Leftarrow$ ”) does not hold.

**PROOF.** By the same arguments as for disjunctive refinement (Proposition 16). Even for (2) the same counter-example (Figure 2), interpreted here as 1MTS, works.  $\square$

### 5.3 All refinements of a simple 1MTS

In this subsection, we give a complete overview over all 1-selecting refinements of a simple 1MTS. The 1MTS of interest is  $\hat{\mathcal{U}}_1 \stackrel{\text{def}}{=} (\{0, 1\}, \{a, b\}, \{(0, \{(a, 1), (b, 1)\}), \{(0, \{(a, 1), (b, 1)\}), \{0\})\}$ . This 1MTS is illustrated at the top of Figure 4. As usual, we do not draw may (hyper-)transitions, if they also exist as must (hyper-)transitions. For this reason, the may hypertransition of  $\hat{\mathcal{U}}_1$  does not appear in the drawing. We show that Figure 4 gives a complete overview over all refinements of  $[\hat{\mathcal{U}}_1]_{\approx_1}$ , up to refinement equivalence. Here, 1MTSs below 1-selecting refine 1MTSs above. Consequently, Figure 4 forms a Hasse diagram of all equivalence classes of 1-selecting refinements of  $\hat{\mathcal{U}}_1$ , ordered by  $\triangleleft_1$ .

**Theorem 26** (1) In Figure 4, every line represents 1-selecting refinement, i.e., if there is a line from  $\mathcal{U}_1$  (below) to  $\tilde{\mathcal{U}}_1$  (above), then  $\mathcal{U}_1 \triangleleft_1 \tilde{\mathcal{U}}_1$ . Furthermore, if there is no ascending path of lines from  $\mathcal{U}_1$  (below) to  $\tilde{\mathcal{U}}_1$  (above), then  $\mathcal{U}_1 \not\triangleleft_1 \tilde{\mathcal{U}}_1$ .

(2) Each refinement of  $\hat{\mathcal{U}}_1$ , shown at the top of Figure 4, is DR-equivalent to one of the 1MTSs shown in Figure 4.

**PROOF.** (1) is straightforwardly checked. The proof of (2) is given in Appendix A.2.  $\square$

## 6 Comparison

### 6.1 A preorder-based embedding from DMTS to 1MTS

We define a function  $f$  from  $\mathbb{D}\text{MTS}$  to  $\mathbb{1}\text{MTS}$  and show that it is a preorder-based embedding. Before giving the formal definition of function  $f$ , we describe informally the idea behind it: In a DMTS, arbitrarily many targets of a hypertransition can be “taken”, whereas in 1MTSs, only one target per hypertransition can be “taken”. The idea is to turn every DMTS-hypertransition with  $n$  targets into  $n$  1MTS-hypertransitions. To achieve this, we need to introduce “state copies” (every DMTS-state becomes two 1MTS-states), because no multisets are used<sup>6</sup>. Using these copies, we can have transitions, that are “behaviourally” the same, but in fact lead to different states (different copies of the same DMTS-state). The formal definition of  $f$  is as follows:

**Definition 27** Define  $f : \mathbb{D}\text{MTS} \rightarrow \mathbb{1}\text{MTS}$ ;  $(U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0) \mapsto (U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0)$ , with

$$\begin{aligned}
U_1 &\stackrel{\text{def}}{=} U_D \times \{0, 1\}, \text{ where we usually write } u_i \text{ instead of } (u, i), \\
u_i \mapsto_1 \Theta_1 &\stackrel{\text{def}}{\iff} \exists \Theta_D \in (u \mapsto_D) : \exists (\tilde{a}, \tilde{u}') \in \Theta_D : \\
&\quad \Theta_1 = \{(a, u'_0) \mid (a, u') \in \Theta_D\} \cup \{(\tilde{a}, \tilde{u}'_1)\}, \\
u_i \dashv\rightarrow_1 \Theta_1 &\stackrel{\text{def}}{\iff} (\exists a \in L, u' \in U_D : \{(a, u')\} \in (u \dashv\rightarrow_D) \wedge \\
&\quad \Theta_1 = \{(a, u'_0), (a, u'_1)\}) \vee (u_i \mapsto_1 \Theta_1), \\
U_1^0 &\stackrel{\text{def}}{=} U_D^0 \times \{0, 1\}.
\end{aligned}$$

The result of transformation  $f$  applied to the DMTS shown in Figure 1(c) is illustrated in Figure 5(a).

**Theorem 28** Let  $f$  be the function defined in Definition 27.

<sup>6</sup> It is possible to use multisets, in which case “state copies” are easier to handle. However, DMTSs have no multisets and we decide not to change the syntax too much.

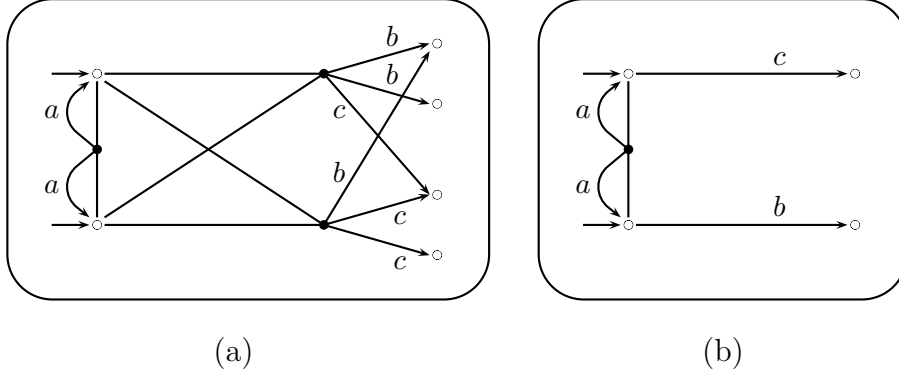


Fig. 5. Left: The result of transformation  $f$  applied to the DMTS shown in Figure 1(c). For better readability, hypertransitions with the same target sets are grouped using a solid circle. Strictly speaking, there are some more may transitions than the ones implied by corresponding must transitions (which are never drawn). However, the 1MTS shown here is 1R-equivalent to the result of  $f$ . Right: The result of transformation  $g$  applied to the 1MTS shown in Figure 1(c).

- (1)  $f$  is a preorder-based embedding.
- (2)  $f$  is an implementation-based embedding.

**PROOF.** The proof of statement (1) is given in Appendix A.3. Statement (2) then follows immediately by Proposition 8(1).  $\square$

## 6.2 A preorder-based homomorphism from 1MTS to DMTS

We define a function  $g$  from 1MTS to DMTS and show that it is a preorder-based homomorphism. Thereafter we prove that  $g$  is not a preorder-based embedding. Nevertheless,  $g$  is an implementation-based embedding, as will be shown at the end of this subsection.

The idea behind function  $g$  is straightforward: Every state of the 1MTS, say  $u$ , is turned into several states in the DMTS, one for each possible choice function  $\gamma \in \text{choice}(u \dashv\rightarrow_1)$ . Thus the states of the DMTS are pairs of a 1MTS-state and a choice function. Then  $(u, \gamma)$  plays the part of  $u$  in the case that choice function  $\gamma$  would be considered in the 1MTS. Formally:

**Definition 29** Define  $g : \mathbf{1MTS} \rightarrow \mathbf{DMTS}$ ;  $(U_1, L, \dashv\rightarrow_1, \dashv\rightarrow_1, U_1^0) \mapsto (U_D, L, \dashv\rightarrow_D, \dashv\rightarrow_D, U_D^0)$ , with

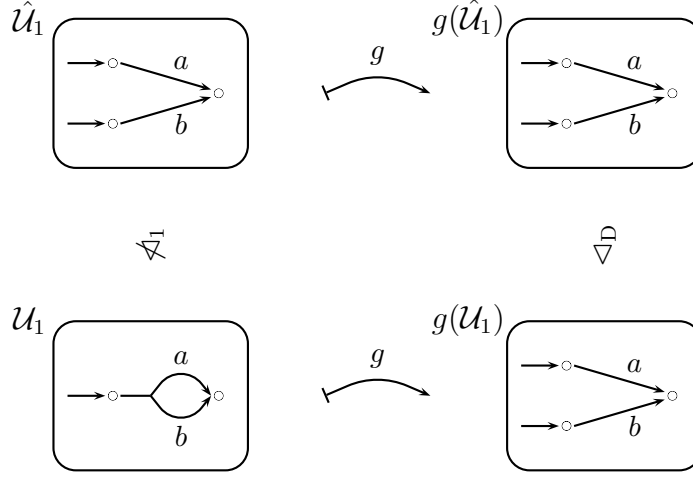


Fig. 6. A counter-example to show that  $g$  is not a preorder-based embedding.

$$\begin{aligned}
U_D &\stackrel{\text{def}}{=} \{(u, \gamma) \mid u \in U_1 \wedge \gamma \in \text{choice}(u \dashrightarrow_1)\}, \\
(u, \gamma) \dashrightarrow_D \Theta &\stackrel{\text{def}}{\Leftrightarrow} \exists(a, u') \in \gamma(u \dashrightarrow_1) : \\
&\quad \Theta = \{(a, (u', \gamma')) \mid \gamma' \in \text{choice}(u' \dashrightarrow_1)\}, \\
(u, \gamma) \dashrightarrow_D \{\vartheta\} &\stackrel{\text{def}}{\Leftrightarrow} \exists(a, u') \in \gamma(u \dashrightarrow_1) : \\
&\quad \vartheta \in \{(a, (u', \gamma')) \mid \gamma' \in \text{choice}(u' \dashrightarrow_1)\}, \\
U_D^0 &\stackrel{\text{def}}{=} \{(u, \gamma) \mid u \in U_1^0 \wedge \gamma \in \text{choice}(u \dashrightarrow_1)\}.
\end{aligned}$$

The result of transformation  $g$  applied to the 1MTS shown in Figure 1(c) is illustrated in Figure 5(b).

**Theorem 30** *Let  $g$  be the function defined in Definition 29.*

- (1)  $g$  is a preorder-based homomorphism.
- (2)  $g$  is not a preorder-based embedding.
- (3)  $g$  is an implementation-based embedding.

**PROOF.** Statements (1) and (3) are proven in Appendices A.4 and A.5. For the proof of statement (2), it is enough to find  $\mathcal{U}_1, \hat{\mathcal{U}}_1 \in \mathbf{1MTS}$  such that  $g(\mathcal{U}_1) \triangleleft_D g(\hat{\mathcal{U}}_1)$ , but  $\mathcal{U}_1 \not\triangleleft_1 \hat{\mathcal{U}}_1$ . Such a counter-example is illustrated in Figure 6.  $\square$

### 6.3 Non-existence of a preorder-based embedding from 1MTS to DMTS

In the previous subsection, a straightforward preorder-based homomorphism from 1MTS to DMTS was presented, which is not a preorder-based embedding. Now, we will see that there exists no preorder-based embedding at all.

Thus, with respect to the preorder-based approach, the two formalisms are not equally expressive; 1MTSs have strictly more expressive power than DMTSs.

We consider the 1MTS  $\hat{\mathcal{U}}_1$  that was examined in Section 5.3, and prove that its Hasse structure (Figure 4) cannot be embedded in the DMTS-formalism such that concrete systems are preserved. As a consequence there is no preorder-based embedding from 1MTS to DMTS.

**Theorem 31** *There is no preorder-based embedding from 1MTS to DMTS.*

**PROOF.** The proof is given in Appendix A.6.  $\square$

## 7 Related work

The approach of extending common transition systems by a second transition relation expressing which steps *may* appear in an implementation was followed by Larsen and Thomsen, who introduced *modal transition systems* [6], and by Dams, who called his extension *mixed transition systems* [7,8]. Modal transition systems require that every must transition has to be contained in the may transition relation, whereas mixed transition systems do not have this restriction and therefore have non-implementable abstractions.

Larsen and Xinxin were the first to extend the must and may transition approach by hypertransitions, which resulted in their definition of *disjunctive modal transition systems* [10]. Larsen and Xinxin also showed how a DMTS can be used to express the solution set of an equation system formulated in process algebra.

Generalized Kripke modal transition systems, which are DMTSs having predicates on states, but no transition labels are considered in [11] in the context of abstraction. In [12] it is shown that transition labels of modal transition systems can be encoded via predicates on states and vice versa.

Alfaro et al. used in [13] a DMTS-like approach for the underspecification of turn-based games, extending these structures by must and may transitions and hypertransitions. This resulted in their definition of *abstract game structures*.

In [14], Dams and Namjoshi have presented yet another transition system variant called *focused transition systems (FTS)*. The corresponding abstraction framework is *complete* in the sense that for every system one can find a finite abstraction such that a given correctness property can be shown. The authors extend mixed transition systems by fairness constraints and two types

of hypertransitions, so called *focus* and *de-focus* steps that model disjunction, respectively conjunction in some sense.

$\mu$ -automata [15], which can be considered to have OR-states introducing disjunction similar to hypertransitions, are shown to yield complete models for abstraction with respect to the modal  $\mu$ -calculus in [16]. There, the authors also define *modal automata*, extending  $\mu$ -automata by may transitions. Thus modal automata have implicit may hypertransitions interpreted as OR. Other abstraction settings that have may hypertransitions are hyper Kripke modal transition systems [17] and hypermixed Kripke structures [18]. In [18], the may hypertransitions, together with fairness constraints, are used to yield a suitable abstract model for predicate abstraction [20] extended with ranking functions such that completeness of these predicate abstraction is obtained. There, may hypertransitions correspond to a conjunctive interpretation (AND). In [17], the may hypertransition targets are interpreted as OR, but contrary to the other approaches, different may transitions with the same source state are interpreted conjunctively (AND). These may hypertransitions are used to increase the precision of abstraction for non-functional abstraction, i.e., such abstraction that a concrete state can be abstracted to different elements. If functional abstractions are considered, i.e., those yielding a state space partition, generalized Kripke modal transition systems and  $\mu$ -automata turn out to yield suitable abstract models for precise abstractions with respect to two abstract-model-independent characterizations of precision [19]. To sum up, there are four different interpretations of may hypertransitions in the four models: Our 1MTSs, Kripke modal transition systems [17], hypermixed Kripke structures [18], and modal automata [16].

The expressiveness of abstraction settings has mainly been examined by comparing refinement preorders with respect to inclusion (possibly after ‘canonical’ transformations of the underlying models), i.e., a refinement relation  $\leq_1$  is less expressive than  $\leq_2$  if every two related elements via  $\leq_1$  are also related via  $\leq_2$ . This is, e.g., done in [21] for some simulation- and trace inclusion-based refinement notions. In [22] testing preorder and in [23] ready simulation (named  $\frac{2}{3}$ -bisimulation) is transformed to prebisimulation. In [24] forward/backward simulation as well as trace inclusion are transformed to disjunctive modal transition systems (named underspecified transition systems). These transformations implicitly show that the transformed settings are less or equally expressive with respect to the describable sets of transition systems.

## 8 Conclusion

We compared the abstraction setting of disjunctive modal transition systems (DMTS) with the newly developed 1-selection modal transition systems

(1MTS). The key difference between the two lies in the interpretation of hyper-transitions: DMTSs require *at least* one alternative to be taken in a concrete refinement, whereas 1MTSs require *exactly* one alternative to be taken. This way, the new approach followed by 1MTS gives a modeler the possibility to express an exclusive choice in a direct way (an implementation is supposed to have exactly one of several alternatives). The refinement notions of the two abstraction settings were illustrated by two diagrams showing all refinements of a simple DMTS example and a simple 1MTS example (up to refinement equivalence).

With the aim of comparing 1MTSs with DMTSs, general abstraction settings and certain types of transformations between them were introduced. The first type of transformation, implementation-based embedding, preserves the sets of concrete refinements and does not take the full refinement preorder into account. The second type of transformation, preorder-based embedding, implies an order embedding on the equivalence classes of abstractions, ordered by the refinement relation. The two approaches differ, if the refinement preorder is approximative, which is usually the case. Approximative refinements are used, because they are defined coinductively and thus refinement can be checked efficiently without considering each implementation. Implementation-based embeddings do not care about the approximative refinement preorder and to this respect have maximum precision. Tools and algorithms for one abstraction setting can be reused in another setting by first applying the transformation. If there are implementation-based embeddings in both directions, like for the settings of DMTSs and 1MTSs, as has been shown in this paper, this implies equal expressive power with respect to the describable sets of implementations. If two abstraction settings shall be compared also with respect to their degree of approximativity, preorder-based embeddings have to be found. Regarding the two considered abstraction settings, we found a preorder-based embedding from DMTS to 1MTS and proved that there is none from 1MTS to DMTS, which indicates that 1MTS have more expressive power (i.e., less approximative refinement). Consequently, it can be promising to use a preferably low-complexity implementation-based embedding to the more expressive abstraction setting and take advantage of the less approximative refinement there, e.g., check there for refinement.

### *Future work*

It is future work to define a three-valued satisfaction notion on 1MTSs. Generally, the question, whether a formula is satisfied by an abstraction, can have three answers: *yes*, *no*, or *undefined*. In thorough semantics, the value *undefined* is only obtained if there exists both a refinement that satisfies the property and one that falsifies the property. Since for thorough semantics there are only expensive model checking algorithms, we are looking for an approximative sat-

isfaction definition for 1MTSs, especially a compositional one.

Abstraction settings should be *sound*, i.e., if a property is satisfied (falsified) in an abstraction, every refinement should also satisfy (resp. falsify) the property. In *complete* abstraction settings, it is also possible to find a finite (introduced here as “relevant”) abstraction that satisfies a formula for each possibly infinite implementation and given formula satisfied by the implementation. It is future work to extend 1MTSs by fairness constraints similar as in [14] and try to establish a complete abstraction setting.

A further possible modification of 1MTSs is to combine DMTS- and 1MTS-hypertransitions in one setting. Moreover, a topic of future work is to develop and examine an abstraction setting, where hypertransitions can be decorated with a number defining exactly how many of the targets should appear in an implementation. The various modifications can be compared with respect to expressiveness (i.e., put into a hierarchy) using the approaches introduced in this paper.

It is not yet clear whether 1MTS really allow more compact/less approximative representations of sets of implementations compared to DMTS. The higher expressive power of 1MTS suggests that, but further examinations are still to be done. Furthermore, for reuse of tools and algorithms, there is an interest to find implementation-based embeddings with minimum complexity. Maybe cheaper transformations can be found than the ones presented here.

## A Proofs

### A.1 Proof of Theorem 17(2)

We start with a couple of definitions and lemmas. Given a DMTS  $\mathcal{U}_D$ , we say that a state  $u$  is  $\mathcal{U}_D$ -*reachable* from  $\tilde{u}$ , if there is a chain of may transitions from  $\tilde{u}$  to  $u$ . Then every state reachable via a chain of must transitions is also reachable in the above sense (via may transitions) due to the implicit-may condition of DMTSs. A state  $u$  is  $\mathcal{U}_D$ -*reachable*, if it is reachable from a root state of  $\mathcal{U}_D$ . A DMTS  $\mathcal{C}$  is a *component*, if it has exactly one root state and all its states are  $\mathcal{C}$ -reachable. A DMTS  $\mathcal{C} = (U_C, L, \mapsto_C, \dashv\rightarrow_C, U_C^0)$  is a *component* of another DMTS  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0)$ , if there is  $u^0 \in U_D^0$  such that  $U_C^0 = \{u^0\}$ ,  $U_C = \{u \in U_D \mid u \text{ is } \mathcal{U}_D\text{-reachable from } u^0\}$ , and  $\mapsto_C$  ( $\dashv\rightarrow_C$ ) is the projection of  $\mapsto_D$  (resp.  $\dashv\rightarrow_D$ ) to  $\mathcal{C}$ , i.e.,  $u \mapsto_C \Theta$  if and only if  $u \in U_C \wedge u \mapsto_D \Theta$ , and  $u \dashv\rightarrow_C \{(a, u')\}$  if and only if  $u \in U_C \wedge u \dashv\rightarrow_D \{(a, u')\}$ . We denote the set of all components of  $\mathcal{U}_D$  by  $\text{Comp}(\mathcal{U}_D)$ . For any DMTS  $\mathcal{U}_D$ , each component of  $\mathcal{U}_D$  is obviously a component. The following lemmata

describe some properties of components and their connection to disjunctive refinement:

**Lemma 32** *Let  $\mathcal{U}_D, \hat{\mathcal{U}}_D \in \mathbb{DMTS}$ . Then  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$  if and only if  $\forall \mathcal{C} \in \text{Comp}(\mathcal{U}_D) : \mathcal{C} \triangleleft_D \hat{\mathcal{U}}_D$ .*

**PROOF.** Obvious by definition of disjunctive refinement.  $\square$

**Lemma 33** *Let  $\mathcal{U}_D \in \mathbb{DMTS}$ . Then  $\forall \mathcal{C} \in \text{Comp}(\mathcal{U}_D) : \mathcal{C} \triangleleft_D \mathcal{U}_D$ .*

**PROOF.** Apply Lemma 32 to  $\mathcal{U}_D$  and  $\mathcal{U}_D$ .  $\square$

**Lemma 34** *Let  $\mathcal{U}_D, \hat{\mathcal{U}}_D \in \mathbb{DMTS}$  such that  $\exists \mathcal{C} \in \text{Comp}(\hat{\mathcal{U}}_D) : \mathcal{U}_D \triangleleft_D \mathcal{C}$ . Then  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ .*

**PROOF.** Obvious by definition of disjunctive refinement.  $\square$

A merge operator  $\otimes$  for components is defined as follows:

**Definition 35** *Let  $I$  be an index set and  $\mathcal{U}_D^i = (U_D^i, L^i, \mapsto_D^i, \dashv\vdash_D^i, U_D^{0i}) \in \mathbb{DMTS}$  for all  $i \in I$ . Then  $\otimes_{i \in I} \mathcal{U}_D^i$  is defined to be the *DMTS*  $(U_\otimes, L, \mapsto_\otimes, \dashv\vdash_\otimes, U_\otimes^0)$ , where*

$$\begin{aligned} U_\otimes &\stackrel{\text{def}}{=} \bigcup_{i \in I} (U_D^i \times \{i\}), \\ L &\stackrel{\text{def}}{=} \bigcup_{i \in I} L^i, \\ (u, i) \mapsto_\otimes \Theta &\stackrel{\text{def}}{\iff} (\forall (a, (u', i')) \in \Theta : i = i') \wedge \\ &\quad u \mapsto_D^i \{(a, u') \mid (a, (u', i)) \in \Theta\}, \\ (u, i) \dashv\vdash_\otimes \{(a, (u', i'))\} &\stackrel{\text{def}}{\iff} i = i' \wedge u \dashv\vdash_D^i \{(a, u')\}, \\ U_\otimes^0 &\stackrel{\text{def}}{=} \bigcup_{i \in I} (U_D^{0i} \times \{i\}). \end{aligned}$$

For any set of *DMTSs*  $\mathbb{U}_D$ , we shortly write  $\otimes \mathbb{U}_D$ , which is defined to be  $\otimes_{\mathcal{U}_D \in \mathbb{U}_D} \mathcal{U}_D$ . If  $\mathbb{U}_D$  has exactly two elements, say  $\mathcal{U}_D^1$  and  $\mathcal{U}_D^2$ , we usually write  $\mathcal{U}_D^1 \otimes \mathcal{U}_D^2$ .

**Lemma 36** *Let  $\mathcal{U}_D \in \mathbb{DMTS}$ . Then  $\mathcal{U}_D \approx_D \otimes \text{Comp}(\mathcal{U}_D)$ .*

**PROOF.** Let  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashrightarrow_D, U_D^0) \in \mathbb{DMTS}$  and  $\otimes \text{Comp}(\mathcal{U}_D) = \otimes_{\mathcal{C} \in \text{Comp}(\mathcal{U}_D)} \mathcal{C} = (U_\otimes, L, \mapsto_\otimes, \dashrightarrow_\otimes, U_\otimes^0)$ . For any component  $\mathcal{C} \in \text{Comp}(\mathcal{U}_D)$ , set  $\mathcal{C} = (U_{\mathcal{C}}^{\mathcal{C}}, L, \mapsto_{\mathcal{C}}^{\mathcal{C}}, \dashrightarrow_{\mathcal{C}}^{\mathcal{C}}, U_{\mathcal{C}}^{0\mathcal{C}})$ .

We start with the proof of  $\mathcal{U}_D \triangleleft_D \otimes \text{Comp}(\mathcal{U}_D)$ . It is straightforwardly shown that  $Q \subseteq U_D \times U_\otimes$ , defined by  $uQ(\hat{u}, \mathcal{C}) \stackrel{\text{def}}{\iff} u = \hat{u}$ , is a disjunctive refinement between  $\mathcal{U}_D$  and  $\otimes \text{Comp}(\mathcal{U}_D)$ .

It remains to prove  $\otimes \text{Comp}(\mathcal{U}_D) \triangleleft_D \mathcal{U}_D$ . It is straightforwardly shown that  $Q \subseteq U_D \times U_\otimes$ , defined by  $(u, \mathcal{C})Q\hat{u} \stackrel{\text{def}}{\iff} u = \hat{u}$ , is a disjunctive refinement between  $\otimes \text{Comp}(\mathcal{U}_D)$  and  $\mathcal{U}_D$ .  $\square$

**Lemma 37** *Let  $\mathcal{U}_D \in \mathbb{DMTS}$ ,  $\mathcal{C}^1 \in \text{Comp}(\mathcal{U}_D)$  and  $\mathcal{C}^2$  be a component such that  $\mathcal{C}^1 \approx_D \mathcal{C}^2$ . Then  $\mathcal{U}_D \approx_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$ .*

**PROOF.** We start with the proof of  $\otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2 \triangleleft_D \mathcal{U}_D$ . We have  $\mathcal{C}^2 \triangleleft_D \mathcal{C}^1$  and by Lemma 33  $\mathcal{C}^1 \triangleleft_D \mathcal{U}_D$ . Transitivity of  $\triangleleft_D$  implies  $\mathcal{C}^2 \triangleleft_D \mathcal{U}_D$ . Since all components of  $\mathcal{U}_D$  disjunctively refine  $\mathcal{U}_D$  (Lemma 33) and  $\mathcal{C}^2$  does as well, we have  $\forall \mathcal{C} \in (\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2 : \mathcal{C} \triangleleft_D \mathcal{U}_D$  and Lemma 32 implies  $\otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2 \triangleleft_D \mathcal{U}_D$ .

Now it remains to prove  $\mathcal{U}_D \triangleleft_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$ . For each  $\mathcal{C} \in \otimes \text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1$ , there obviously exists  $\mathcal{C}' \in \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$  such that  $\mathcal{C} \approx_D \mathcal{C}'$  (simply choose  $\mathcal{C}' = \mathcal{C}$ ). Then Lemma 34 implies  $\forall \mathcal{C} \in \otimes \text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1 : \mathcal{C} \triangleleft_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$ . Again with Lemma 34,  $\mathcal{C}^1 \triangleleft_D \mathcal{C}^2$  implies  $\mathcal{C}^1 \triangleleft_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$ . Consequently  $\forall \mathcal{C} \in \text{Comp}(\mathcal{U}_D) : \mathcal{C} \triangleleft_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$  and Lemma 32 implies  $\mathcal{U}_D \triangleleft_D \otimes(\text{Comp}(\mathcal{U}_D) \setminus \mathcal{C}^1) \cup \mathcal{C}^2$ .  $\square$

Now we are ready to prove Theorem 17(2):

**Theorem 17** (2) *Each refinement of  $\hat{\mathcal{U}}_D$ , shown at the top of Figure 3, is DR-equivalent to one of the DMTSs shown in Figure 3.*

**PROOF.** Let  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashrightarrow_D, U_D^0) \in \mathbb{DMTS}$  such that  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ , where  $\hat{\mathcal{U}}_D = (\{0, 1\}, \{a, b\}, \{(0, \{(a, 1), (b, 1)\})\}, \{(0, \{(a, 1)\}), (0, \{(b, 1)\})\}, \{0\})$ . Choose a disjunctive refinement  $Q \subseteq U_D \times \{0, 1\}$  between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ . We examine an arbitrary component of  $\mathcal{U}_D$ . Thus let  $\mathcal{C} \in \text{Comp}(\mathcal{U}_D)$  and let  $u$  be the root state of  $\mathcal{C}$ . Then we have  $uQ0$ .  $\mathcal{C}$  satisfies the following properties:

- (1) There are no may transitions (and consequently no must transitions) starting in  $u$  with a label different from  $a$  and  $b$ , because otherwise,  $Q$  would not be a refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .

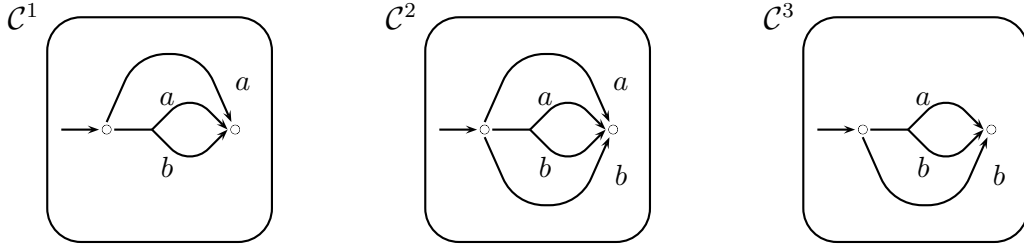


Fig. A.1. Components  $\mathcal{C}^1$ ,  $\mathcal{C}^2$  and  $\mathcal{C}^3$ .

- (2) There are no may transitions (and consequently no must transitions) starting in a state that is a target of a transition starting in  $u$ , because this would again contradict the fact that  $Q$  is a disjunctive refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .
- (3) There is a must transition starting in  $u$ , because otherwise,  $Q$  would not be a disjunctive refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ .
- (4) Properties (2) and (3) imply that  $\mathcal{C}$  is DR-equivalent to a component that has exactly two states, where one of them is the root state, having an outgoing must transition, and the other does not have any outgoing transitions. Due to Lemma 37, it is enough to consider components with exactly two states.

Due to these properties, the only possible components of  $\mathcal{U}_D$  are  $\mathcal{U}_D^1$ ,  $\mathcal{U}_D^2$ ,  $\mathcal{U}_D^3$ ,  $\mathcal{U}_D^7$ ,  $\mathcal{U}_D^9$ ,  $\hat{\mathcal{U}}_D$  (those are marked in Figure 3 by double-lined frames) and the three components  $\mathcal{C}^1$ ,  $\mathcal{C}^2$ ,  $\mathcal{C}^3$  shown in Figure A.1. However,  $\mathcal{C}^1$  is DR-equivalent to  $\mathcal{U}_D^7$ ,  $\mathcal{C}^2$  is DR-equivalent to  $\mathcal{U}_D^2$  and  $\mathcal{C}^3$  is DR-equivalent to  $\mathcal{U}_D^9$ . Consequently we need not consider the components from Figure A.1, because due to Lemma 37, refinements including those components are DR-equivalent to other refinements considered.

Thus all refinements of  $\hat{\mathcal{U}}_D$  are DR-equivalent to a DMTS built up from the components  $\mathcal{U}_D^1, \mathcal{U}_D^2, \mathcal{U}_D^3, \mathcal{U}_D^7, \mathcal{U}_D^9, \hat{\mathcal{U}}_D$ . Now we already have that all refinements with a single component are DR-equivalent to a DMTS shown in Figure 3. DMTSs with more than one component that have  $\hat{\mathcal{U}}_D$  as one of their components need not be considered, because they either do not refine  $\hat{\mathcal{U}}_D$  or are DR-equivalent to  $\hat{\mathcal{U}}_D$ . We consider the remaining DMTSs built up out of two components:

$$\begin{array}{lll}
\hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^2 \approx_D \hat{\mathcal{U}}_D^4 & \hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^3 \approx_D \hat{\mathcal{U}}_D^5 & \hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^7 \approx_D \hat{\mathcal{U}}_D^7 \\
\hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^9 \approx_D \hat{\mathcal{U}}_D^{11} & \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^3 \approx_D \hat{\mathcal{U}}_D^6 & \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^7 \approx_D \hat{\mathcal{U}}_D^{11} \\
\hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^9 \approx_D \hat{\mathcal{U}}_D^{10} & \hat{\mathcal{U}}_D^3 \otimes \hat{\mathcal{U}}_D^7 \approx_D \hat{\mathcal{U}}_D^{10} & \hat{\mathcal{U}}_D^3 \otimes \hat{\mathcal{U}}_D^9 \approx_D \hat{\mathcal{U}}_D^9 \\
\hat{\mathcal{U}}_D^7 \otimes \hat{\mathcal{U}}_D^9 \approx_D \hat{\mathcal{U}}_D^{12} & & 
\end{array}$$

Thus, all refinements built up out of two components are DR-equivalent to a DMTS shown in Figure 3.

We continue with remaining DMTSs built up out of three components. We need not consider combinations including both  $\hat{\mathcal{U}}_D^1$  and  $\hat{\mathcal{U}}_D^7$ , and both  $\hat{\mathcal{U}}_D^3$  and  $\hat{\mathcal{U}}_D^9$ , because both of these pairs is DR-equivalent to a single component from our remaining set and consequently all such combinations have already been considered above.

$$\begin{aligned} \hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^3 &\approx_D \hat{\mathcal{U}}_D^8 & \hat{\mathcal{U}}_D^1 \otimes \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^9 &\approx_D \hat{\mathcal{U}}_D^{11} \\ \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^3 \otimes \hat{\mathcal{U}}_D^7 &\approx_D \hat{\mathcal{U}}_D^{10} & \hat{\mathcal{U}}_D^2 \otimes \hat{\mathcal{U}}_D^7 \otimes \hat{\mathcal{U}}_D^9 &\approx_D \hat{\mathcal{U}}_D^{12} \end{aligned}$$

Thus, all refinements built up out of three components are DR-equivalent to a DMTS shown in Figure 3.

Next, we would have to consider DMTSs build up out of four, respectively five components, where combinations including both  $\hat{\mathcal{U}}_D^1$  and  $\hat{\mathcal{U}}_D^7$ , and both  $\hat{\mathcal{U}}_D^3$  and  $\hat{\mathcal{U}}_D^9$  need not be considered, because they are DR-equivalent to a DMTS built up out of three, respectively four components. However, each such combination includes  $\hat{\mathcal{U}}_D^1$  and  $\hat{\mathcal{U}}_D^7$ , or  $\hat{\mathcal{U}}_D^3$  and  $\hat{\mathcal{U}}_D^9$ , thus there are no further refinements of  $\hat{\mathcal{U}}_D$ .  $\square$

## A.2 Proof of Theorem 26(2)

Definitions and lemmas that are analogous to the ones mentioned in the proof of Theorem 17(2) can be established in a completely analogous fashion for 1MTSs. This includes the definitions of reachability, components and the merge-operator. Lemmas analogous to Lemmas 32, 33, 34, 36 and 37 can be shown as presented in the proof of Theorem 17(2).

**Theorem 26** (2) *Each refinement of  $\hat{\mathcal{U}}_1$ , shown at the top of Figure 4, is DR-equivalent to one of the 1MTSs shown in Figure 4.*

**PROOF.** Let  $\mathcal{U}_1 = (U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0) \in \mathbb{1MTS}$  such that  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1$ , where  $\hat{\mathcal{U}}_1 \stackrel{\text{def}}{=} (\{0, 1\}, \{a, b\}, \{(0, \{(a, 1), (b, 1)\})\}, \{(0, \{(a, 1), (b, 1)\})\}, \{0\})$ . Choose an 1-selecting refinement  $Q \subseteq U_1 \times \{0, 1\}$  between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ . We examine an arbitrary component of  $\mathcal{U}_1$ . Thus let  $\mathcal{C} \in \mathbf{Comp}(\mathcal{U}_1)$  and let  $u$  be the root state of  $\mathcal{C}$ . Then we have  $uQ0$ .  $\mathcal{C}$  satisfies the following properties:

- (1) There are no may (hyper-)transitions starting in  $u$  with a label different from  $a$  and  $b$ , because otherwise,  $Q$  would not be an 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ .
- (2) There are no may (hyper-)transitions starting in a state targeted by a transition starting in  $u$ , because this would again contradict the fact that  $Q$  is an 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ .

- (3) There is a must transition starting in  $u$ , because otherwise,  $Q$  would not be an 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ .
- (4) There are no two may (hyper-)transitions starting in  $u$ , where one of them includes a label  $a$  and the other includes a label  $b$ . If two such transitions existed, then one could not find an appropriate choice function in  $\hat{\mathcal{U}}_1$  for the choice function that selects  $a$  in one of the transitions in  $\mathcal{C}$  and  $b$  in the other.

Property (3) states that there is a must (hyper-)transition starting in  $u$ . By property (1), it has labels  $a$ ,  $b$  or both. Property (4) implies that further transitions can only exist, if all labels of the component are the same. Then the only possible components are the following:

- $\hat{\mathcal{U}}_1$ .
- Components, that have arbitrarily many (hyper-)transitions starting in the root state, where each transition has only labels  $a$ . Due to property (2), all these are 1R-equivalent to  $\mathcal{U}_1^1$ .
- Components, that have arbitrarily many (hyper-)transitions starting in the root state, where each transition has only labels  $b$ . Due to property (2), all these are 1R-equivalent to  $\mathcal{U}_1^2$ .

We now have that for all refinements with only one component there is an 1MTS in Figure 4 that is 1R-equivalent to it. 1MTSs with more than one component that have  $\hat{\mathcal{U}}_1$  as one of their components need not be considered, because they either do not refine  $\hat{\mathcal{U}}_1$  or are 1R-equivalent to  $\hat{\mathcal{U}}_1$ . The only remaining 1MTSs built up out of two components is  $\mathcal{U}_1^1 \otimes \mathcal{U}_1^2$ , which is 1R-equivalent to  $\mathcal{U}_1^3$ . Refinements built up out of more components are 1R-equivalent to refinements already considered.  $\square$

### A.3 Proof of Theorem 28(1)

**Theorem 28** (1)  $f$  is a preorder-based embedding.

**PROOF.** First, we show that for all  $\mathcal{U}_D, \hat{\mathcal{U}}_D \in \mathbb{D}\text{MTS}$  we have  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D \Leftrightarrow f(\mathcal{U}_D) \triangleleft_1 f(\hat{\mathcal{U}}_D)$ . Before proving the two implications of this statement, we introduce some useful notation. Remember that states with indices 0 or 1 refer to the two copies created by the transformation  $f$  (see Definition 27).

- For a target  $\vartheta_D = (a, u')$  in the DMTS and  $k \in \{0, 1\}$ , define  $\vartheta_D \uparrow_k \stackrel{\text{def}}{=} (a, u'_k)$ .
- For a target  $\vartheta_1 = (a, u'_k)$  in the 1MTS, define  $\vartheta_1 \downarrow \stackrel{\text{def}}{=} (a, u')$ .
- For a target set  $\Theta_1$  in the 1MTS, define  $\Theta_1 \downarrow \stackrel{\text{def}}{=} \{\vartheta \downarrow \mid \vartheta \in \Theta_1\}$ .

- For a target set  $\Theta_D$  in the DMTS, define  $[\Theta_D]_1 \stackrel{\text{def}}{=} \{\{\vartheta \uparrow_0 \mid \vartheta \in \Theta_D\} \cup \{\hat{\vartheta} \uparrow_1\} \mid \hat{\vartheta} \in \Theta_D\}$ .

Let  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashrightarrow_D, U_D^0)$ ,  $\hat{\mathcal{U}}_D = (\hat{U}_D, L, \hat{\mapsto}_D, \hat{\dashrightarrow}_D, \hat{U}_D^0) \in \mathbb{DMTS}$  and  $\mathcal{U}_1 = (U_1, L, \mapsto_1, \dashrightarrow_1, U_1^0) \stackrel{\text{def}}{=} f(\mathcal{U}_D)$ ,  $\hat{\mathcal{U}}_1 = (\hat{U}_1, L, \hat{\mapsto}_1, \hat{\dashrightarrow}_1, \hat{U}_1^0) \stackrel{\text{def}}{=} f(\hat{\mathcal{U}}_D)$ .

We now prove the implication “ $\Rightarrow$ ”: Suppose  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ . Then choose a disjunctive refinement  $Q_D$  between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$  and define  $Q_1 \subseteq U_1 \times \hat{U}_1$  by  $u_i Q_1 \hat{u}_j \stackrel{\text{def}}{\Leftrightarrow} u Q_D \hat{u}$ . Then  $Q_1$  is an 1-selecting refinement between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ :

The root state condition is shown as follows: Let  $u_i \in U_1^0$ . Then  $u \in U_D^0$  and due to the root state condition of the disjunctive refinement  $Q_D$ , we can choose  $\hat{u} \in \hat{U}_D^0$  such that  $u Q_D \hat{u}$ . Then  $\hat{u}_0 \in \hat{U}_1^0$  and  $u_i Q_1 \hat{u}_0$ .

Now let  $(u_i, \hat{u}_j) \in Q_1$ . Then  $u Q_D \hat{u}$  and due to property (1) of the disjunctive refinement  $Q_D$ , we can choose a function  $\iota : (u \dashrightarrow_D) \rightarrow (\hat{u} \hat{\dashrightarrow}_D)$  such that for each  $\{\vartheta\} \in (u \dashrightarrow_D)$  we have  $\vartheta Q_D \hat{\vartheta}$ , where  $\{\hat{\vartheta}\} \stackrel{\text{def}}{=} \iota(\{\vartheta\})$ . Furthermore, due to property (2) of the disjunctive refinement  $Q_D$ , we can choose a function  $\tilde{\iota} : (\hat{u} \hat{\dashrightarrow}_D) \rightarrow (u \mapsto_D)$  such that for each  $\hat{\Theta}_D \in (\hat{u} \hat{\dashrightarrow}_D)$  we have  $\forall \vartheta \in \tilde{\iota}(\hat{\Theta}_D) : \exists \hat{\vartheta} \in \hat{\Theta}_D : \vartheta Q_D \hat{\vartheta}$ , and for each  $\hat{\Theta}_D \in (\hat{u} \hat{\dashrightarrow}_D)$ , we can choose another function  $\tilde{\kappa}_{\tilde{\iota}} : \tilde{\iota}(\hat{\Theta}_D) \rightarrow \hat{\Theta}_D$  such that for all  $\vartheta \in \tilde{\iota}(\hat{\Theta}_D)$  we have  $\vartheta Q_D \tilde{\kappa}_{\tilde{\iota}}(\vartheta)$ .

Let  $\gamma \in \text{choice}(u_i \dashrightarrow_1)$ . We need to find an appropriate  $\hat{\gamma} \in \text{choice}(\hat{u}_j \hat{\dashrightarrow}_1)$ . Thus let  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1)$ . In order to define  $\hat{\gamma}(\hat{\Theta}_1)$ , we distinguish the following cases:

- Case 1.**  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1) \setminus (\hat{u}_j \hat{\mapsto}_1)$ . In this case, choose an arbitrary  $\hat{\vartheta} \in \hat{\Theta}_1$  and define  $\hat{\gamma}(\hat{\Theta}_1) \stackrel{\text{def}}{=} \hat{\vartheta}$ .
- Case 2.**  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\mapsto}_1)$ . In this case, choose some  $\vartheta \in \gamma([\tilde{\iota}(\hat{\Theta}_1 \downarrow)]_1)$  and define  $\hat{\gamma}(\hat{\Theta}_1) \stackrel{\text{def}}{=} \tilde{\kappa}_{\tilde{\iota}}(\vartheta \downarrow) \uparrow_0$ .

We have defined  $\hat{\gamma}$  and proceed with checking the properties (1) and (2) of an 1-selecting refinement.

- (1) Let  $\Theta_1 \in (u_i \dashrightarrow_1)$ .
- Case 1.**  $\Theta_1 \notin (u_i \mapsto_1)$ . Then there exists  $\vartheta$  such that  $\{\vartheta\} = \Theta_1 \downarrow \in (u \dashrightarrow_D)$ , and there exists  $\hat{\vartheta}$  such that  $\{\hat{\vartheta}\} = \iota(\{\vartheta\}) \in (\hat{u} \hat{\dashrightarrow}_D)$ . Define  $\hat{\Theta}_1 \stackrel{\text{def}}{=} \{\hat{\vartheta} \uparrow_0, \hat{\vartheta} \uparrow_1\}$ . By definition of  $\iota$ , we have  $\vartheta Q_D \hat{\vartheta}$ . By definition of  $Q_1$ , this implies  $\gamma(\Theta_1) Q_1 \hat{\gamma}(\hat{\Theta}_1)$ , as required.
- Case 2.**  $\Theta_1 \in (u_i \mapsto_1)$ . Define  $\vartheta \stackrel{\text{def}}{=} \gamma(\Theta_1) \downarrow$ . Then  $\vartheta \in \Theta_1 \downarrow \in (u \mapsto_D)$ . The implicit-may condition of DMTSs implies  $\{\vartheta\} \in (u \dashrightarrow_D)$ . There exists  $\hat{\vartheta}$  such that  $\{\hat{\vartheta}\} = \iota(\{\vartheta\}) \in (\hat{u} \hat{\dashrightarrow}_D)$ . Define  $\hat{\Theta}_1 \stackrel{\text{def}}{=} \{\hat{\vartheta} \uparrow_0, \hat{\vartheta} \uparrow_1\}$ . By definition of  $\iota$ , we have  $\vartheta Q_D \hat{\vartheta}$ . By definition of  $Q_1$ , this implies  $\gamma(\Theta_1) Q_1 \hat{\gamma}(\hat{\Theta}_1)$ , as required.
- (2) Let  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\mapsto}_1)$ . By definition of  $\hat{\gamma}$ , we have  $\vartheta \in \gamma([\tilde{\iota}(\hat{\Theta}_1 \downarrow)]_1)$  such

that  $\hat{\gamma}(\hat{\Theta}_1) = \tilde{\kappa}_{\tilde{i}}(\vartheta \downarrow) \uparrow_0$ . Choose  $\Theta_1 \in [\tilde{i}(\hat{\Theta}_1 \downarrow)]_1$  such that  $\vartheta = \gamma(\Theta_1)$ . By definition of  $\tilde{\kappa}_{\tilde{i}}$ , we have  $\vartheta \downarrow \ Q_D \ \tilde{\kappa}_{\tilde{i}}(\vartheta \downarrow)$ . Since  $\vartheta \downarrow = \gamma(\Theta_1) \downarrow$  and  $\tilde{\kappa}_{\tilde{i}}(\vartheta \downarrow) = \hat{\gamma}(\hat{\Theta}_1) \downarrow$ , we get  $\gamma(\Theta_1) \downarrow \ Q_D \ \hat{\gamma}(\hat{\Theta}_1) \downarrow$  and by definition of  $Q_1$ , this implies  $\gamma(\Theta_1) \ Q_1 \ \hat{\gamma}(\hat{\Theta}_1)$ , as required.

It remains to show the implication “ $\Leftarrow$ ”. Suppose  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1$ . Choose an 1-selecting refinement  $Q_1$  between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ . Define  $Q_D \subseteq U_D \times \hat{U}_D$  by  $u Q_D \hat{u} \stackrel{\text{def}}{\iff} \exists i, j \in \{0, 1\} : u_i Q_1 \hat{u}_j$ . We prove that  $Q_D$  is a disjunctive refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .

The root state condition is shown as follows: Let  $u \in U_D^0$ . Then  $u_0 \in U_1^0$  and due to the root state condition of the 1-selecting refinement  $Q_1$ , we can choose  $\hat{u}_i \in \hat{U}_1^0$  such that  $u_0 Q_1 \hat{u}_i$ . Then  $\hat{u} \in \hat{U}_D^0$  and  $u Q_D \hat{u}$ .

Now let  $(u, \hat{u}) \in Q_D$ . Then we can choose  $i, j \in \{0, 1\}$  such that  $u_i Q_1 \hat{u}_j$ . Since  $Q_1$  is a 1-selecting refinement, we can choose a function  $\sigma : \text{choice}(u_i \dashrightarrow_1) \rightarrow \text{choice}(\hat{u}_j \hat{\dashrightarrow}_1)$  such that for all  $\gamma \in \text{choice}(u_i \dashrightarrow_1)$  two functions  $\iota : (u_i \dashrightarrow_1) \rightarrow (\hat{u}_j \hat{\dashrightarrow}_1)$  and  $\tilde{\iota} : (\hat{u}_j \hat{\dashrightarrow}_1) \rightarrow (u_i \dashrightarrow_1)$  can be chosen such that

$$\forall \Theta_1 \in (u_i \dashrightarrow_1) : \gamma(\Theta_1) \ Q_1 \ \sigma(\gamma)(\iota(\Theta_1)) \quad (\text{A.1})$$

and

$$\forall \hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1) : \gamma(\tilde{\iota}(\hat{\Theta}_1)) \ Q_1 \ \sigma(\gamma)(\hat{\Theta}_1). \quad (\text{A.2})$$

- (1) Let  $\{\vartheta\} \in (u \dashrightarrow_D)$  and  $\gamma \in \text{choice}(u_i \dashrightarrow_1)$ . Define  $\Theta_1 \stackrel{\text{def}}{=} \{\vartheta \uparrow_0, \vartheta \uparrow_1\}$  and  $\hat{\Theta}_1 \stackrel{\text{def}}{=} \iota(\Theta_1)$ . Then  $\Theta_1 \in (u_i \dashrightarrow_1)$  and  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1)$ . Due to (A.1), we have

$$\gamma(\Theta_1) \ Q_1 \ \sigma(\gamma)(\hat{\Theta}_1). \quad (\text{A.3})$$

Define  $\hat{\vartheta} \stackrel{\text{def}}{=} \sigma(\gamma)(\hat{\Theta}_1) \downarrow$ . We have  $\{\hat{\vartheta}\} \in (\hat{u} \hat{\dashrightarrow}_D)$ :

**Case 1.** If  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1) \setminus (\hat{u}_j \dashrightarrow_1)$ , we know  $\{\hat{\vartheta}\} \in (\hat{u} \hat{\dashrightarrow}_D)$  (and  $\hat{\Theta}_1 = \{\hat{\vartheta} \uparrow_0, \hat{\vartheta} \uparrow_1\}$ ).

**Case 2.** If  $\hat{\Theta}_1 \in (\hat{u}_j \dashrightarrow_1)$ , then  $\hat{\Theta}_1 \downarrow \in (\hat{u} \dashrightarrow_D)$  and the implicit-may condition of DMTSs implies  $\{\hat{\vartheta}\} \in (\hat{u} \hat{\dashrightarrow}_D)$ .

By definition of  $Q_D$ , (A.3) implies  $\gamma(\Theta_1) \downarrow \ Q_D \ \sigma(\gamma)(\hat{\Theta}_1) \downarrow$ , thus  $\vartheta Q_D \hat{\vartheta}$ , as required.

- (2) Let  $\hat{\Theta}_D \in (\hat{u} \hat{\dashrightarrow}_D)$ . Choose arbitrary  $\hat{\Theta}_1 \in [\hat{\Theta}_D]_1$ . Then  $\hat{\Theta}_1 \in (\hat{u}_j \hat{\dashrightarrow}_1)$ . Let  $\Theta_D \stackrel{\text{def}}{=} \tilde{\iota}(\hat{\Theta}_1) \downarrow$ . Then  $\Theta_D \in (u \dashrightarrow_D)$ . Let  $\vartheta \in \Theta_D$  and consider  $\gamma \in \text{choice}(u_i \dashrightarrow_1)$  that satisfies  $\gamma(\tilde{\iota}(\hat{\Theta}_1)) \downarrow = \vartheta$ . Define  $\hat{\vartheta} \stackrel{\text{def}}{=} \sigma(\gamma)(\hat{\Theta}_1) \downarrow$ . Then  $\hat{\vartheta} \in \hat{\Theta}_D$ . By definition of  $Q_D$ , (A.2) implies  $\vartheta Q_D \hat{\vartheta}$ , as required.

This completes the proof of  $\forall \mathcal{U}_D, \hat{\mathcal{U}}_D \in \text{DMTS} : \mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D \iff f(\mathcal{U}_D) \tilde{\preceq} f(\hat{\mathcal{U}}_D)$ . We continue with showing that  $f$  keeps implementations fixed, i.e.,  $\forall \mathcal{T} \in \text{TS} : f(\pi_{\text{TS,DMTS}}(\mathcal{T})) \approx_1 \pi_{\text{TS,IMTS}}(\mathcal{T})$ . Let  $\mathcal{T} \in \text{TS}$ ,  $\mathcal{U}_1^1 = (U_1^1, L, \dashrightarrow_1^1, \dashrightarrow_1^1, U_1^{01}) \stackrel{\text{def}}{=} \pi_{\text{TS,IMTS}}(\mathcal{T})$  and  $\mathcal{U}_1^2 = (U_1^2, L, \dashrightarrow_1^2, \dashrightarrow_1^2, U_1^{02}) \stackrel{\text{def}}{=} f(\pi_{\text{TS,DMTS}}(\mathcal{T}))$ . We have to prove  $\mathcal{U}_1^1 \triangleleft_1 \mathcal{U}_1^2$  and  $\mathcal{U}_1^2 \triangleleft_1 \mathcal{U}_1^1$ . For the first, it is easily checked that

$Q \subseteq U_1^1 \times U_1^2$ , defined by  $uQ\hat{u}_i \stackrel{\text{def}}{\Leftrightarrow} u = \hat{u}$ , is an 1-selecting refinement between  $\mathcal{U}_1^1$  and  $\mathcal{U}_1^2$ . For the second, it is easily checked that  $Q \subseteq U_1^2 \times U_1^1$ , defined by  $u_iQ\hat{u} \stackrel{\text{def}}{\Leftrightarrow} u = \hat{u}$ , is an 1-selecting refinement between  $\mathcal{U}_1^2$  and  $\mathcal{U}_1^1$ . Both proofs are simple, because only concrete systems are considered and consequently choice functions can only be chosen in a unique way.

Now it remains to check that  $f$  maps relevant abstractions to relevant abstractions, but this is straightforward.  $\square$

#### A.4 Proof of Theorem 30(1)

**Theorem 30** (1)  $g$  is a preorder-based homomorphism.

**PROOF.** Three statements have to be proven: (1)  $\forall \mathcal{U}_1, \hat{\mathcal{U}}_1 : \mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1 \Rightarrow g(\mathcal{U}_1) \triangleleft_D g(\hat{\mathcal{U}}_1)$ , (2) that  $g$  keeps implementations fixed, i.e.,  $\forall \mathcal{T} \in \mathbb{T}\mathbb{S} : g(\pi_{\mathbb{T}\mathbb{S}, \mathbb{1}\mathbb{M}\mathbb{T}\mathbb{S}}(\mathcal{T})) \approx_D \pi_{\mathbb{T}\mathbb{S}, \mathbb{D}\mathbb{M}\mathbb{T}\mathbb{S}}(\mathcal{T})$ , and (3) that  $g$  maps relevant abstractions to relevant abstractions. (2) is obvious, because for concrete systems,  $g$  only renames the states. (3) is also straightforwardly checked. For (1), let  $\mathcal{U}_1 = (U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0)$ ,  $\hat{\mathcal{U}}_1 = (\hat{U}_1, L, \hat{\mapsto}_1, \hat{\dashv}\rightarrow_1, \hat{U}_1^0) \in \mathbb{1}\mathbb{M}\mathbb{T}\mathbb{S}$  and define  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0) \stackrel{\text{def}}{=} g(\mathcal{U}_1)$ ,  $\hat{\mathcal{U}}_D = (\hat{U}_D, L, \hat{\mapsto}_D, \hat{\dashv}\rightarrow_D, \hat{U}_D^0) \stackrel{\text{def}}{=} g(\hat{\mathcal{U}}_1)$ . Suppose  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1$ . Choose an 1-selecting refinement  $Q_1$  between  $\mathcal{U}_1$  and  $\hat{\mathcal{U}}_1$ . For all  $(u, \hat{u}) \in Q_1$ , we can choose a function  $\sigma_{(u, \hat{u})} : \text{choice}(u \dashv\rightarrow_1) \rightarrow \text{choice}(\hat{u} \hat{\dashv}\rightarrow_1)$  such that for all  $\gamma \in \text{choice}(u \dashv\rightarrow_1)$  two functions  $\iota : (u \dashv\rightarrow_1) \rightarrow (\hat{u} \hat{\dashv}\rightarrow_1)$  and  $\tilde{\iota} : (\hat{u} \hat{\mapsto}_1) \rightarrow (u \mapsto_1)$  can be chosen such that  $\forall \Theta_1 \in (u \dashv\rightarrow_1) : \gamma(\Theta_1) Q_1 \sigma_{(u, \hat{u})}(\gamma)(\iota(\Theta_1))$  and  $\forall \hat{\Theta}_1 \in (\hat{u} \hat{\mapsto}_1) : \gamma(\tilde{\iota}(\hat{\Theta}_1)) Q_1 \sigma_{(u, \hat{u})}(\gamma)(\hat{\Theta}_1)$ . Now it is straightforwardly checked that  $Q_D \subseteq U_D \times \hat{U}_D$ , defined by  $(u, \gamma) Q_D (\hat{u}, \hat{\gamma}) \stackrel{\text{def}}{\Leftrightarrow} (uQ_1\hat{u} \wedge \hat{\gamma} = \sigma_{(u, \hat{u})}(\gamma))$ , is a disjunctive refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .  $\square$

#### A.5 Proof of Theorem 30(3)

**Theorem 30** (3)  $g$  is an implementation-based embedding.

**PROOF.** Let  $\mathcal{T} \in \mathbb{T}\mathbb{S}$ ,  $\hat{\mathcal{U}}_1 = (\hat{U}_1, L, \hat{\mapsto}_1, \hat{\dashv}\rightarrow_1, \hat{U}_1^0) \in \mathbb{1}\mathbb{M}\mathbb{T}\mathbb{S}$  and define  $\hat{\mathcal{U}}_D = (\hat{U}_D, L, \hat{\mapsto}_D, \hat{\dashv}\rightarrow_D, \hat{U}_D^0) \stackrel{\text{def}}{=} g(\hat{\mathcal{U}}_1)$ ,  $\mathcal{U}_1 = (U_1, L, \mapsto_1, \dashv\rightarrow_1, U_1^0) \stackrel{\text{def}}{=} \pi_{\mathbb{T}\mathbb{S}, \mathbb{1}\mathbb{M}\mathbb{T}\mathbb{S}}(\mathcal{T})$ ,  $\mathcal{U}_D = (U_D, L, \mapsto_D, \dashv\rightarrow_D, U_D^0) \stackrel{\text{def}}{=} \pi_{\mathbb{T}\mathbb{S}, \mathbb{D}\mathbb{M}\mathbb{T}\mathbb{S}}(\mathcal{T})$ . It is obvious that  $g$  maps relevant abstractions to relevant abstractions. Thus it remains to prove

that  $\mathcal{U}_1 \triangleleft_1 \hat{\mathcal{U}}_1$  if and only if  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ . The implication “ $\Rightarrow$ ” follows from the fact that  $g$  is a preorder-based homomorphism. The other implication (“ $\Leftarrow$ ”) is shown directly: Suppose  $\mathcal{U}_D \triangleleft_D \hat{\mathcal{U}}_D$ . Choose a disjunctive refinement  $Q_D \subseteq U_D \times \hat{U}_D$  between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ . Then it is straightforwardly checked that  $Q_1 \subseteq U_1 \times \hat{U}_1$ , defined by  $uQ_1\hat{u} \stackrel{\text{def}}{\Leftrightarrow} \exists \gamma \in \text{choice}(\hat{\mathcal{U}}_1 \vdash \hat{\rightarrow}_1) : uQ_D(\hat{u}, \gamma)$  is an 1-selecting refinement between  $\mathcal{U}_D$  and  $\hat{\mathcal{U}}_D$ .  $\square$

### A.6 Proof of Theorem 31

**Theorem 31** *There is no preorder-based embedding from 1MTS to DMTS.*

**PROOF.** Assume there is a preorder-based embedding  $h$  from 1MTS to DMTS. We consider  $\hat{\mathcal{U}}_1$  defined at the beginning of Section 5.3. We will show that  $\hat{\mathcal{U}}_1$  and its refinements (which are fully characterized up to refinement equivalence in Theorem 26) cannot be mapped to DMTSs such that  $h$  is a preorder-based embedding, which is a contradiction.

Let  $\mathcal{U}_1^1, \mathcal{U}_1^2, \mathcal{U}_1^3, \hat{\mathcal{U}}_1$  be as illustrated in Figure 4. Since  $h$  is a preorder-based embedding, the following statements hold:  $h(\mathcal{U}_1^1) \approx_D \pi_{\text{TS,DMTS}}(\pi_{1\text{MTS,TS}}(\mathcal{U}_1^1))$ ,  $h(\mathcal{U}_1^2) \approx_D \pi_{\text{TS,DMTS}}(\pi_{1\text{MTS,TS}}(\mathcal{U}_1^2))$ ,  $h(\mathcal{U}_1^3) \triangleleft_D h(\hat{\mathcal{U}}_1)$ ,  $h(\hat{\mathcal{U}}_1) \not\triangleleft_D h(\mathcal{U}_1^3)$ ,  $h(\mathcal{U}_1^3) \not\triangleleft_D h(\mathcal{U}_1^1)$ ,  $h(\mathcal{U}_1^3) \not\triangleleft_D h(\mathcal{U}_1^2)$ . Furthermore, we have for all  $\mathcal{T} \in \mathbb{TS}$ :

$$\pi_{\text{TS,DMTS}}(\mathcal{T}) \triangleleft_D h(\hat{\mathcal{U}}_1) \Leftrightarrow (\pi_{\text{TS,DMTS}}(\mathcal{T}) \approx_D h(\mathcal{U}_1^1) \vee \pi_{\text{TS,DMTS}}(\mathcal{T}) \approx_D h(\mathcal{U}_1^2)) \quad (\text{A.4})$$

by completeness of the characterization of all refinements of  $\hat{\mathcal{U}}_1$  (Theorem 26(2)).  $\hat{\mathcal{U}}_D$  satisfies the following properties:

- (1)  $h(\hat{\mathcal{U}}_1)$  has no components where labels different from  $a$  and  $b$  occur. Otherwise, there were concrete refinements not DR-equivalent to  $h(\mathcal{U}_1^1)$  and not DR-equivalent to  $h(\mathcal{U}_1^2)$ , which would be a contradiction to (A.4).
- (2)  $h(\hat{\mathcal{U}}_1)$  has no components with an outgoing transition from a state that is the target of an outgoing transition from the root state. Otherwise, there were concrete refinements not DR-equivalent to  $h(\mathcal{U}_1^1)$  and not DR-equivalent to  $h(\mathcal{U}_1^2)$ , which would be a contradiction to (A.4). This implies that there are no components with loops and each component of  $h(\hat{\mathcal{U}}_1)$  is DR-equivalent to a component that has at most two states, where one of them is the root state of the component and a possible other one has no outgoing transitions.
- (3)  $h(\hat{\mathcal{U}}_1)$  has no components that have two outgoing may transitions starting in their root state, where one of them is labeled with  $a$  and the other is labeled with  $b$ . Otherwise, it would have a concrete refinement with one transition labeled with  $a$  and another transition labeled with  $b$ , which

is not DR-equivalent to  $h(\mathcal{U}_1^1)$  and not DR-equivalent to  $h(\mathcal{U}_1^2)$ , contradicting (A.4). This implies that  $h(\hat{\mathcal{U}}_1)$  has no components that have two outgoing *must* transitions starting in their root state, where one of them is labeled with  $a$  and the other is labeled with  $b$ . Furthermore, this implies that  $h(\hat{\mathcal{U}}_1)$  has no components that have a must hypertransition with label  $a$  and  $b$  starting in their root state (if there was such a component with such a hypertransition, the implicit-may condition of DMTSs would imply may transitions with labels  $a$  and  $b$ ). As other labels than  $a$  and  $b$  are not possible (property (1) in this list), we cannot have hypertransitions with different labels. However, if all labels in a hypertransition are equal, then the second property in this list implies that each component is DR-equivalent to a component with no hypertransitions at all.

- (4)  $h(\hat{\mathcal{U}}_1)$  has a component with a root state that has an outgoing may transition with label  $a$ . Otherwise,  $h(\mathcal{U}_1^1)$  would not be a concrete refinement of  $h(\hat{\mathcal{U}}_1)$ , which would be a contradiction to (A.4). Furthermore,  $h(\hat{\mathcal{U}}_1)$  has a component with a root state with outgoing may transition with label  $b$ . Otherwise,  $h(\mathcal{U}_1^2)$  would not be a concrete refinement of  $h(\hat{\mathcal{U}}_1)$ , which would be a contradiction to (A.4).
- (5) The root state of each component must have an outgoing must transition. Otherwise, the DMTS with no transitions would be a concrete refinement that is DR-equivalent to  $h(\mathcal{U}_1^1)$  and not DR-equivalent to  $h(\mathcal{U}_1^1)$ , which would be a contradiction to (A.4).

By property (5), each component has a must transition. By property (3), it is not a hypertransition and there are no further may or must transitions in the component. By property (1), the must transition is labeled with either  $a$  or  $b$ . Then, together with (4), we have one component with a must transition labeled with  $a$  and one component with a must transition labeled with  $b$ . Possibly existing further components must be DR-equivalent to one of these two components. Thus  $h(\hat{\mathcal{U}}_1)$  is DR-equivalent to  $\mathcal{U}_D^5$  from Figure 3. Then Theorem 17 implies that there is no refinement of  $h(\hat{\mathcal{U}}_1)$  that is not DR-equivalent to  $h(\hat{\mathcal{U}}_1)$ , not DR-equivalent to  $h(\mathcal{U}_1^1)$  and not DR-equivalent to  $h(\mathcal{U}_1^2)$ . This, however, is a contradiction, because  $h(\mathcal{U}_1^3)$  satisfies these properties.  $\square$

## References

- [1] E. M. Clarke, O. Grumberg, D. E. Long, Model checking and abstraction, *ACM Transactions on Programming Languages and Systems* 16 (5) (1994) 1512–1542.
- [2] N. Wirth, Program development by stepwise refinement, *Communications of the ACM* 14 (4) (1971) 221–227.
- [3] R. Milner, *A Calculus for Communicating Systems*, Vol. 92 of LNCS, Springer-Verlag, 1980.

- [4] R. Milner, An algebraic definition of simulation between programs, in: International Joint Conference on Artificial Intelligence, 1971, pp. 481–489.
- [5] N. Lynch, F. Vaandrager, Forward and backward simulations I. Untimed systems, *Inf. Comput.* 121 (2) (1995) 214–233.
- [6] K. G. Larsen, B. Thomsen, A modal process logic, in: LICS '88, IEEE Computer Society Press, 1988, pp. 203–210.
- [7] D. Dams, Abstract interpretation and partition refinement for model checking, Ph.D. thesis, Technische Universiteit Eindhoven, The Netherlands (1996).  
URL <http://cm.bell-labs.com/who/dennis/Thesis/the.ps>
- [8] D. Dams, R. Gerth, O. Grumberg, Abstract interpretation of reactive systems, *ACM Transactions on Programming Languages and Systems* 19 (2) (1997) 253–291.
- [9] G. Bruns, P. Godefroid, Generalized model checking: Reasoning about partial state spaces, *LNCS* 1877 (2000) 168–182.
- [10] K. G. Larsen, L. Xinxin, Equation solving using modal transition systems, in: LICS '90, IEEE Computer Society Press, 1990, pp. 108–117.
- [11] S. Shoham, O. Grumberg, Monotonic abstraction-refinement for CTL, in: K. Jensen, A. Podelski (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 2988 of LNCS, Springer-Verlag, 2004, pp. 546–560.
- [12] P. Godefroid, R. Jagadeesan, On the expressiveness of 3-valued models, in: L. Zuck, P. Attie, A. Cortesi, S. Mukhopadhyay (Eds.), *VMCAI 2003*, Vol. 2575 of LNCS, Springer-Verlag, 2003, pp. 206–222.
- [13] L. de Alfaro, P. Godefroid, R. Jagadeesan, Three-valued abstractions of games: Uncertainty, but with precision, in: LICS '04 [25], pp. 170–179.
- [14] D. Dams, K. S. Namjoshi, The existence of finite abstractions for branching time model checking, in: LICS '04 [25], pp. 335–344.
- [15] D. Janin, I. Walukiewicz, Automata for the modal  $\mu$ -calculus and related results, in: J. Wiedermann, P. Hájek (Eds.), *Mathematical Foundations of Computer Science*, Vol. 969 of LNCS, Springer-Verlag, 1995, pp. 552–562.
- [16] D. Dams, K. S. Namjoshi, Automata as abstractions, in: R. Cousot (Ed.), *Verification, Model Checking, and Abstract Interpretation*, Vol. 3385 of LNCS, Springer-Verlag, 2005, pp. 216–232.
- [17] S. Shoham, O. Grumberg, 3-valued abstraction: More precision at less cost., in: LICS '06, IEEE Computer Society Press, 2006, pp. 399–410.
- [18] H. Fecher, M. Huth, Ranked predicate abstraction for branching time: Complete, incremental, and precise., in: *ATVA*, Vol. 4218 of LNCS, Springer-Verlag, 2006, pp. 322–336.
- [19] H. Fecher, M. Huth, More precise partition abstraction., in: *VMCAI*, Vol. 4349 of LNCS, Springer-Verlag, 2007, pp. 167–181.

- [20] S. Graf, H. Saidi, Construction of abstract state graphs with PVS, in: O. Grumberg (Ed.), CAV, Vol. 1254 of LNCS, Springer-Verlag, 1997, pp. 72–83.
- [21] R. Eshuis, M. M. Fokkinga, Comparing refinements for failure and bisimulation semantics, *Fundam. Inf.* 52 (4) (2002) 297–321.
- [22] R. Cleaveland, M. Hennessy, Testing equivalence as a bisimulation equivalence, *Formal Aspects of Computing* 5 (1993) 1–20.
- [23] K. Larsen, A. Skou, Bisimulation through probabilistic testing, *Inf. Comput.* 94 (1) (1991) 1–28.
- [24] H. Fecher, M. Steffen, Characteristic  $\mu$ -calculus formula for an underspecified transition system, in: EXPRESS'04, Vol. 128 of Electronic Notes in Theoretical Computer Science, Elsevier Science Publishers, 2005, pp. 103–116.
- [25] Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, 2004.