Characteristic μ-Calculus Formulas for Underspecified Transition Systems ⋆

Harald Fecher and Martin Steffen

Christian-Albrechts-University Kiel, Germany
{*hf,ms*}*@informatik. uni-kiel. de*

Abstract

Underspecification, which is essential for specification formalisms, is usually expressed by equivalences, simulations, or logic approaches. We introduce underspecified transition systems (UTSs) as general model general model for underspecification, where, e.g., transitions point to sets of states. We argue for the generality of the UTSs by showing that the class of all UTSs is strictly more expressive than the standard equivalences and simulation approaches, in the sense that more sets of transition systems can be expressed. Additionally, a characteristic formula in terms of the μ-calculus is presented for every finite state UTS. Furthermore, we show that UTSs can finitely describe sets of transition systems, whenever they can be described finitely by the other standard approaches except for trace-set extension or μ-calculus descriptions.

Key words: underspecification, transition systems, bisimulation, simulation, μ-calculus

1 Introduction

Specification languages abstract away from program details and describe programs on an abstract level. Hence, underspecification is an important feature of specification languages. For instance, program are often specified by pre- and post conditions, which usually allow different possible implementations, in other words, underspecification is used.

Action based systems are typically specified by temporal logics, like the μ-calculus [10], or by using more operational approaches, like labeled transition systems [9]. In logical approaches, underspecification appears naturally, as usually not the system is specified in all detail. In labeled transition systems, on the other hand, equivalence or simulation relations are usually used to

⋆ Part of this work has been financially supported by IST project Omega (IST-2001-33522) and NWO/DFG project Mobi-J (RO 1122/9-1, RO 1122/9-2).

achieve underspecification. Unfortunately, approaches based standard equivalence or simulation relations are insufficient to describe relevant underspecification appearing in practice, since they can either describe safety or liveness properties but no combinations of these properties. For example, specifying that action a must be possible, action b may be possible and no other action is allowed [1] cannot be done by standard equivalences or simulation approaches [2]. Therefore, a more expressive operational based formalism is needed in order to handle more practically relevant underspecification. Such formalisms are also suitable as semantic domains for modeling languages like UML [19,20].

Therefore, we introduce an extended version of labeled transition systems, called underspecified transition systems or UTSs for short. They already were investigated in different context under the name of *disjunctive modal transition systems* [13] and they are a generalization of Larsen and Thomsen's *modal transition systems* [11,12]. In other words we have must transition that must be matched by the implementation and we have may transition that can be matched by the implementation (but it is not necessary to match them). Furthermore, the implementation may not do more as the must and may transition allow. Our generalization of this approach is that (must/may) transitions of a UTS points from a state to a set of combinations of actions and states. The meaning is that one, but not necessarily all of the action/state combination (a, s') of the set of a transition from s has to be matched in the implementation. This generalization is necessary to model underspecification that appear, for example, in specification given by pre- and postconditions where we know that one action specified by the pre- and postcondition has to be possible but not necessarily all of them.

In this paper, we examine the expressive power of the class of all UTSs in the sense that we compare the sets of transition systems describable by UTSs using U-bisimulation with the standard equivalence and simulation techniques and with the μ-calculus. Note that U-bisimulation denotes our exact definition that an implementation in terms of transition systems satisfy the specification in terms of UTSs. For reason of simplicity, we do not consider weak versions, i.e., we do not abstract away from internal execution. In particular, we show that UTSs are strictly more expressive than transition systems compared by trace inclusion, trace-set extension, bisimulation or simulation. Moreover, we show that the UTS can be chosen to be finite, whenever the transition system of the trace inclusion, bisimulation or simulations specification is finite. This is in general not the case for trace-set extension. Furthermore, we show that every set of transition systems describable by a finite UTS can be described by

[1] For example the implementation that only allow a and the implementation that only allow a and b satisfy this specification.

[2] This simple example can be specified by using two different specification one for safety and one for liveness. Nevertheless, if specification with a more complicated branching structure are considered, simulation techniques are not sufficient, e.g., bisimulation is different from simulation-equivalence [4].

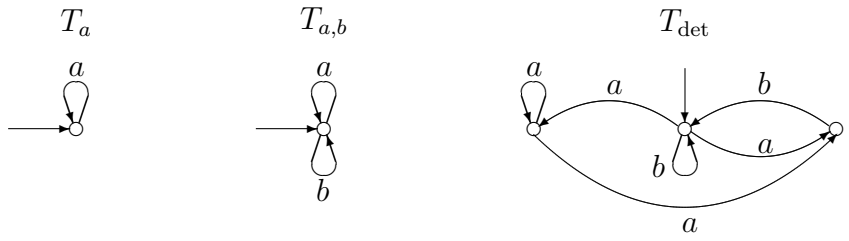

Fig. 1. Some Transition Systems

a μ-calculus formula. This is done by defining characteristic formulas for finite UTSs. On the other hand, we show that not every set of transition systems describable by a μ-calculus formula can be described by a finite UTS.

In outline, the paper is organized as follows: Section 2 introduces transition systems together with the concepts of trace-inclusion, trace-set extension, bisimulation, and simulation. Underspecified transition systems together with a corresponding notion of bisimulation, called U-bisimulation, are presented in Section 3. This section also compares UTSs with the concepts presented in Section 2 wrt. their expressive power. After a short review of the μ-calculus in Section 4, the characteristic formula of a UTS is given in Section 5, where we also show that not all μ-calculus formulas can be described by finite UTSs. Section 6 concludes the paper, discussing related and future work.

2 Transition Systems

Labeled transition system [9] are defined as follows:

Definition 2.1 A (labeled) *transition system* (TS) is a tuple $(S, \mathcal{L}, \longrightarrow, \overline{s})$ where S is the set of *states*, $\mathcal{L}$ the set of *labels*, $\longrightarrow \subseteq S \times \mathcal{L} \times S$ is the *transition relation*, and $\overline{s} \in S$ is the *root state*.
A TS is *finite* if both the set of states and the sets of labels are finite.

Some transition systems are illustrated in Figure 1. Transition systems are used for underspecification, i.e., they can describe sets of transition systems via equivalence and simulation relations. Standard representatives of these approaches are given in the next subsections.

2.1 Traces

The typical equivalence notion in non-branching systems is trace equivalence:

Definition 2.2 The set of traces of a transition system $T = (S, \mathcal{L}, \longrightarrow, \overline{s})$ is defined by

$$\mathrm{Tr}(T) = \{(a_1, ..., a_n) \in \mathcal{L}^\star \mid \exists s_0, ..., s_n : s_0 = \overline{s} \wedge \forall i < n : s_i \xrightarrow{a_{i+1}} s_{i+1}\}.$$

The transition systems T_1 and T_2 are called *trace equivalent* if $\mathrm{Tr}(T_1) = \mathrm{Tr}(T_2)$.

The transition systems $T_{a,b}$ and T_{det} of Figure 1 are trace equivalent. In the context of trace semantics, a TS is used for underspecification in that it is meant to describe the set of all TS trace equivalent to it. Specification based on equivalence is not always enough, since it does not allow less (respectively more) behavior: all traces have to be matched. For example, we cannot specify that the system has at least trace a or that the system cannot have more traces as trace a and the empty trace. To gain more flexibility, trace set inclusion has been considered:

Definition 2.3 Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS over the same set of labels. Then T_1 is *trace included* in T_2 (and T_2 is *trace-set extended* from T_1) if $\text{Tr}(T_1) \subseteq \text{Tr}(T_2)$.

Now, the set of all TSs that only allow the execution of actions a and b is obtained by taking all TS that are trace included in $T_{a,b}$ of Figure 1. On the other hand, the set of all TS that are trace-set extended from $T_{a,b}$ are those TS exhibiting at least all traces generated from actions a and b.

2.2 Bisimulation

Traces as specification capture the linear behavior of a system and are too abstract when the branching structure is of import. The fundamental notion of equivalence in branching systems is *bisimulation* [16,21]:

Definition 2.4 Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS over the same set of labels. A *bisimulation* between T_1 and T_2 is a relation $\widetilde{R} \subseteq S_1 \times S_2$ such that $(\overline{s}_1, \overline{s}_2) \in \widetilde{R}$, and for all pairs $(s_1, s_2) \in \widetilde{R}$ and labels $a \in \mathcal{L}$ we have

- if $s_1 \xrightarrow{a} s_1'$, then $s_2 \xrightarrow{a} s_2'$ and $(s_1', s_2') \in \widetilde{R}$ for some s_2', and conversely,
- if $s_2 \xrightarrow{a} s_2'$, then $s_1 \xrightarrow{a} s_1'$ and $(s_1', s_2') \in \widetilde{R}$ for some s_1'.

Two transition systems are called *bisimilar* or bisimulation equivalent, if there exists a bisimulation relating them.

A TS is used for specification in the sense that it describes the set of all TS that are bisimilar to it. The standard technique in branching systems to allow less (respectively more) behavior is simulation:

2.3 Simulation

Bisimulation can be seen as a symmetric variant of the notion of simulation, which was introduced in [15]:

Definition 2.5 Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS over the same set of labels. A *simulation* from T_1 to T_2 is a relation $\widehat{R} \subseteq S_1 \times S_2$ such that $(\overline{s}_1, \overline{s}_2) \in \widehat{R}$, and for all pairs $(s_1, s_2) \in \widehat{R}$ and labels $a \in \mathcal{L}$ we have

$$\text{if } s_1 \xrightarrow{a} s_1', \text{ then } s_2 \xrightarrow{a} s_2' \text{ and } (s_1', s_2') \in \widehat{R}, \text{ for some } s_2' \ .$$

T_2 *liveness-simulates* T_1 (and T_1 *safety-simulates* T_2) if there exists a simulation from T_1 to T_2, relating their root states.

For example, the set of all TSs that safety-simulate T_{det} of Figure 1 are those TSs that and after an a-action, and not both a and b are possible. In particular, $T_{a,b}$ of Figure 1 is not included. Note that transitions system in simulation relation with T_{det} must have exactly a and b as set of labels. The set of all TS that liveness-simulates $T_{a,b}$ are those TSs where at every reachable state both actions a and b are possible, e.g., T_{det} is not included.

Relating systems by simulation for underspecification is still too coarse, since all behavior of the specification can be neglected when the safety-simulation is used, respectively arbitrary behavior can be added in the case of liveness-simulation. In other words, only safety or liveness properties can be specified but no combinations of them. [3] In particular the following example cannot be specified using simulations.

Example 2.6 Consider the set of all TS where at the beginning action a has to be possible, action b is allowed but no further actions are allowed at the beginning.

That the set of TS described by this example cannot be described by trace inclusion, trace-set extension or simulations can be seen as follows: If trace inclusion or safety-simulation is used, then no a is necessary. If trace-set extension or liveness-simulation is used, then also action c is allowed.

3 Underspecified Transition Systems

To handle situations of underspecification as given in Example 2.6, one can use transition systems with *two different kinds* of transitions. One to denote the steps mandatory for the implementation, called *must transitions*, and a second transition relation to indicate steps which may occur, but that are not necessary for the implementation, called *may transitions*. Such transition systems are called *modal transition systems* [11]. Modal transition systems, however, are not sufficient to model all behaviors that appear in practice: Consider, for example, a system required to send a value v, denoted by the action $send(v)$. If this value v is specified by a pre- and postcondition, e.g., it should be between 4 and 6, then one action of $send(4)$, $send(5)$, or , $send(6)\}$, possibly leading to different states, has to be allowed but not necessarily all of them. This cannot be specified by the modal transition systems, since if no must transition exists, then no sending has to be present, but if a must transition with label $send(i)$ is used, then exactly this action $send(i)$ has to be present, which does not reflect the specification.

[3] Using two different specifications for safety and liveness properties will not be sufficient if systems with complicated branching structures are considered, e.g., bisimulation is different from simulation-equivalence [4].

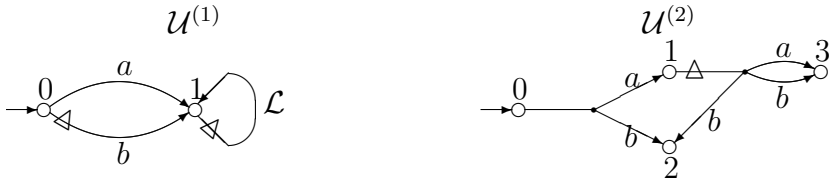

Fig. 2. Two Underspecified Transition Systems

In order to solve this problem, we generalize transitions from a relation between states, labels, and states to a relation between states and sets of combinations of labels and states. This idea is formalized in the following definition, where the special element $\triangle$ is contained in the set of combinations of labels and states iff this transition encodes a may transition.

Definition 3.1 [UTS] An *underspecified transition system* (UTS) is a tuple $(U, \mathcal{L}, \longmapsto, \overline{U})$, where U is the set of states, $\mathcal{L}$ the set of labels, $\longmapsto \subseteq U \times \big(\mathcal{P}((\mathcal{L} \times U) \cup \{\triangle\}) \setminus \{\emptyset\}\big)$ is the *underspecified transition relation*, where $\mathcal{P}(\_)$ denotes the power set construction and $\_ \setminus \_$ denotes the element removal function, and $\overline{U} \subseteq U$ is the *set of root states*.
A transition $u \longmapsto M$ is called *must transition*, if $\triangle \notin M$ and *may transition*, if $\triangle \in M$.
A UTS is *finite* if its set of states and its set of labels are finite. A UTS is *fully determined,* if it does not have may transitions and if the right-hand side of every must transition and as well as the set of root states contain exactly one element.

Underspecified transition system were also investigated under the name of disjunctive modal transition systems in [13] and they are further considered in the context of abstraction techniques in [22,3]. The intuitive meaning of a UTS is made clear by defining the set of TS that satisfy a UTS. This will be done in the next subsection, where it is also explained why the empty set is not allowed in the underspecified transition relation. A graphical notation of UTSs is given by dividing the head of an arrow such that the heads pointing to the target states and such that the labels are drawn behind the division (we use set of labels when the arrows corresponding to these labels pointing to the same state). Furthermore, the beginning of an arrow is decorated by the symbol $\triangle$ if the arrow corresponds to a may transition. The UTSs

$$\mathcal{U}^{(1)} = \big(\{0,1\}, \mathcal{L}, \{(0, \{(a,1)\}), (0, \{(b,1), \triangle\}), (1, \{\triangle\} \cup (\mathcal{L} \times \{1\}))\}, 0\big)$$
$$\mathcal{U}^{(2)} = \big(\{0,1,2,3\}, \mathcal{L}, \{(0, \{(a,1), (b,2)\}), (1, \{(a,3), (b,2), (b,3), \triangle\})\}, 0\big)$$

are drawn in Figure 2, where $a, b \in \mathcal{L}$.

Proposition 3.2 *The class of all fully determined UTSs corresponds to the class of all TS. Furthermore, the isomorphism ι^{b} is obtained by mapping the transition system $(S, \mathcal{L}, \longrightarrow, \overline{s})$ to $(S, \mathcal{L}, \{(s, \{(a, s')\}) \mid s \xrightarrow{a} s'\}, \{\overline{s}\})$.*

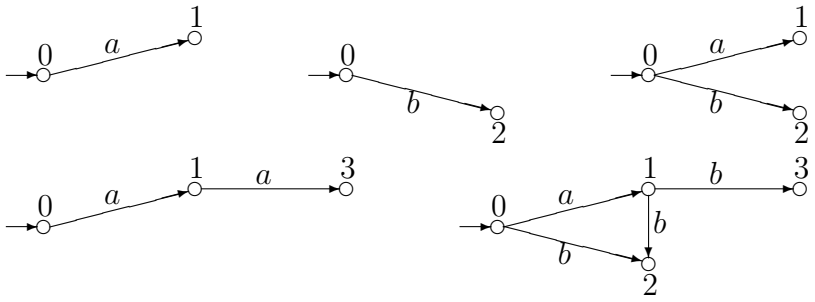

Fig. 3. Transition System U-bisimilar to the UTS of Figure 2

3.1 Underspecified Bisimulation

Next we formalize when an implementation given as TSsatisfies a specification defined as UTS.

Definition 3.3 [U-bisimulation] Let $T = (S, \mathcal{L}, \longrightarrow, \overline{s})$ be a TS and $\mathcal{U} = (U, \mathcal{L}, \longmapsto, \overline{U})$ be a UTS over the same set of labels. An *underspecified bisimulation* or U-bisimulation for short, between T and $\mathcal{U}$ is a relation $R \subseteq S \times U$ such that $\exists \overline{u} \in \overline{U} : (\overline{s}, \overline{u}) \in R$ and for all $(s, u) \in R$ we have

- $s \xrightarrow{a} s' \Rightarrow \exists M' : u \longmapsto M' \wedge \exists u' : (a, u') \in M' \wedge (s', u') \in R$, and
- $(u \longmapsto M' \wedge \triangle \notin M') \Rightarrow \exists (a, u') \in M' : \exists s' : s \xrightarrow{a} s' \wedge (s', u') \in R$.

T and $\mathcal{U}$ are called *underspecified bisimilar* (U-bisimilar) if there exists a U-bisimulation between them.

The first equation in the above definition ensures that every state s' in the implementation that can be reached by an execution $s \xrightarrow{a} s'$ has an equivalent counterpart u' in the specification that is not forbidden to be reached with an a-action ($\exists M' : u \longmapsto M' \wedge \exists u' : (a, u') \in M'$). On the other hand, for every must step of the specification ($u \longmapsto M' \wedge \triangle \notin M'$) there is an element in the right hand side ($(a, u') \in M'$) such that its state has an equivalent counterpart s' which can be reached by an a-step ($s \xrightarrow{a} s'$). This makes clear that underspecified transitions of form $u \longmapsto \emptyset$ are useless, and therefore forbidden, since they cannot be matched by any implementation. Note that may steps of the specification do not have to be matched by the implementation, but can be used to match steps of the implementation. Obviously, transitions of the form $u \longmapsto \{\triangle\}$ are redundant and can be omitted. Transition systems that are U-bisimilar to $\mathcal{U}^{(2)}$ of Figure 2 are presented in Figure 3. On the other hand, the transition system that only consists of state 0 and possesses no transitions is not U-bisimilar to $\mathcal{U}^{(2)}$.

Furthermore, the set of all TS described in Example 2.6 is the set of all TS that are U-bisimilar to $T^{(1)}$ of Figure 2. Note that it would be sufficient to restrict may transitions to singleton sets, i.e., may transitions be elements of $U \times \mathcal{L} \times U$. We decided to take the more general approach in order to simplify the notations (must and may transitions are encoded within the same relation).

Proposition 3.4 *U-bisimilarity is closed under bisimilarity, i.e., if T_1 is bisimilar to T_2 and T_2 is U-bisimilar to $\mathcal{U}$ then T_1 is U-bisimilar to $\mathcal{U}$.*

For each UTS, there always exists a U-bisimilar TS. More precisely, the transition system obtained by removing all may transitions and by choosing an element from every must transition yields a U-bisimilar one. Formally:

Proposition 3.5 *Let $\mathcal{U} = (U, \mathcal{L}, \longmapsto, \overline{U})$ be a UTS and f be a function from $\longmapsto \cap (U \times \mathcal{P}((\mathcal{L} \times U)))$ to $\mathcal{L} \times U$ with $\forall (u, M) \in (\longmapsto \cap (U \times \mathcal{P}((\mathcal{L} \times U)))) : f((u, M)) \in M$. Then the transition system $(U, \mathcal{L}, \{(u, a, u') \mid \exists M : f((u, M)) = (a, u')\}, \overline{u})$ is U-bisimilar to $\mathcal{U}$ whenever $\overline{u} \in \overline{U}$.*

The following proposition gives the justification to call the relation of Definition 3.3 U-bisimulation, even if it is not an equivalence relation.

Proposition 3.6 *Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS. Then T_1 is bisimilar to T_2 iff T_1 is U-bisimilar to $\iota^{\mathrm{b}}(T_2)$, with ι^{b} as given in Proposition 3.2.*

Weak versions of bisimulation, like *weak bisimulation*, *delay bisimulation*, *η-bisimulation* and *branching bisimulation* (see, e.g., [5]), can be analogously defined for U-bisimulation.

3.2 UTS versus Simulation

¿From Proposition 3.6, we obtain that every set of transition systems derived from bisimulation equivalence can also be described by U-bisimulation such that finiteness of the specification is preserved, i.e., the UTS can be chosen to be finite whenever the used TS is finite. Furthermore, simulations can also be described by U-bisimulation such that finiteness of the specification is preserved, which is stated in the following propositions.

Proposition 3.7 *Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS. Then T_1 safety-simulates T_2 iff T_1 is U-bisimilar to $\iota^{\mathrm{s}}(T_2)$, where ι^{s} is obtained by mapping the transition system $(S, \mathcal{L}, \longrightarrow, \overline{s})$ to $(S, \mathcal{L}, \{(s, \{(a, s'), \triangle\}) \mid s \xrightarrow{a} s'\}, \{\overline{s}\})$.*

Proposition 3.8 *Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS. Then T_1 liveness-simulate T_2 iff T_1 is U-bisimilar to $\iota^{\mathrm{g}}(T_2)$, where ι^{g} is obtained by mapping the transition system $(S, \mathcal{L}, \longrightarrow, \overline{s})$ to $(S, \mathcal{L}, \{(s, \{(a, s')\}) \mid s \xrightarrow{a} s'\} \cup \{(s, \{(a, u_{\mathrm{true}}), \triangle\}) \mid a \in \mathcal{L} \wedge s \in S \cup \{u_{\mathrm{true}}\}\}, \{\overline{s}\})$.*

3.3 UTS versus Traces

Trace inclusion can be approximated by safety-simulation (and consequently by U-bisimulation) such that finiteness of the specification is preserved:

Proposition 3.9 *Let $T_i = (S_i, \mathcal{L}, \longrightarrow_i, \overline{s}_i)$ with $i \in \{1, 2\}$ be two TS. Then T_1 is trace included in T_2 iff T_1 is U-bisimilar to $\iota^{\mathrm{ts}}(T_2)$, where ι^{ts} is obtained by*

mapping the transition system $(S, \mathcal{L}, \longrightarrow, \overline{s})$ *to*
$\Big(\mathcal{P}(S), \mathcal{L}, \Big\{(\widetilde{S}, \{(a, \widetilde{S}'), \triangle\}) \mid \emptyset \neq \widetilde{S}' \wedge \widetilde{S}' = \{s' \mid \exists s \in \widetilde{S} : s \xrightarrow{a} s'\}\Big\}, \{\overline{s}\}\Big)$.

Obviously, every set of TSs obtained by trace-set extension can be described by UTS, since every set of TSs closed under bisimulation can be described by an UTS. This can be done by taking the disjoint union of their states and their 'transition' and take the root state set as the collection of their roots. But this technique in general yields an infinite UTS. Nevertheless, is it possible to characterize sets of TSs obtained by trace-set-extension from finite TSs by finite UTSs? The following proposition gives a negative answer to this question:

Proposition 3.10 *The set of all TSs that are trace-set extended from* T_a *of Figure 1 cannot be described by a finite UTS using U-bisimulation.*

¿From these propositions, we obtain that the class of all UTSs with U-bisimulation is strictly more expressive (in the sense that more sets of transition systems are describable) than by trace-inclusion, bisimulation, and simulation, since the set of transition systems described in Example 2.6 cannot be described by these approaches.

4 μ-Calculus

In this section, we represent the well-known μ-calculus [10,2]: Let $\mathcal{V}\mathrm{ar}^\mu$ be a set of *logical variables*. The formulas of the μ-calculus are given by the following grammar:

$$\phi ::= \mathrm{true} \mid \mathrm{false} \mid X \mid \langle a \rangle \phi \mid [a]\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mu X.\phi \mid \nu X.\phi,$$

where $X \in \mathcal{V}\mathrm{ar}^\mu$ and $a \in \mathcal{L}$. The set of all formulas is denoted by $\mathcal{F}$. A formula is *closed,* if every occurrence of a variable X occurs inside a formula of form $\mu X.\phi$ or $\nu X.\phi$. The semantics of the μ-calculus is given as follows:

Definition 4.1 Let $T = (S, \mathcal{L}, \longrightarrow, \overline{s})$ be a TS. Then the interpretation function $[\![\_]\!]_T : \mathcal{F} \times (\mathcal{V}\mathrm{ar}^\mu \to \mathcal{P}(S)) \to \mathcal{P}(S)$ is defined as:

$$\begin{aligned}
[\![\mathrm{false}]\!]^\rho_T &= \emptyset \\
[\![\mathrm{true}]\!]^\rho_T &= S \\
[\![X]\!]^\rho_T &= \rho_{(X)} \\
[\![\langle a \rangle \phi]\!]^\rho_T &= \{s \in S \mid \exists s' \in [\![\phi]\!]^\rho_T : s \xrightarrow{a} s'\} \\
[\![[a]\phi]\!]^\rho_T &= \{s \in S \mid \forall s' \in S : s \xrightarrow{a} s' \Rightarrow s' \in [\![\phi]\!]^\rho_T\} \\
[\![\phi_1 \wedge \phi_2]\!]^\rho_T &= [\![\phi_1]\!]^\rho_T \cap [\![\phi_2]\!]^\rho_T \\
[\![\phi_1 \vee \phi_2]\!]^\rho_T &= [\![\phi_1]\!]^\rho_T \cup [\![\phi_2]\!]^\rho_T \\
[\![\mu X.\phi]\!]^\rho_T &= \bigcap \{M \in \mathcal{P}(S) \mid [\![\phi]\!]^{\rho[X \mapsto M]}_T \subseteq M\} \\
[\![\nu X.\phi]\!]^\rho_T &= \bigcup \{M \in \mathcal{P}(S) \mid [\![\phi]\!]^{\rho[X \mapsto M]}_T \supseteq M\}
\end{aligned}$$

where $\rho[X \mapsto M]$ denotes the function equals ρ except on X where it is equal to M.

A transition system $T = (S, \mathcal{L}, \longrightarrow, \overline{s})$ satisfies a closed μ-calculus formula ϕ, written as $T \models \phi$, if $\overline{s} \in [\![\phi]\!]^{\rho}_{T}$ for some ρ.

5 The Characteristic μ-Calculus Formula of a UTS

To illustrate that the μ-calculus is more expressive than UTSs, we define characteristic μ-calculus formulas for UTSs:

Definition 5.1 Let $\mathcal{U} = (U, \mathcal{L}, \longmapsto, \overline{U})$ be a finite UTS. The transformation function $\Upsilon_{\mathcal{U}} : \mathcal{P}(U) \times U \to \mathcal{F}$ (where we assume that $U \subset \mathcal{V}\mathrm{ar}^{\mu}$) is given by

$$\Upsilon_{\mathcal{U}}(V,u) = u \qquad \text{if } u \in V$$

$$\Upsilon_{\mathcal{U}}(V,u) = \nu u.\left(\bigwedge_{M : u \longmapsto M \wedge \Delta \notin M}\left(\bigvee_{(a,u')\in M}\langle a\rangle \Upsilon_{\mathcal{U}}(V\cup\{u\},u')\right)\wedge \bigwedge_{a\in\mathcal{L}}[a]\left(\bigvee_{u' : \exists M : u \longmapsto (M\cup(a,u'))}\Upsilon_{\mathcal{U}}(V\cup\{u\},u')\right)\right) \qquad \text{if } u \notin V$$

The *characteristic formula* of $\mathcal{U}$ is given by $\phi_{\mathcal{U}} = \bigvee_{\overline{u}\in\overline{U}} \Upsilon_{\mathcal{U}}(\emptyset, \overline{u})$.

The characteristic formula is well-defined, since $|U| < \infty$ and $|\mathcal{L}| < \infty$. It is also easily seen that the characteristic formula is closed. The characteristic formula of $T^{(1)}$ of Figure 2 is:

$$\nu X_0.\big(\langle a\rangle\phi_1^{(1)}\big) \wedge \big([a]\phi_1^{(1)}\big) \wedge \big([b]\phi_1^{(1)}\big) \wedge \bigwedge_{c\in\mathcal{L}\setminus\{a,b\}}[c]\mathrm{false},$$

which is equivalent to $\langle a\rangle\mathrm{true} \wedge \bigwedge_{c\in\mathcal{L}\setminus\{a,b\}}[c]\mathrm{false}$, and where state i is denoted by X_i and $\phi_1^{(1)} = \nu X_1.\bigwedge_{c\in\mathcal{L}}[c]X_1)$. The characteristic formula of $T^{(2)}$ of Figure 2 is

$$\nu X_0.\Big(\big((\langle a\rangle\phi_1^{(2)}) \vee (\langle b\rangle\phi_2^{(2)})\big) \wedge ([a]\phi_1^{(2)}) \wedge ([b]\phi_2^{(2)}) \wedge \bigwedge_{c\in\mathcal{L}\setminus\{a,b\}}[c]\mathrm{false}\Big) \quad \text{with}$$

$$\phi_1^{(2)} = \nu X_1.\Big(([a]\phi_3^{(2)}) \wedge ([b](\phi_3^{(2)} \vee \phi_2^{(2)})) \wedge \bigwedge_{c\in\mathcal{L}\setminus\{a,b\}}\mathrm{false}\Big)$$

$$\phi_2^{(2)} = \nu X_2.\bigwedge_{c\in\mathcal{L}}[c]\mathrm{false} \qquad \phi_3^{(2)} = \nu X_3.\bigwedge_{c\in\mathcal{L}}[c]\mathrm{false}.$$

Theorem 5.2 *For all transition systems $T = (S, \mathcal{L}, \longrightarrow, \overline{s})$ and for all finite underspecified transition systems $\mathcal{U} = (U, \mathcal{L}, \longmapsto, \overline{U})$ we have*

$$T \text{ is U-bisimilar to } \mathcal{U} \text{ iff } T \models \phi_{\mathcal{U}}$$

Note that not all sets of TSs describable by infinite UTSs can be described by a μ-calculus formula. This follows immediately from the fact that a set of bisimilar TSs that do not have a finite representation cannot be characterized by a μ-calculus formula, as shown in [18]. Furthermore, the μ-calculus can describe sets of TSs that cannot be described by finite UTSs:

Proposition 5.3 *The set of all TSs that satisfy the μ-calculus formula*

$$\widehat{\phi} = \mu X.\Big((\langle a\rangle X) \vee (\langle b\rangle \text{true})\Big)$$

cannot be described by a finite UTS using U-bisimulation.

6 Conclusion and Related Work

We examined the expressive power of UTSs with respect to the set of transition system describable by them. We showed that they are strictly more expressive than specifications based on trace inclusion, bisimulation, and simulation. Therefore, UTSs yield a nice specification formalism, since it allows standard underspecification techniques in a single setting and allow even more detailed specifications. Furthermore, we showed that there are sets of TSs obtained from a finite transition system using trace-set extension that can only be described by infinite UTSs. We presented characteristic μ-calculus formulas for UTS s and showed that not all μ-calculus formulas can be represented by finite UTSs. We do not consider UTSs with propositions, but they can be straightforwardly extended to deal with them.

In Section 3, we already discussed the modal transition systems of [11]. A process algebra that deals with underspecification is introduced in [25] (see also [14]). Their process algebra has, in particular, an alternative operator, where exactly one of its argument has to be implemented. This is contrary to our approach where we also allow that more than one alternative is provided by the implementation. Another form of underspecification is the usage of orders on actions [24], where the implementation may have a better action as the recommended one. Our approach is more flexible, since it can use different orders at every state.

By Hennessy and Milner [7], bisimilarity of (image-finite) processes can be characterized by sets of μ-calculus formulas that does not contain recursion. The characterization of bisimilarity for finite CCS processes [17] by a single formula was investigated in [6]. This was generalized in [23] to a construction of a characteristic formula for finite-state processes. The direct derivation of characteristic formulas from classic greatest fixpoint characterization of bisimulation is presented in [18].

Future work is to examine weak versions of U-bisimulation. Another future direction is to investigate a modified version of UTSs where the transitions are interpreted such that exactly one instead of at least one element of the right hand side of the transitions must (respectively may) be used for the

implementation. Further investigation of algebraic formalisms like process algebras [17,8], that can handle underspecification, is also of interest. In our opinion, the class of all UTSs yields an appropriate semantical language for such kinds of formalisms.

References

[1] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, 2001.

[2] J. Bradfield and C. Stirling. Modal logics and mu-calculi: An introduction. In Bergstra et al. [1], pages 293–330.

[3] L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2004.

[4] R. v. Glabbeek. The linear time–branching time spectrum I. The semantics of concrete, sequential processes. In Bergstra et al. [1], pages 3–99.

[5] R. v. Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[6] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68(1–3):125–145, 1986.

[7] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[8] C. A. R. Hoare. *Communications Sequential Processes*. International Series in Computer Science. Prentice Hall, 1985.

[9] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19:371–384, 1976.

[10] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[11] K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of the 3rd Annual IEEE Symposium on Logic in Computer Science*, pages 203–210. IEEE Computer Society Press, 1988.

[12] K. G. Larsen and B. Thomsen. Partial specifications and compositional verification. *Theoretical Computer Science*, 88:15–32, 1991.

[13] K. G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science*, pages 108–117. IEEE Computer Society Press, 1990.

[14] M. E. Majster-Cederbaum. Underspecification for a simple process algebra of recursive processes. *Theoretical Computer Science*, 266:935–950, 2001.

[15] R. Milner. An algebraic definition of simulation between programs. In *IJCAI 71*, pages 481–489. British Computer Society, 1971.

[16] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[17] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[18] M. Müller-Olm. Derivation of characteristic formulae. In *MFCS'98 Workshop on Concurrency*, volume 18 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 1998.

[19] Object Management Group. *UML 2.0 Infrastructure Specification*, 2003. `http://www.omg.org/cgi-bin/doc?ptc/03-09-15`.

[20] Object Management Group. *UML 2.0 Superstructure Specification*, 2003. `http://www.omg.org/cgi-bin/doc?ptc/03-08-02`.

[21] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

[22] S. Shoham and O. Grumberg. Monotonic abstraction-refinement for CTL. In K. Jensen and A. Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2988 of *LNCS*, pages 546–560. Springer-Verlag, 2004.

[23] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

[24] B. Thomsen. An extended bisimulation induced by a preorder on actions. Master's thesis, Aalborg University Centre, 1987.

[25] S. Veglioni and R. De Nicola. Possible worlds for process algebras. In D. Sangiorgi and R. de Simone, editors, *CONCUR '98: Concurrency Theory*, volume 1466 of *LNCS*, pages 179–193. Springer-Verlag, 1998.