State Space Representation for Verification of Open Systems

Irem Aktug and Dilian Gurov

KTH Computer Science and Communication
Osquars Backe 2, 100 44
Stockholm, Sweden
{irem, dilian}@nada.kth.se

Abstract. When designing an open system, there might be no implementation available for certain components at verification time. For such systems, verification has to be based on assumptions on the underspecified components. When component assumptions are expressed in Hennessy-Milner logic (HML), the state space of open systems can be naturally represented with modal transition systems (MTS), a graphical specification language equiexpressive with HML. Having an explicit state space representation supports state space exploration based verification techniques. Besides, it enables proof reuse and facilitates visualization for the user guiding the verification process in interactive verification. As an intuitive representation of system behavior, it aids debugging when proof generation fails in automatic verification.

However, HML is not expressive enough to capture temporal assumptions. For this purpose, we extend MTSs to represent the state space of open systems where component assumptions are specified in modal μ-calculus. We present a two-phase construction from process algebraic open system descriptions to such state space representations. The first phase deals with component assumptions, and is essentially a maximal model construction for the modal μ-calculus. In the second phase, the models obtained are combined according to the structure of the open system to form the complete state space. The construction is sound and complete for systems with a single unknown component and sound for those without dynamic process creation. For establishing open system properties based on the representation, we present a proof system which is sound and complete for prime formulae.

1 Introduction

In an *open system*, certain components can join the system after it has been put in operation. For example, applications can be loaded on a smart card after the card has been issued (see e.g. [SGH04]). Since the implementations of certain components are not yet available, the verification of the system has to be based on behavioural assumptions on such components. Security protocols can be verified in this manner, for instance by treating an unpredictable attacker as an unknown component of the system [1].

M. Johnson and V. Vene (Eds.): AMAST 2006, LNCS 4019, pp. 5–20, 2006.

Modal transition systems (MTS) were introduced by Larsen as a graphical specification language [2]. Certain kinds of properties are easier to express graphically than in temporal logics. Each MTS specifies a set of processes as an interval determined by necessary and admissable transitions. MTSs are equiexpressive with Hennessy-Milner logic (HML), i.e. an HML formula can be characterized by an MTS and vice versa. As a result, MTSs provide a natural representation of the state space of open systems when assumptions on the behavior of the not-yet-available components are specified in HML. When the assumptions are temporal properties, however, MTSs are not expressive enough for this purpose. In [3], we extend MTSs to represent the state space of open systems when the component assumptions are written in the modal μ-calculus [4]. This logic adds the expressive power of least and greatest fixed point recursion to HML. Besides the *must* (necessary) and *may* (admissable) transitions of MTS, our notion, *extended modal transition system* (EMTS) has sets of states (instead of single states) as targets to transitions - an extension which is needed for dealing with disjunctive assumptions, and well-foundedness constraints to handle least fixed point assumptions.

Having a way to capture the state space of an open system explicitly can be useful in various phases of the development of open systems. In *the modeling phase*, this formalism can be used as an alternative means of graphical specification of open system behavior. In *interactive verification*, an explicit state space representation facilitates visualization of the system behaviour, assisting the user in guiding the proof. This visualization facility is beneficial in *automatic verification* when the automatic proof construction fails and an understanding of the open system behaviour becomes necessary for debugging. Furthermore, computing the whole state space enables proof reuse when the same system is to be checked for several properties.

In this paper, we address the problem of constructing an explicit state space representation from an open system description and verifying open system properties based on this representation. In a process algebraic setting, the behaviour of an open system can be specified by an *open process term with assumptions* (OTA). An OTA consists of a process term equipped with a list of behavioral assumptions on the free variables of the term. We offer a two-phase construction that, under given restrictions, automatically extracts an EMTS from an OTA. The first phase in the construction corresponds to a maximal model construction for each component assumption. For the fixed point cases, a powerset construction is used that is similar to the one used in the Büchi automata constructions of [5] and [6]. In the second phase, the maximal models are composed according to the structure of the open system. The construction is *sound* (resp. *complete*) if the set of systems denoted by the OTA is a subset (resp. superset) of the denotation of the resulting EMTS. We show soundness of the construction for systems without dynamic process creation, and soundness and completeness for systems with a single unknown component. Finally, we present a proof system for showing open system properties based on EMTSs. The proof system is sound and complete for *prime* formulae, a prime formula being one that logically im-

plies one of the disjuncts whenever it logically implies a disjunction. The relative simplicity of the proof system and its use is an indication of the adequateness of EMTSs for open system state space representation.

Related Work. In this strand of research, our work follows earlier work on using maximal model constructions for modular verification for various fragments of the μ-calculus: for ACTL by Grumberg and Long [7], ACTL* by Kupferman and Vardi [8], and the fragment without least fixed points and diamond modalities by Sprenger et al [9]. In automata based approaches (see for instance [10, 6, 11]), various structures like alternating tree automata, Büchi and Rabin automata have been employed for capturing temporal properties. Although expressively powerful, we argue that these structures do not provide an intuitive representation of the state space for branching-time logics.

Proof system based methods have previously been suggested for the interactive verification of open systems [12, 13] where modal μ-calculus is used to express the temporal assumptions on components as well as the desired property of the system. These interactive methods explore the state space implicitly as much as it is necessary for the particular verification task. In contrast to these methods, we separate the tasks of constructing a finite representation of the state space of an open system from the task of verifying its properties. This separation provides a state visualization facility to the user guiding the interactive proof, and offers greater possibilities for proof reuse.

Organization. The paper is organized as follows. In section 2, we make the syntax of OTAs precise by a brief account of the logic used in behavioral assumptions and the process algebra used to define the process term. Section 3 is a summary of important definitions related to the notion of EMTS. We present the translation from OTA to EMTS in Section 4, and provide correctness results. In Section 5, we give a proof system for showing open system properties of EMTSs. The last section presents conclusions and identifies directions for future work.

2 Specifying Open Systems Behaviour

A system, the behaviour of which is parameterized on the behaviour of certain components, is conveniently represented as a pair $\Gamma \rhd E$, where E is an open process-algebraic term, and Γ is a list of assertions of the shape $X : \Phi$ where X is a process variable free in E and Φ is a closed formula in a process logic.

In the present study, we work with the class of Basic Parallel Processes (BPP)[14]. The terms of BPP are generated by:

$$E ::= \mathbf{0} \mid X \mid a.E \mid E + E \mid E \parallel E \mid \mathit{fix}X.E$$

where X ranges over a set of process variables $\mathit{ProcVar}$ and a over a finite set of actions A. We assume that $\mathit{ProcVar}$ is partitioned into assumption process variables $\mathit{AssProcVar}$ used in assertions, and recursion process variables $\mathit{RecProcVar}$ bound by *fix*. A term E is called *linear* if every assumption process variable occurs in E at most once. The operational semantics of closed process terms (called

processes and ranged over by t) is standard, where the operator $\|$ signifies *merge composition*. Closed process terms give rise to *labeled transition systems* (LTS) through this standard semantics.

As a process logic for specifying behavioural assumptions of components, as well as for specifying system properties to be verified, we consider the modal μ-calculus [4]. Its formulas are generated by:

$$\Phi ::= \mathrm{tt} \mid \mathrm{ff} \mid Z \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [a]\,\Phi \mid \langle a \rangle\,\Phi \mid \nu Z.\Phi \mid \mu Z.\Phi$$

where Z ranges over a set of propositional variables *PropVar*. The semantics of the μ-calculus is standard and given in terms of its denotation on some LTS $\mathcal{T}$=($S_{\mathcal{T}}$, A, $\longrightarrow_{\mathcal{T}}$). The denotation of a modal μ-calculus formula Φ, written $||\Phi||^{\mathcal{T}}_{\mathrm{V}}$, is a subset of the set of states of $\mathcal{T}$, where $\mathrm{V} : PropVar \rightarrow S_{\mathcal{T}}$ is a valuation that maps propositional variables to states of $\mathcal{T}$. As usual, we write $t \models^{\mathcal{T}}_{\mathrm{V}} \Phi$ whenever $t \in ||\Phi||^{\mathcal{T}}_{\mathrm{V}}$. In the sequel, we omit the subscript V when Φ is a closed formula.

We say that an OTA $\Gamma \rhd E$ is *guarded* when the term E and all modal μ-calculus formula Φ in Γ are guarded. Similarly, we say an OTA is linear when the term it contains is linear.

The behaviour specified by an open term with assumptions is given with respect to a LTS $\mathcal{T}$ that is closed under the transition rules and is closed under substitution of processes for assumption process variables in subterms of the OTA. The denotation of an OTA is then the set of all processes obtained by substituting each assumption process variable in the term by a process from $\mathcal{T}$ satisfying the respective assumptions.

Definition 1 (OTA Denotation). *Let $\Gamma \rhd E$ be an OTA, $\mathcal{T}$ be an LTS, and $\rho_R : RecProcVar \rightarrow S_{\mathcal{T}}$ be a recursion environment. The* denotation *of $\Gamma \rhd E$ relative to $\mathcal{T}$ and ρ_R is defined as:*

$$[\![\Gamma \rhd E]\!]_{\rho_R} \triangleq \{E\rho_R\rho_A \mid \forall (X : \Phi) \in \Gamma.\ \rho_A(X) \models^{\mathcal{T}} \Phi\}$$

where $\rho_A : AssProcVar \rightarrow S_{\mathcal{T}}$ ranges over assumption environments.

Example. Consider an operating system in the form of a concurrent server that spawns off *Handler* processes each time it receives a request. These processes run system calls for handling the given requests to produce a result (modeled by the action $\overline{out}$). *Handler* is defined as $Handler \stackrel{def}{=} In \parallel \overline{out}.\mathbf{0}$ where $In \stackrel{def}{=} in.In$. Although it is possible to communicate with request handlers through the attached channel (modeled by the action *in*), they do not react to further input. A property one would like to prove of such a server is that it stabilizes whenever it stops receiving new requests. Eventual stabilization can be formalized in the modal μ-calculus as $\mathbf{stab} \stackrel{\Delta}{=} \nu X.\mu Y.\,[in]\,X \wedge \langle\overline{out}\rangle\,Y$. We can reduce this verification task to proving that the open system modeled by the OTA

$$X : \mathbf{stab} \rhd X \parallel Handler$$

which consists of *Handler* and any stabilizing process X, eventually stabilizes.

3 Extended Modal Transition Systems

In [3], we proposed Extended Modal Transition Systems (EMTS) as an explicit state space representation for open systems with temporal assumptions, with an extensional representation for the well-foundedness constraints. In this section, we summarize the main definitions, and propose a concrete representation of well-foundedness constraints. The notion of EMTS is based on Larsen's Modal Transition Systems [2]. Kripke Modal Transition Systems (KMTS) have been first introduced by Huth et. al. [15], and later refined by Grumberg and Shoham [16] for representing state space abstractions in an abstraction refinement framework. EMTS is similar to KMTS with a constraint added to deal with termination assumptions.

In addition to may and must transitions for dealing with modalities, EMTSs include sets of states (instead of single states) as targets to transitions to capture disjunctive assumptions, and a set of prohibited infinite runs defined through a coloring function to represent termination assumptions.

Definition 2 (EMTS). *An* extended modal transition system *is a structure*

$$\mathcal{E} = (S_\mathcal{E}, A, \longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}}, c)$$

where (i) $S_\mathcal{E}$ *is a set of* abstract states*, (ii)* A *is a set of* actions*, (iii)* $\longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}} \subseteq S_\mathcal{E} \times A \times 2^{S_\mathcal{E}}$ *are* may *and* must transition relations*, and (iv)* $c : S_\mathcal{E} \to \mathbb{N}^k$ *is a* coloring function *for some* $k \in \mathbb{N}$*.*

May transitions of an EMTS show possible behaviours of the closed systems represented, while must transitions specify behaviour shared by all these closed systems. A *run* (or may–run) of $\mathcal{E}$ is a possibly infinite sequence of transitions $\rho_\mathcal{E} = s_0 \xrightarrow{a_0}_{\mathcal{E}} s_1 \xrightarrow{a_1}_{\mathcal{E}} s_2 \xrightarrow{a_2}_{\mathcal{E}} \ldots$ where for every $i \geq 0$, $s_i \overset{a_i}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S$ for some S such that $s_{i+1} \in S$. Must–runs are defined similarly. We distinguish between two kinds of *a-derivatives* of a state s: $\partial^{\Diamond}_a(s) \triangleq \{S \mid s \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S\}$ and $\partial^{\Box}_a(s) \triangleq \{S \mid s \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S\}$.

The coloring function c specifies a set $W_\mathcal{E}$ of prohibited infinite runs, which plays a similar role to fairness constraints of e.g. [7], by means of a *parity acceptance condition* (cf. [17, 10]). The function c is extended to infinite runs so that $c(\rho_\mathcal{E}) = (c(s_0)(1) \cdot c(s_1)(1) \ldots, \ldots, c(s_0)(k) \cdot c(s_1)(k) \ldots)$ is a k-tuple of infinite words where $c(s)(j)$ denotes the j^{th} component of $c(s)$. Let $inf(c(\rho_\mathcal{E})(i))$ denote the set of infinitely occurring colors in the i^{th} word of this tuple. Then the run $\rho_\mathcal{E}$ is prohibited, $\rho_\mathcal{E} \in W_\mathcal{E}$, if and only if $max\,(inf\,(c(\rho_\mathcal{E})(i)))$ is odd for some $1 \leq i \leq k$, i.e. the greatest number that occurs infinitely often in one of these k infinite words is odd.

Next, we define a simulation relation between the states of an EMTS as a form of mixed fair simulation (cf. e.g. [7, 18]).

Definition 3 (Simulation). $R \subseteq S_\mathcal{E} \times S_\mathcal{E}$ *is a* simulation relation *between the states of* $\mathcal{E}$ *if whenever* $s_1 R s_2$ *and* $a \in A$*:*

1. *if* $s_1 \xrightarrow{a}{}^{\Diamond}_{\mathcal{E}} S_1$, *then there is a* S_2 *such that* $s_2 \xrightarrow{a}{}^{\Diamond}_{\mathcal{E}} S_2$ *and for each* $s_1' \in S_1$, *there exists a* $s_2' \in S_2$ *such that* $s_1' R s_2'$;
2. *if* $s_2 \xrightarrow{a}{}^{\Box}_{\mathcal{E}} S_2$, *then there is a* S_1 *such that* $s_1 \xrightarrow{a}{}^{\Box}_{\mathcal{E}} S_1$ *and for each* $s_1' \in S_1$, *there exists a* $s_2' \in S_2$ *such that* $s_1' R s_2'$;
3. *if the run* $\rho_{s_2} = s_2 \xrightarrow{a_1}_{\mathcal{E}} s_2^1 \xrightarrow{a_2}_{\mathcal{E}} s_2^2 \xrightarrow{a_3}_{\mathcal{E}} \ldots$ *is in* $W_{\mathcal{E}}$ *then every infinite run* $\rho_{s_1} = s_1 \xrightarrow{a_1}_{\mathcal{E}} s_1^1 \xrightarrow{a_2}_{\mathcal{E}} s_1^2 \xrightarrow{a_3}_{\mathcal{E}} \ldots$ *such that* $s_1^i R\, s_2^i$ *for all* $i \geq 1$ *is also in* $W_{\mathcal{E}}$.

We say that abstract state s_2 *simulates* abstract state s_1, denoted $s_1 \preceq s_2$, if there is a simulation relation R such that $s_1 R s_2$. Simulation can be generalized to two different EMTSs $\mathcal{E}_1$ and $\mathcal{E}_2$ in the natural way.

Labeled transition systems can be viewed as a special kind of EMTS, where: $\longrightarrow^{\Box}_{\mathcal{E}} = \longrightarrow^{\Diamond}_{\mathcal{E}}$, the target sets of the transition relation are singleton sets of states, and the set of prohibited runs W is empty. We give the meaning of an abstract state relative to a given LTS, as the set of concrete LTS states simulated by the abstract state.

Definition 4 (Denotation). *Let* $\mathcal{E}$ *be an EMTS, and let* $\mathcal{T}$ *be an LTS. The* denotation *of abstract state* $s \in S_{\mathcal{E}}$ *is the set* $[\![s]\!]_{\mathcal{T}} \triangleq \{t \in S_{\mathcal{T}} \mid t \preceq s\}$. *This notion is lifted to sets of abstract states* $S' \subseteq S_{\mathcal{E}}$ *in the natural way:* $[\![S']\!]_{\mathcal{T}} \triangleq \bigcup\{[\![s]\!]_{\mathcal{T}} \mid s \in S'\}$.

In the rest of the paper, we shall assume that EMTSs obey the following *consistency* restrictions: $\longrightarrow^{\Box}_{\mathcal{E}} \subseteq \longrightarrow^{\Diamond}_{\mathcal{E}}$, $s \xrightarrow{a}{}^{\Box}_{\mathcal{E}} S$ implies S is non-empty, and W does not contain runs corresponding to infinite must–runs of the EMTS. The meaning of abstract states would not be altered if the targets of may transitions were restricted to singletons, but we prefer the targets of both kinds of transitions to be sets of states for reasons of uniformity.

In section 5, we present a proof system for proving properties of abstract states. For this purpose, we define when an abstract state s satisfies a modal μ-calculus formula Φ. The global nature of the set W in EMTSs makes it cumbersome to define the denotation of a fixed point formula compositionally as a set of abstract states. We therefore give an indirect definition of satisfaction, by means of the denotation $[\![s]\!]_{\mathcal{T}}$ of a state s.

Definition 5 (Satisfaction). *Let* $\mathcal{E}$ *be an EMTS,* $s \in S_{\mathcal{E}}$ *be an abstract state of* $\mathcal{E}$ *and* Φ *be a modal* μ*-calculus property. Then* s *satisfies* Φ *under valuation* $\mathcal{V} : PropVar \to 2^{S_{\mathcal{E}}}$, *denoted* $s \models^{\mathcal{E}}_{\mathcal{V}} \Phi$, *if and only if for any LTS* $\mathcal{T}$ $[\![s]\!]_{\mathcal{T}} \models^{\mathcal{T}}_{\mathrm{V}} \Phi$ *where valuation* $\mathrm{V} : PropVar \to 2^{S_{\mathcal{T}}}$ *is induced by* $\mathcal{V}$ *as* $\mathrm{V}(Z) \triangleq \bigcup\{[\![s]\!]_{\mathcal{T}} \mid s \in \mathcal{V}(Z)\}$.

Example. The state space of the open system introduced in the previous section is captured by the EMTS in Figure 1. For any labeled transition system $\mathcal{T}$, the processes simulated by the state s_1 are those denoted by the open term $X : \mathbf{stab} \rhd X \parallel Handler$. The EMTS consists of six abstract states, each state

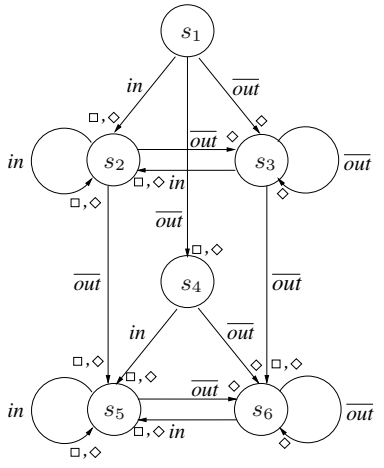

Fig. 1. EMTS for $X : \boldsymbol{stab} \rhd X \parallel Handler$

denoting the set of processes which it simulates. For instance, states s_5 and s_6 in the example denote all processes which can engage in arbitrary interleavings of *in* and $\overline{out}$ actions, but so that *in* has to be enabled throughout while $\overline{out}$ has not. Infinite runs stabilizing on $\overline{out}$ actions are prohibited by the coloring of s_3 and s_6. A proof of eventual stabilization of the system using this representation can be found in [19].

4 From OTA to EMTS

In this section, we address the problem of providing an explicit state space representation for a given open term $\Gamma \rhd E$, by means of an EMTS $\mathcal{E}$. While it is tempting to define $\longrightarrow^{\Diamond}_{\mathcal{E}}$ and $\longrightarrow^{\Box}_{\mathcal{E}}$ through transition rules, the global nature of the well-foundedness constraints suggests that a direct construction would be more convenient for automatic construction. We propose a two-phase construction ε that translates an open term $\Gamma \rhd E$ to an EMTS, denoted $\varepsilon(\Gamma \rhd E)$. In the first phase, an EMTS is constructed for each underspecified component. This part is essentially a maximal model construction as developed by Grumberg and Long for ACTL [7], extended to ACTL* by Kupferman and Vardi [8], and applied by Sprenger et al to the fragment of the modal μ-calculus without least fixed points and diamond modalities [9]. For the construction of the fixed point cases, we adapt a powerset construction used earlier to convert fragments of the modal μ-calculus to Büchi automata which was introduced by Dam [5] for linear time μ-calculus and extended by Kaivola [6] to the Π_2 fragment. The second phase consists of combining the EMTSs produced in the first step according to the structure of the term E. We then show the correctness of the construction by relating the set of states simulated by the constructed EMTS to the denotation of the given OTA.

4.1 Maximal Model Construction

We define the function ε which maps modal μ-calculus formulas to triples of the shape $(\mathcal{E}, S, \lambda)$, where $\mathcal{E} = (S_{\mathcal{E}}, A, \longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}}, c)$ is an *EMTS*, $S \subseteq S_{\mathcal{E}}$ is a set of *start states* of $\mathcal{E}$, and $\lambda : S_{\mathcal{E}} \to 2^{PropVar}$ is a *labeling function*.

The function is defined inductively on the structure of Φ as shown in Table 1. The meaning of open formulae that arises in intermediate steps are given by the valuation which assigns the whole set of processes $S_{\mathcal{T}}$ to each propositional variable. Essentially, the particular valuation used does not play a role in the final EMTS, since the properties used as assumptions of an OTA are closed.

In the definition, let $\varepsilon(\Phi_1)$ be $((S_{\mathcal{E}_1}, A, \longrightarrow^{\Diamond}_{\mathcal{E}_1}, \longrightarrow^{\Box}_{\mathcal{E}_1}, c_1), S_1, \lambda_1)$ and $\varepsilon(\Phi_2)$ be $((S_{\mathcal{E}_2}, A, \longrightarrow^{\Diamond}_{\mathcal{E}_2}, \longrightarrow^{\Box}_{\mathcal{E}_2}, c_2), S_2, \lambda_2)$ where $S_{\mathcal{E}_1}$ and $S_{\mathcal{E}_2}$ are disjoint sets. The new state s_{new} is not in $S_{\mathcal{E}_1}$ and a and a' are actions in A. The coloring functions $c_1 : S_{\mathcal{E}_1} \to \mathbb{N}^{k_1}$ and $c_2 : S_{\mathcal{E}_2} \to \mathbb{N}^{k_2}$ color the states of $\mathcal{E}_1$ and $\mathcal{E}_2$ with integer tuples of length k_1 and k_2 respectively.

For a set S, $S\,|_{\Box}$ denotes the largest transition-closed set contained in S such that there is no element $s \in S\,|_{\Box}$ with the empty set as the target to a must transition, that is, there is no s such that for some $a \in A$, $s \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} \emptyset$ and each state s is reachable from some start state.

In what follows, we explain the various cases of the construction. The EMTS for formula tt consists of the single state s_{tt} with may transitions to itself for every action, while the EMTS for ff is the empty EMTS. The EMTS for a propositional variable consists of a single start state with may transitions to s_{tt} for each action.

The states of the EMTS for the conjunction of two formulas is the cross product of the states of the EMTSs constructed for each conjunct, excluding pairs with incompatible capabilities. If a state s_1, which has a must transition for an action a to some set S_1, is producted with a state s_2 that has multiple may transitions for a, then the product state has a must a-transition to the product of S_1 with the set of all may-successors of s_2. The color of a state of $\varepsilon(\Phi_1 \wedge \Phi_2)$ is the concatenation of the colors of the paired states. In the case of disjunction, the set of start states of $\varepsilon(\Phi_1 \vee \Phi_2)$ is the union of the start states of $\varepsilon(\Phi_1)$ and $\varepsilon(\Phi_2)$ which reflects the union of their denotation. The color of a state is given by padding with 0's from either the left or right.

For the modal cases, a new state s_{new} is set as the start state. The EMTS for $\varepsilon([a]\,\Phi)$ has a single may transition for a, which is to the set of initial states of $\varepsilon(\Phi)$. This is to ensure all simulated processes satisfy Φ after engaging in an a. Additionally, there is a may transition to s_{tt} for all other actions. The EMTS for $\varepsilon(\langle a\rangle\,\Phi)$ includes a must transition for a from this start state to the start states of $\varepsilon(\Phi)$, along with may transitions for all actions to s_{tt} forcing the simulated processes to have an a transition to some process satisfying Φ and allowing any other transitions besides.

The construction for fixed point formulae is a powerset construction, which is similar to the constructions given in [5] and [6] for the purpose of constructing Büchi Automata for linear time and the alternation-depth class Π_2 fragments of the μ-calculus, respectively. The states of $\varepsilon(\sigma Z.\Phi)$ consist of sets of states of

Table 1. Maximal Model Construction

- $\varepsilon(\mathrm{tt}) \triangleq ((\{s_{\mathrm{tt}}\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \emptyset,\ \{s_{\mathrm{tt}} \mapsto 0\}),\ \{s_{\mathrm{tt}}\},\ \{s_{\mathrm{tt}} \mapsto \emptyset\})$
  where $s_{\mathrm{tt}} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\}$ for all $a \in A$.
- $\varepsilon(\mathrm{ff}) \triangleq ((\emptyset,\ A,\ \emptyset,\ \emptyset,\ \emptyset),\ \emptyset,\ \emptyset)$
- $\varepsilon(Z) \triangleq ((\{s_{new}, s_{\mathrm{tt}}\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \emptyset,\ \{s_{\mathrm{tt}} \mapsto 0, s_{new} \mapsto 0\}),\ \{s_{new}\},\ \{s_{new} \mapsto \{Z\}, s_{\mathrm{tt}} \mapsto \emptyset\})$
  where $s_{new} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\}$ and $s_{\mathrm{tt}} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\}$ for all $a \in A$.
- $\varepsilon(\Phi_1 \wedge \Phi_2) \triangleq (((S_{\mathcal{E}_1} \times S_{\mathcal{E}_2})|_{\Box},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ W),\ (S_{\mathcal{E}_1} \times S_{\mathcal{E}_2})|_{\Box} \cap (S_1 \times S_2),\ \lambda)$ where
  - $\longrightarrow^{\Diamond}_{\mathcal{E}} \triangleq \{(s,r) \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S' \times \cup\partial^{\Diamond}_a(r) \mid s \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S'\}$
    $\cup\ \{(s,r) \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \cup\partial^{\Diamond}_a(s) \times R' \mid r \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_2} R'\}$
    $\cup\ \{(s,r) \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S' \times R' \mid s \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S' \wedge r \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_2} R' \wedge S' \notin \partial^{\Box}_a(s) \wedge R' \notin \partial^{\Box}_a(r)\}$
  - $\longrightarrow^{\Box}_{\mathcal{E}} \triangleq \{(s,r) \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} (S' \times \cup\partial^{\Diamond}_a(r)) \mid s \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S'\}$
    $\cup\ \{(s,r) \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} (\cup\partial^{\Diamond}_a(s) \times R') \mid r \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_2} R'\}$
  - $c \triangleq \{(s,r) \mapsto c_1(s) \cdot c_2(r) \mid s \in S_{\mathcal{E}_1} \wedge r \in S_{\mathcal{E}_2}\}$
  - $\lambda \triangleq \{(s,r) \mapsto \lambda_1(s) \cup \lambda_2(r) \mid s \in S_{\mathcal{E}_1} \wedge r \in S_{\mathcal{E}_2}\}$
- $\varepsilon(\Phi_1 \vee \Phi_2) \triangleq ((S_{\mathcal{E}_1} \cup S_{\mathcal{E}_2},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c),\ S_1 \cup S_2,\ \lambda_1 \cup \lambda_2)$ with:
  - $\longrightarrow^{\Diamond}_{\mathcal{E}} \triangleq \longrightarrow^{\Diamond}_{\mathcal{E}_1} \cup \longrightarrow^{\Diamond}_{\mathcal{E}_2}$
  - $\longrightarrow^{\Box}_{\mathcal{E}} \triangleq \longrightarrow^{\Box}_{\mathcal{E}_1} \cup \longrightarrow^{\Box}_{\mathcal{E}_2}$
  - $c \triangleq \{s \mapsto c_1(s) \cdot 0^{k_2} \mid s \in S_{\mathcal{E}_1}\} \cup \{s \mapsto 0^{k_1} \cdot c_2(s) \mid s \in S_{\mathcal{E}_2}\}$
- $\varepsilon([a]\,\Phi_1) \triangleq ((S_{\mathcal{E}_1} \cup \{s_{new}, s_{\mathrm{tt}}\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}_1},\ c),\ \{s_{new}\},\ \lambda)$ with:
  - $\longrightarrow^{\Diamond}_{\mathcal{E}} \triangleq \longrightarrow^{\Diamond}_{\mathcal{E}_1} \cup \{s_{\mathrm{tt}} \overset{a'}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\} \mid a' \in A\} \cup \{s_{new} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S_1\}$
    $\cup\ \{s_{new} \overset{a'}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\} \mid a' \neq a \wedge a' \in A\}$
  - $c \triangleq c_1 \cup \{s_{new} \mapsto 0^{k_1}\} \cup \{s_{\mathrm{tt}} \mapsto 0^{k_1}\}$
  - $\lambda \triangleq \lambda_1 \cup \{s_{new} \mapsto \emptyset\} \cup \{s_{\mathrm{tt}} \mapsto \emptyset\}$
- $\varepsilon(\langle a \rangle\,\Phi_1) \triangleq \varepsilon(\mathrm{ff})$ if $S_1 = \emptyset$
  $\varepsilon(\langle a \rangle\,\Phi_1) \triangleq ((S_{\mathcal{E}_1} \cup \{s_{new}, s_{\mathrm{tt}}\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c),\ \{s_{new}\},\ \lambda)$ otherwise, with:
  - $\longrightarrow^{\Diamond}_{\mathcal{E}} \triangleq \longrightarrow^{\Diamond}_{\mathcal{E}_1} \cup \{s_{new} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S_1\} \cup \{s_{new} \overset{a'}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\} \mid a' \in A\} \cup \{s_{\mathrm{tt}} \overset{a'}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s_{\mathrm{tt}}\} \mid a' \in A\}$
  - $\longrightarrow^{\Box}_{\mathcal{E}} \triangleq \longrightarrow^{\Box}_{\mathcal{E}_1} \cup \{s_{new} \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S_1\}$
  - $c \triangleq c_1 \cup \{s_{new} \mapsto 0^{k_1}\} \cup \{s_{\mathrm{tt}} \mapsto 0^{k_1}\}$
  - $\lambda \triangleq \lambda_1 \cup \{s_{new} \mapsto \emptyset\} \cup \{s_{\mathrm{tt}} \mapsto \emptyset\}$
- $\varepsilon(\sigma Z.\Phi_1)\ ((2^{S_{\mathcal{E}_1}}|_{\Box}, A, \longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}}, c_\sigma), 2^{S_{\mathcal{E}_1}}|_{\Box} \cap \{\{s\} \mid s \in S_1\}, \lambda)$ where $\sigma \in \{\nu, \mu\}$ with:
  - $\longrightarrow^{\Diamond}_{\mathcal{E}} \triangleq \{\{s_1, \ldots, s_n\} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S \mid \exists i.\exists S'_i.s_i \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S'_i \wedge S = \partial_{\mathcal{P}}((\cup\partial^{\Diamond}_a(s_1), \ldots, S'_i, \ldots, \cup\partial^{\Diamond}_a(s_n)), S_1, \lambda_1, Z)\}$
    $\cup\ \{\{s_1, \ldots, s_n\} \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S \mid \forall j.\exists S'_j.s_j \overset{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S'_j \wedge S'_j \notin \partial^{\Box}_a(s_j) \wedge S = \partial_{\mathcal{P}}((S'_1, \ldots, S'_n), S_1, \lambda_1, Z)\}$
  - $\longrightarrow^{\Box}_{\mathcal{E}} \triangleq \{\ \{s_1, \ldots, s_n\} \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S \mid \exists i.\exists S'_i.s_i \overset{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S'_i \wedge S = \partial_{\mathcal{P}}((\cup\partial^{\Diamond}_a(s_1), \ldots, S'_i, \ldots, \cup\partial^{\Diamond}_a(s_n)), S_1, \lambda_1, Z)\}$
  - $c_\nu(\{s_1, \ldots, s_n\})(j) \triangleq \begin{cases} \underset{1 \le i \le n}{maxodd}(c_1(s_i)(j)) & \text{if } \forall i.Z \notin \lambda_1(s_i) \\ \underset{s \in S_{\mathcal{E}_1}}{\overset{even}{\prod}} c_1(s)(j) & \text{if } \exists i.Z \in \lambda_1(s_i) \end{cases}$
  - $c_\mu(\{s_1, \ldots, s_n\})(j) \triangleq \begin{cases} \underset{1 \le i \le n}{maxodd}(c_1(s_i)(j)) & \text{if } \forall i.Z \notin \lambda_1(s_i) \\ \underset{s \in S_{\mathcal{E}_1}}{\overset{odd}{\prod}} c_1(s)(j) & \text{if } \exists i.Z \in \lambda_1(s_i) \end{cases}$
  - $\lambda \triangleq \{\{s_1, \ldots, s_n\} \mapsto \bigcup_{1 \le i \le n} \lambda_1(s_i) - \{Z\} \mid \{s_1, \ldots, s_n\} \in 2^{S_{\mathcal{E}_1}}\}$

$\varepsilon(\Phi)$ and its start states are singletons containing some start state of $\varepsilon(\Phi)$. An invariant of the maximal model construction is that start states do not have incoming transitions. (The case for $\varepsilon(\mathrm{tt})$ is the only exception and can be easily adapted to satisfy the invariant.) For a transition of state $q = \{s_1, \ldots, s_n\}$ of $\varepsilon(\sigma Z.\Phi)$, each state s_i has a transition in $\varepsilon(\Phi)$. A member state of the target of this transition, then, contains a derivative for each s_i. A member of the target state additionally contains an initial state of $\varepsilon(\Phi)$ if one of the derivatives included is labeled by Z. The definition of Table 1 makes use of the target set function $\partial_{\mathcal{P}}$ defined below.

Definition 6 (Target Set Function $\partial_{\mathcal{P}}$). *Let Φ be a modal μ-calculus formula, σ be either μ or ν, $\varepsilon(\Phi)$ be $(\mathcal{E}_1,S,\lambda)$ where $\mathcal{E}_1 = (S_{\mathcal{E}_1}, A, \longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}}, c)$ is an EMTS, $S \subseteq S_{\mathcal{E}_1}$ is a set of start states, $\lambda : S_{\mathcal{E}_1} \to 2^{PropVar}$ is a function that maps states of $\mathcal{E}$ to propositional variables, $c : S_{\mathcal{E}} \to N^k$ is a coloring function that maps states of $\mathcal{E}$ to k-tuples, and let $Z \in PropVar$ be a propositional variable. Given a tuple consisting of a target set for each element of a state of $\varepsilon(\sigma Z.\Phi)$, the function $\partial_{\mathcal{P}} : (2^{S_{\mathcal{E}_1}} \times \ldots \times 2^{S_{\mathcal{E}_1}}) \times 2^{S_{\mathcal{E}_1}} \times (S_{\mathcal{E}_1} \to 2^{PropVar}) \times PropVar \to 2^{2^{S_{\mathcal{E}_1}}}$ defines the target set of a transition of $\varepsilon(\sigma Z.\Phi)$ for this state as follows:*

$$\partial_{\mathcal{P}}((S_1, \ldots, S_n), S, \lambda, Z) \triangleq \{\{s_1, \ldots, s_n\} \mid \forall i.s_i \in S_i \wedge \nexists j.Z \in \lambda(s_j)\} \cup \{\{s_1, \ldots, s_n, s_0\} \mid \forall i.s_i \in S_i \wedge \exists j.Z \in \lambda(s_j) \wedge s_0 \in S\}$$

Each component of the color of state q is determined by comparing the corresponding entries of the member states s_i. When, for at least one s_i, this entry is odd, the greatest of the corresponding odd entries is selected as the entry of q, otherwise the maximum entry is selected for the same purpose. In Table 1, the function *maxodd* selects the greater of two numbers if both of them are odd or both of them are even, and the odd one otherwise. The color of q is further updated if it contains a state s_i labeled by Z. When Z identifies a greatest fixed point formula, each entry of the constructed tuple is defined to be the least even upper bound of the integers used in this entry of $\varepsilon(\Phi)$. Whereas, when Z identifies a least fixed point formula, the least odd upper bound of the integers is the entry for the color of q. In Table 1, least even and least odd upper bounds are denoted by the operators $\overset{even}{\sqcap}$ and $\overset{odd}{\sqcap}$, respectively.

4.2 Composing EMTSs

We extend the function ε to the domain of OTAs so that $\varepsilon(\Gamma \rhd E) = (\mathcal{E},S,\lambda)$, where $\mathcal{E} = (S_{\mathcal{E}}, A, \longrightarrow^{\Diamond}_{\mathcal{E}}, \longrightarrow^{\Box}_{\mathcal{E}}, c)$ is an EMTS, $S \subseteq S_{\mathcal{E}}$ is the set of *start states* of $\mathcal{E}$, and $\lambda : S_{\mathcal{E}} \to 2^{RecProcVar}$ is a *labeling function.*

The function ε is defined inductively on the structure of E as shown in Table 2. In the definition, we let $\varepsilon(\Gamma \rhd E_1)$ be $((S_{\mathcal{E}_1}, A, \longrightarrow^{\Diamond}_{\mathcal{E}_1}, \longrightarrow^{\Box}_{\mathcal{E}_1}, c_1), S_1, \lambda_1)$ and $\varepsilon(\Gamma \rhd E_2)$ be $((S_{\mathcal{E}_2}, A, \longrightarrow^{\Diamond}_{\mathcal{E}_2}, \longrightarrow^{\Box}_{\mathcal{E}_2}, c_2), S_2, \lambda_2)$, where $S_{\mathcal{E}_1}$ and $S_{\mathcal{E}_2}$ are disjoint sets. The new state s_{new} is not in $S_{\mathcal{E}_1}$. The coloring functions $c_1 : S_{\mathcal{E}_1} \to \mathbb{N}^{k_1}$ and $c_2 : S_{\mathcal{E}_2} \to \mathbb{N}^{k_2}$ color the states of $\mathcal{E}_1$ and $\mathcal{E}_2$ with integer tuples of length k_1 and k_2 respectively.

Table 2. EMTS Construction for Process Algebra Terms

- $\varepsilon(\Gamma \rhd \mathbf{0}) \stackrel{\Delta}{=} ((\{s_{new}\},\ A,\ \emptyset,\ \emptyset,\ \{s_{new} \mapsto 0\}),\ \{s_{new}\},\ \{s_{new} \mapsto \emptyset\})$
- $\varepsilon(\Gamma \rhd X) \stackrel{\Delta}{=} \varepsilon(\Phi)$ if $X \in AssProcVar$
  where $\Phi = \bigwedge_{X:\Psi \in \Gamma} \Psi$ (defaults to tt when Γ contains no assumption on X).
- $\varepsilon(\Gamma \rhd X) \stackrel{\Delta}{=} ((\{s_{new}\},\ A,\ \emptyset,\ \emptyset,\ \{s_{new} \mapsto 0\}),\ \{s_{new}\},\ \{s_{new} \mapsto \{X\}\})$ if $X \in RecProcVar$
- $\varepsilon(\Gamma \rhd a.E_1) \stackrel{\Delta}{=} ((S_{\mathcal{E}_1} \cup \{s_{new}\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c),\ \{s_{new}\},\ \lambda_1 \cup \{s_{new} \mapsto \emptyset\})$ with:
  $\longrightarrow^{\Diamond}_{\mathcal{E}} \stackrel{\Delta}{=} \longrightarrow^{\Diamond}_{\mathcal{E}_1} \cup \{s_{new} \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S_1\}$
  $\longrightarrow^{\Box}_{\mathcal{E}} \stackrel{\Delta}{=} \longrightarrow^{\Box}_{\mathcal{E}_1} \cup \{s_{new} \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S_1\}$
  $c \stackrel{\Delta}{=} c_1 \cup \{s_{new} \mapsto 0^{k_1}\}$
- $\varepsilon(\Gamma \rhd E_1 + E_2) \stackrel{\Delta}{=} ((S_{\mathcal{E}_1} \cup S_{\mathcal{E}_2} \cup (S_1 \times S_2),\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c),\ S_1 \times S_2,\ \lambda)$
  $\longrightarrow^{\Diamond}_{\mathcal{E}} \stackrel{\Delta}{=} \longrightarrow^{\Diamond}_{\mathcal{E}_1} \cup \longrightarrow^{\Diamond}_{\mathcal{E}_2} \cup \{(s,r) \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S' \mid s \in S_1 \wedge r \in S_2 \wedge (s \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S' \vee r \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_2} S')\}$
  $\longrightarrow^{\Box}_{\mathcal{E}} \stackrel{\Delta}{=} \longrightarrow^{\Box}_{\mathcal{E}_1} \cup \longrightarrow^{\Box}_{\mathcal{E}_2} \cup \{(s,r) \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S' \mid s \in S_1 \wedge r \in S_2 \wedge (s \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S' \vee r \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_2} S')\}$
  $c \stackrel{\Delta}{=} \{s \mapsto c_1(s) \cdot 0^{k_2} \mid s \in S_{\mathcal{E}_1}\} \cup \{r \mapsto 0^{k_1} \cdot c_2(r) \mid r \in S_{\mathcal{E}_2}\}$
  $\cup\ \{(s,r) \mapsto c_1(s) \cdot c_2(r) \mid (s,r) \in S_1 \times S_2\}$
  $\lambda \stackrel{\Delta}{=} \lambda_1 \cup \lambda_2 \cup \{(s,r) \mapsto \lambda_1(s) \cup \lambda_2(r) \mid s \in S_1 \wedge r \in S_2\}$
- $\varepsilon(\Gamma \rhd \mathit{fix}X.E_1) \stackrel{\Delta}{=} ((S_{\mathcal{E}_1},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c_1),\ S_1,\ \lambda)$ with:
  $\longrightarrow^{\Diamond}_{\mathcal{E}} \stackrel{\Delta}{=} \{s \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S \mid (s \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S) \vee$
  $(\exists s_1 \in S_1 . s_1 \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S \wedge\ X \in \lambda_1(s) \wedge s \text{ is reachable from } s_1)\}$
  $\longrightarrow^{\Box}_{\mathcal{E}} \stackrel{\Delta}{=} \{s \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S \mid (s \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S) \vee$
  $(\exists s_1 \in S_1 . s_1 \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S \wedge\ X \in \lambda_1(s) \wedge s \text{ is reachable from } s_1)\}$
  $\lambda \stackrel{\Delta}{=} \{s \mapsto (\lambda_1(s) - \{X\}) \mid s \in S_{\mathcal{E}_1}\}$
- $\varepsilon(\Gamma \rhd E_1 \parallel E_2) \stackrel{\Delta}{=} ((S_{\mathcal{E}_1} \times S_{\mathcal{E}_2} \times \{1,2\},\ A,\ \longrightarrow^{\Diamond}_{\mathcal{E}},\ \longrightarrow^{\Box}_{\mathcal{E}},\ c),\ S_1 \times S_2 \times \{1,2\},\ \lambda)$
  $\longrightarrow^{\Diamond}_{\mathcal{E}} \stackrel{\Delta}{=} \{(s,r,x) \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} S' \times \{r\} \times \{1\} \mid s \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_1} S'\}$
  $\cup\ \{(s,r,x) \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}} \{s\} \times R' \times \{2\} \mid r \stackrel{a}{\longrightarrow}^{\Diamond}_{\mathcal{E}_2} R'\}$
  $\longrightarrow^{\Box}_{\mathcal{E}} \stackrel{\Delta}{=} \{(s,r,x) \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}} S' \times \{r\} \times \{1\} \mid s \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_1} S'\}$
  $\cup\ \{(s,r,x) \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}} \{s\} \times R' \times \{2\} \mid r \stackrel{a}{\longrightarrow}^{\Box}_{\mathcal{E}_2} R'\}$
  $c \stackrel{\Delta}{=} \{(s,r,1) \mapsto c_1(s) \cdot 0^{k_2} \mid s \in S_{\mathcal{E}_1} \wedge r \in S_{\mathcal{E}_2}\}$
  $\cup\ \{(s,r,2) \mapsto 0^{k_1} \cdot c_2(r) \mid s \in S_{\mathcal{E}_1} \wedge r \in S_{\mathcal{E}_2}\}$
  $\lambda \stackrel{\Delta}{=} \{(s,r,x) \mapsto \emptyset \mid s \in S_{\mathcal{E}_1} \wedge r \in S_{\mathcal{E}_2} \wedge x \in \{1,2\}\}$

The EMTS corresponding to the nil process **0** consists of an abstract state without outgoing transitions, indicating that no transition is allowed for processes simulated by this state. If a process variable X in the term E stands for an underspecified component of the system, that is if X is an assumption process variable, then the EMTS for X is a maximal model for the conjunction of the properties specified for this component in the assumption list Γ.

The EMTS for a recursion process variable X is a single state without outgoing transitions, since the capabilities of the processes simulated are determined by the binding *fix*-expression. The function λ labels the state X. Given the EMTS

for the term of the *fix*-expression where X is free, the transitions of the start states are transferred to the states labeled by X.

The EMTS for a subterm prefixed by an action a is given by a start state with a must a-transition to the set of start states of the EMTS for the subterm. The EMTS for the sum operator consists of an EMTS where the start states are the cross product of the start states of the EMTSs for the subterms. It is assumed for this case that there are no incoming transitions to the start states of the EMTSs being combined. This is an invariant of the construction, except the case for tt which can be trivially converted to an equivalent EMTS to satisfy the property.

Finally, the states of the EMTS for a parallel composition of two components consist of a state from each component. Each state has transitions such that one of the components make a transition while the other stays in the same state. Each state is further marked by 1 or 2 to keep track of which component has performed the last transition; this is necessary to enable a run of the composition if the interleaved runs are enabled.

4.3 Correctness Results

The aim of the above construction is to capture, by means of an EMTS, exactly those behaviors denoted by the given OTA. The construction is *sound* (resp. *complete*) if the denotation of the OTA is a subset (resp. superset) of the denotation of the resulting EMTS. Our first result establishes that the first part of the construction is a maximal model construction for the modal μ-calculus.

Theorem 1. *Let $\mathcal{T}$ be a transition-closed LTS, Φ be a closed and guarded modal μ-calculus formula and $\varepsilon(\Phi) = (\mathcal{E}, S, \lambda)$. Then $[\![S]\!]_{\mathcal{T}} = ||\Phi||^{\mathcal{T}}$.*

Our next result shows that the construction is sound and complete when assumptions exist on only one of the components that are running in parallel and the rest of the system is fully determined.

Theorem 2. *Let $\mathcal{T}$ be a transition-closed LTS, $\Gamma \rhd E \| t$ be a guarded linear OTA where E does not contain parallel composition and t is closed, and let $\varepsilon(\Gamma \rhd E \| t) = (\mathcal{E}, S, \lambda)$. Then $[\![S]\!]_{\mathcal{T}}$ is equal to the set $[\![\Gamma \rhd E \| t]\!]_{\rho_0}$ up to bisimulation, where ρ_0 maps each recursion process variable X to $\mathbf{0}$.*

Theorems 1 and 2 are proved by induction on the structure of the logical formula and the process term, respectively. The proofs can be found in [19].

In the general case, when multiple underspecified components run in parallel, we only have soundness: our construction is sound for systems without dynamic process creation. For systems with dynamic process creation, the construction does not terminate.

Theorem 3. *Let $\mathcal{T}$ be a transition-closed LTS, $\Gamma \rhd E$ be a guarded linear OTA where every recursion process variable in the scope of parallel composition is bound by a* fix *operator in the same scope, and let $\varepsilon(\Gamma \rhd E) = (\mathcal{E}, S, \lambda)$. Then the set $[\![S]\!]_{\mathcal{T}}$ includes $[\![\Gamma \rhd E]\!]_{\rho_0}$ up to bisimulation.*

The proof of the theorem is as the proof of Theorem 2, but includes a more general case for parallel composition and can be found in [19].

Our last result reflects the fact that verification of open systems in the presence of parallel composition is undecidable for the modal μ-calculus in general. Completeness results can, however, be obtained for various fragments of the μ-calculus, such as ACTL, ACTL* and the simulation logic of [9]. In our approach, the tasks of constructing a finite representation of the state space in the form of an EMTS and the task of verifying properties of this representation are separated. This allows different logics to be employed for expressing assumptions on components and for specifying system properties, giving rise to more refined completeness results.

5 A Proof System for EMTS

In [3], we presented a proof system for verifying that an abstract state s of an EMTS $\mathcal{E}$ satisfies a modal μ-calculus formula Φ. In this section, we give a summary of this proof system and provide an alternative termination condition that uses the coloring function c instead of the earlier condition that assumed an extensional definition of the set of prohibited runs $W_{\mathcal{E}}$. The system is a specialization of a proof system by Bradfield and Stirling [20, 21] for showing μ-calculus properties for sets of LTS states. The relationship between the two proof systems is clear when one considers that each EMTS state denotes a set of LTS states.

A proof tree is constructed using the rules below, where σ ranges over μ and ν. The construction starts with the goal and progresses in a goal-directed fashion, checking at each step if a terminal node was reached.

$$\frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \wedge \Psi}{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \quad s \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi} \qquad \frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \vee \Psi}{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi} \qquad \frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \vee \Psi}{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi}$$

$$\frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} \sigma Z.\Phi}{s \vdash^{\mathcal{E}}_{\mathcal{V}} Z} \qquad \frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} [a]\Phi}{s_1 \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \ \ldots \ s_n \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi} \{s_1, \ldots, s_n\} = \cup\, \partial^{\Diamond}_a(s)$$

$$\frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} Z}{s \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi} Z \text{ identifies } \sigma Z.\Phi \qquad \frac{s \vdash^{\mathcal{E}}_{\mathcal{V}} \langle a\rangle \Phi}{s_1 \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi \ \ldots \ s_n \vdash^{\mathcal{E}}_{\mathcal{V}} \Phi} \{s_1, \ldots, s_n\} \in \partial^{\Box}_a(s)$$

A successful tableau (or proof) is a finite proof tree having successful terminals as leaves. If $n : r \vdash^{\mathcal{E}}_{\mathcal{V}} Z$ is a node where Z identifies a fixed point formula, and there is an identical ancestor node of n, $n' : r \vdash^{\mathcal{E}}_{\mathcal{V}} Z$ and for any other fixed point variable Y on this path, Z subsumes Y, then node n is called a σ-terminal. So no further rules are applied to it. The most recent node making n a σ-terminal is named n's companion. The conditions for a leaf node $r \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi$ of a proof tree to be a successful terminal are listed below.

Successful Terminals

1. $\Psi = \mathrm{tt}$, or else $\Psi = Z$, Z is free in the initial formula, and $r \in \mathcal{V}(Z)$
2. $\Psi = [a]\Phi$ and $\cup \partial^{\Diamond}_a(r) = \emptyset$

3. $\Psi = Z$ where Z identifies a fixed point formula $\sigma Z.\Phi$, and the sequent is a σ-terminal with companion node $n : r \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi$, then
   (a) If $\sigma = \nu$, then the terminal is successful.
   (b) If $\sigma = \mu$, then the terminal is successful if every infinite run of the EMTS that corresponds to an infinite sequence of trails of the companion node n_0 is in $W_{\mathcal{E}}$. (The notion of trail is explained below.) When the set $W_{\mathcal{E}}$ is encoded using a coloring function c, the condition is that for any set S_T of trails of n_0, there should exist $1 \leq j \leq k$, so that $max(\bigcup_{T \in S_T} c(\alpha(T))(j))$ is odd. This ensures, for an infinite run $w_{n_0} = \alpha(T_1) \circ \alpha(T_2) \circ \alpha(T_3) \ldots$ where for all $i \geq 1$, T_i is a trail of n_0, that there exists some $1 \leq j' \leq k$ such that $max\,(inf(c(w_{n_0})(j')))$ is odd.

Unsuccessful Terminals

1. $\Psi = \mathrm{ff}$, or else $\Psi = Z$, Z is free in the initial formula, and $r \notin \mathcal{V}(Z)$
2. $\Psi = \langle a \rangle \Phi$ and $\cup \partial^{\Box}_a(r) = \emptyset$
3. $\Psi = Z$ where Z identifies the least fixed point formula $\mu Z.\Phi$, and the sequent is a σ-terminal with companion node n_0, then the terminal is unsuccessful if there exists a set S_T of trails of n_0 such that for every $1 \leq j \leq k$, $max\,(\bigcup_{T \in S_T} c(\alpha(T))(j))$ is even. This means that some infinite run w_{n_0} of the EMTS, which corresponds to an infinite sequence of trails of the companion node n_0, is not in $W_{\mathcal{E}}$.

Trails and corresponding runs are defined as follows. Assume that node $n_k{:}r \vdash^{\mathcal{E}}_{\mathcal{V}} Z$ is a μ-terminal and node $n_0{:}r \vdash^{\mathcal{E}}_{\mathcal{V}} Z$ is its companion. A trail T of the companion node n_0 is a sequence of state–node pairs $(r, n_0), \ldots, (r, n_k)$ such that for all $0 \leq i < k$, one of the following holds:

1. $n_{i+1} : r_{i+1} \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi_{i+1}$ is an immediate successor of $n_i : r_i \vdash^{\mathcal{E}}_{\mathcal{V}} \Psi_i$, or
2. n_i is the immediate predecessor of a σ-terminal node $n' : r' \vdash^{\mathcal{E}}_{\mathcal{V}} Z'$ where $n' \neq n_k$ whose companion is $n_j : r' \vdash^{\mathcal{E}}_{\mathcal{V}} Z'$ for some $j : 0 \leq j \leq i$, $n_{i+1} = n_j$, and $r_{i+1} = r'$.

In order to convert a trail to a corresponding run, we use the function α, which returns the empty string when the trail contains only one pair, and is defined for longer trails as follows:

$$\alpha((r_1, n_1) \cdot (r_2, n_2) \cdot T) \triangleq \begin{cases} (r_1 \xrightarrow{a}_{\mathcal{E}} r_2) \cdot \alpha((r_2, n_2) \cdot T) & \Box_a \text{ or } \Diamond_a\text{-rule is applied to } n_1 \\ \alpha((r_2, n_2) \cdot T) & \text{otherwise.} \end{cases}$$

A formula is prime if whenever it logically implies a disjunction then it also implies one of the disjuncts. As we show in [3], the proof system is sound and complete for all formulas with only prime subformulas. An example proof is given in [19].

6 Conclusion

In this paper we investigate a state space representation for open systems specified as open process terms with behavioural assumptions written in the modal μ-calculus. This representation can serve both as a graphical specification formalism and as a basis for verification, supporting state space exploration based techniques and state visualization for interactive methods. We present a two-phase construction of such a representation from an open term with assumptions, and show it sound for terms without dynamic process creation and complete for systems with a single underspecified component. Finally, we adapt an existing proof system for the task of proving behavioural properties of open systems based on the given state space representation. The relative simplicity of the proof system and its use is an indication of the adequateness of EMTSs for open system state space representation.

Future work is required to characterize more precisely the construction and the μ-calculus fragments for which it is complete, taking into account that the fragment for specifying component assumptions need not be the same as the fragment chosen for specifying system properties. In addition to automatic state space construction, interactive state space exploration will be considered, allowing a wider class of open systems to be handled. Finally, we plan to demonstrate the utility of the proposed approach by means of tool support and case studies.

References

1. Martinelli, F.: Analysis of security protocols as open systems. Theoretical Computer Science **290** (2003) 1057–1106
2. Larsen, K.G.: Modal specifications. Automatic Verification Methods for Finite State Systems (1989) 232–246
3. Aktug, I., Gurov, D.: Verification of open systems based on explicit state space representation. In: AVIS'05: Proceedings of Automated Verification of Infinite Systems. To appear (2005)
4. Kozen, D.: Results on the propositional mu-calculus. Theoretical Computer Science **27** (1983) 333–354
5. Dam, M.: Fixed points of Büchi automata. In: FSTTCS '92: Proceedings of 12th Conference on Foundations of Software Technology and Theoretical Computer Science. Volume 652 of Lecture Notes in Computer Science. (1992) 39–50
6. Kaivola, R.: On modal mu-calculus and Büchi tree automata. Information Processing Letters **54** (1995) 17–22
7. Grumberg, O., Long, D.: Model checking and modular verification. ACM Transactions on Programming Languages and Systems **16(3)** (1994) 843–871
8. Kupferman, O., Vardi, M.: An automata-theoretic approach to modular model checking. ACM Transactions on Programming Languages and Systems **22** (2000) 87–128
9. Sprenger, C., Gurov, D., Huisman, M.: Compositional verification for secure loading of smart card applets. In Heitmeyer, C., Talpin, J.P., eds.: Proc. MEMOCODE'04, IEEE (2004) 211–222

10. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: Proceedings of 32nd Annual Symposium on Foundations of Computer Science. IEEE, Computer Society Press (1991) 368–377
11. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. Journal of ACM **47** (2000) 312–360
12. Dam, M., Fredlund, L., Gurov, D.: Toward parametric verification of open distributed systems. In Langmaack, H., Pnueli, A., de Roever, W.P., eds.: Compositionality: the Significant Difference. Volume 1536 of Lecture Notes in Computer Science. Springer-Verlag (1998) 150–185
13. Dam, M., Gurov, D.: Compositional verification of CCS processes. In: PSI '99: Proceedings of the Third International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Springer-Verlag (2000) 247–256
14. Christensen, S.: Decidability and Decomposition in Process Algebras. PhD thesis, University of Edinburgh (1993)
15. Huth, M., Jagadeesan, R., Schmidt, D.A.: Modal transition systems: A foundation for three-valued program analysis. In: ESOP '01: Proceedings of the 10th European Symposium on Programming Languages and Systems. Volume 2028., London, UK, Springer-Verlag (2001) 155–169
16. Grumberg, O., Shoham, S.: Monotonic abstraction-refinement for CTL. In: TACAS'04: Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Volume 2988 of Lecture Notes in Computer Science., Springer-Verlag (2004) 546–560
17. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. Computation Theory **208** (1984) pages 157–168
18. Bustan, D., Grumberg, O.: Applicability of fair simulation. In: TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Volume 2280 of Lecture Notes in Computer Science., Springer-Verlag (2002) 401–414
19. Aktug, I., Gurov, D.: State space representation for verification of open systems. Technical report, KTH CSC, `http://www.nada.kth.se/~irem/sefros/techrep06.ps` (2006)
20. Bradfield, J., Stirling, C.: Local model checking for infinite state spaces. Theoretical Computer Science **96** (1992) 157–174
21. Stirling, C.: Modal and Temporal Properties of Processes. Texts in Computer Science. Springer-Verlag (2001)