FI MU

Faculty of Informatics
Masaryk University Brno

Modal Process Rewrite Systems

by

Nikola Beneš
Jan Křetínský

FI MU Report Series **FIMU-RS-2012-02**

 June 2012

Publications in the FI MU Report Series are in general accessible via WWW:

http://www.fi.muni.cz/reports/

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

Modal Process Rewrite Systems

Nikola Beneš*
Faculy of Informatics, Masaryk University, Brno, Czech Republic
xbenes3@fi.muni.cz

Jan Křetínský†
Faculty of Informatics, Masaryk University, Brno, Czech Republic
Institut für Informatik, Technische Universität München, Germany
jan.kretinsky@fi.muni.cz

June 27, 2012

Abstract

We consider modal transition systems with infinite state space generated by finite sets of rules. In particular, we extend process rewrite systems to the modal setting and investigate decidability of the modal refinement relation between systems from various subclasses. Since already simulation is undecidable for most of the cases, we focus on the case where either the refined or the refining process is finite. Namely, we show decidability for pushdown automata extending the non-modal case and surprising undecidability for basic parallel processes. Further, we prove decidability when both systems are visibly pushdown automata. For the decidable cases, we also provide complexities. Finally, we discuss a notion of bisimulation over MTS.

1 Introduction

The ever increasing complexity of software systems together with their reuse call for efficient *component-based* design and verification. One of the major theoretically well founded frameworks that answer this call are *modal transition systems* (MTS) [LT88].

*The author has been supported by the Czech Science Foundation, grant No. GAP202/11/0312.
†The author is a holder of Brno PhD Talent Financial Aid and is supported by the Czech Science Foundation, grant No. P202/12/G061.

Their success resides in natural combination of two features. Firstly, it is the simplicity of labelled transition systems, which have proved appropriate for behavioural description of systems as well as their compositions; MTS as their extension inherit this appropriateness. Secondly, as opposed to temporal logic specifications, MTS can be easily *gradually refined* into implementations while preserving the desired behavioural properties.

MTS consist of a set of states and two transition relations. The *must* transitions prescribe which behaviour has to be present in every refinement of the system; the *may* transitions describe the behaviour that is allowed, but need not be realized in the refinements. This allows for underspecification of non-critical behaviour in the early stage of design, focusing on the main properties, verifying them and sorting out the details of the yet unimplemented non-critical behaviour later.

The formalism of MTS has proven to be useful in practice. Industrial applications are as old as [Bru97] where MTS have been used for an air-traffic system at Heathrow airport. Besides, MTS are advocated as an appropriate base for interface theories in [RBB$^+$09] and for product line theories in [Nym08]. Further, MTS based software engineering methodology for design via merging partial descriptions of behaviour has been established in [UC04]. Moreover, the tool support is quite extensive, e.g. [BLS95, DFFU07, BML11, BČK11].

Over the years, many extensions of MTS have been proposed. While MTS can only specify whether or not a particular transition is required, some extensions equip MTS with more general abilities to describe what *combinations* of transitions are possible [LX90, FS08, BK10, BKL$^+$11]. Further, MTS framework has also been lifted to *quantitative settings*. This includes probabilistic [CDL$^+$10] and timed systems [ČGL93, JLS11, BFJ$^+$11, BKL$^+$12, DLL$^+$10, BLPR11] with clear applications in the embedded systems design. As far as the infinite state systems are concerned, only a few more or less ad hoc extensions have been proposed, such as systems with asynchronous communication based on FIFO [BHJ10] or Petri nets [EBHH10]. In this paper, we introduce modalities into a general framework for infinite-state systems, where we study modal extensions of well-established classes of infinite-state systems.

Such a convenient unifying framework for infinite-state systems is provided by *Process rewrite systems* (PRS) [May00]. They encompass many standard models such as pushdown automata (PDA) or Petri nets (PN) as syntactic subclasses. A PRS consists of a set of rewriting rules that model computation. These rules may contain sequential and parallel composition. For example, a transition $\mathtt{t}$ of a Petri net with input places $\mathtt{I}_1, \mathtt{I}_2$ and output places $\mathtt{O}_1, \mathtt{O}_2$ can be described by the rule $\mathtt{I}_1 \parallel \mathtt{I}_2 \xrightarrow{\mathtt{t}} \mathtt{O}_1 \parallel \mathtt{O}_2$. A transition of a pushdown automaton in a state $\mathtt{s}$ with a top stack symbol $\mathtt{X}$ reading a letter $\mathtt{a}$ resulting in changing the state to $\mathtt{q}$ and pushing $\mathtt{Y}$ to the stack can be written as $s\mathtt{X} \xrightarrow{\mathtt{a}} qYX$. Limiting the occurrences of parallel and sequential composition on the left and right sides of the rules yields the most common automata theoretic models. For these syntactic subclasses of PRS, see Figure 1 and a more detailed description in Section 2.

Motivation One can naturally lift PRS to the modal world by having two sets of rules, may and must rules. What is then the use of such *modal process rewrite systems* (mPRS)? Firstly, potentially infinite-state systems such as Petri nets are very popular for modelling whenever communication or synchronization between processes occurs. This is true even when they are actually bounded and thus with a finite state space.

Example 1.1. *Consider the following may rule (we use dashed arrows to denote may rules) generating a small Petri net.*

$$\mathsf{resource} \parallel \mathsf{customer} \overset{\mathsf{consume}}{\dashrightarrow} \mathsf{trash}$$

This rewrite rule implies that e.g. a process resource ∥ customer ∥ customer *may be changed into* trash ∥ customer. *Therefore, if there is no other rule with* trash *on the right side a safety property is guaranteed for all implementations of this system, namely that* trash *can only arise if there is at least one* resource *and one* customer. *On the other hand, it is not guaranteed that* trash *can indeed be produced in such a situation. This is very useful as during the design process new requirements can arise, such as necessity of adding more participants to perform this transition. For instance,*

$$\mathsf{resource} \parallel \mathsf{customer} \parallel \mathsf{permit} \overset{\mathsf{consume}}{\dashrightarrow} \mathsf{trash}$$

expresses an auxiliary condition required to produce trash, *namely that* permit *is available. Replacing the old rule with the new one is equivalent to adding an input place* permit *to the Petri net. In the modal transition system view, the new system* refines *the old one. Indeed, the new system is only more specific about the allowed behaviour than the old one and does not permit any previously forbidden behaviour. One can further refine the system by the one given by*

$$\mathsf{resource} \parallel \mathsf{customer} \parallel \mathsf{permit} \parallel \mathsf{bribe} \xrightarrow{\mathsf{consume}} \mathsf{trash}$$

where additional condition is imposed and now the trash*-producing transition has to be available (denoted by an unbroken arrow) whenever the left hand side condition is satisfied.*

Secondly, even if an original specification is finite its refinements and the final implementation might be infinite. For instance, consider a specification where permit needs to be available but is not consumed or there is an unlimited amount of permits. In an implementation, the number of permits could be limited and thus this number with no known bounds needs to be remembered in the state of the system. Similarly, consider a finite safety specification of a browser together with its implementation that due to the presence of back button requires the use of stack, and is thus a pushdown system. Further, sometimes both the specification and the implementation are infinite such as a stateless BPA specification of a stateful component implemented by a PDA.

Example 1.2. *Consider a basic process algebra (BPA) given by rules* $X \xrightarrow{(} XX$ *and* $X \xrightarrow{)} \varepsilon$ *for correctly parenthesized expressions with* $X \dashrightarrow^{a} X$ *for all other symbols* a*, i.e. with no restriction on the syntax of expressions. One can easily refine this system into a PDA that accepts correct arithmetic expressions by remembering in the state whether the last symbol read was an operand or an operator.*

Further, opposite to the design of correct software where an abstract verified MTS is transformed into a concrete implementation, one can consider checking correctness of software through abstracting a concrete implementation into a coarser system. The use of MTS as abstractions has been advocated e.g. in [GHJ01]. While usually overapproximations (or underapproximations) of systems are constructed and thus only purely universal (or existential) properties can be checked, [GHJ01] shows that using MTS one can check mixed formulae (arbitrarily combining universal and existential properties) and, moreover, at the same cost as checking universal properties using traditional conservative abstractions. This advantage has been investigated also in the context of systems equivalent or closely related to MTS [HJS01, DGG97, Nam03, DN04, CGLT09, GNRT10]. Although one is usually interested in generating finite abstractions of infinite systems, it might be interesting to consider situations where the abstract system is infinite. For instance, if one is interested in a property that is inherently non-regular such as correct parenthesizing in the previous example, the abstraction has to capture this feature. One could thus abstract the PDA from the previous example into the smaller BPA above and prove the property here using algorithms for BPA. Moreover, if one is interested in mixed properties the abstract system has to be modal. It would be useful to extend the verification algorithms for systems such as PDA to their modal versions along the lines of the generalized model checking approach [BG00, BČK11]. This is, however, beyond the scope of this paper.

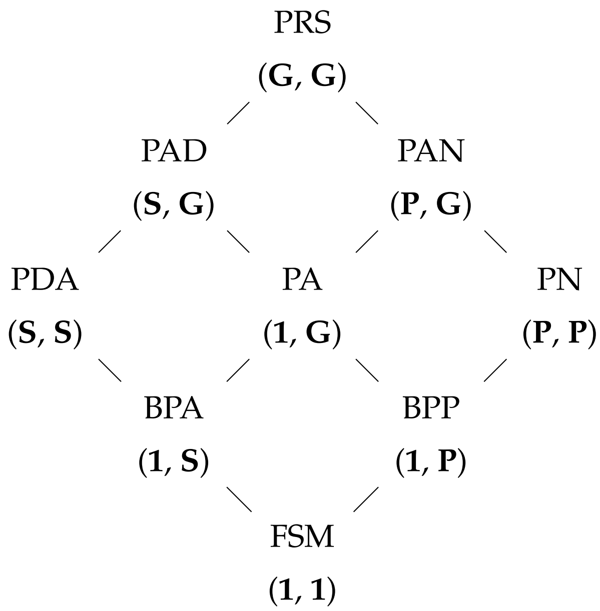

Figure 1: PRS hierarchy

Our contribution In this paper, we focus on modal infinite-state systems and decidability of the most fundamental problem here, namely deciding the refinement relation, for most common classes of systems. Since simulation is undecidable already on basic parallel processes (BPP) [Hüt94] and basic process algebras (BPA) [GH94], cf. Figure 1, the refinement as a generalization of simulation is undecidable in general. However, one can consider the case where either the refined or the refining system is finite (a finite state machine, FSM). This case is still very interesting, e.g. in the context of finite abstractions or implementations with bounded resources. [KM99] shows that while simulation remains undecidable between process algebras (PA) and FSM, it is decidable between PDA and FSM. We extend this result using methods of [KM02b] to the modal setting. Further, although simulation is decidable between PN and FSM [JM95] (in both directions), we show that surprisingly this result cannot be extended and the refinement is undecidable even for BPP and FSM in the modal setting. Although the decidability of the refinement seems quite limited now, we show that refinement is sometimes decidable even between two infinite-state systems, namely between modal extensions of visibly pushdown automata [AM04], cf. Example 1.2; for this, we use the methods of [Srb06]. To summarize:

- We introduce a general framework for studying modal infinite-state system, namely we lift process rewrite systems to the modal setting. This definition comes along with the appropriate notion of refinement.
- We prove un/decidability of the refinement problem for modal extensions of standard classes of infinite-state systems. Apart from trivial corollaries due to the undecidability of simulation, this amount to proving undecidability of refinement between Petri nets and FSM (on either side) and decidability between pushdown systems and FSM (again on either side). Moreover, we prove decidability for visibly PDA. For the decidable cases, we show that the complexity is the same as for checking the respective simulation in the non-modal setting. Finally, we discuss a notion of bisimulation over MTS, which we name birefinement.

Related work There are various other approaches to deal with component refinements. They range from subtyping [LW94] over Java modelling language [JP01] to interface theories close to MTS such as interface automata [dAH01]. Similarly to MTS, interface automata are behavioural interfaces for components. However, their composition works very differently. Furthermore, its notion of refinement is based on alternating simulation [AHKV98], which has been proved strictly less expressive than MTS refinement—actually coinciding on a subclass of MTS—in a paper [LNW07] that combines MTS and interface automata based on I/O automata [Lyn88]. The compositionality of this combination is further investigated in [RBB$^+$11].

MTS can also be viewed as a fragment of mu-calculus that is "graphically representable" [BL90]. The graphical representability of a variant of alternating simulation called covariant-contravariant simulation has been recently studied in [AFdFE$^+$11].

The PRS framework has been introduced in [May00]. Simulation on classes of PRS tends to be computationally harder than bisimilarity [KM02b]. While e.g. bisimulation between any PRS and FSM is decidable [KŘS05], simulation with FSM is undecidable already for PA (see above). Therefore, the decidability is limited to PDA and PN, and we show that refinement is even harder (undecidability for BPP). Another aspect that could help to extend the decidability is determinism. For instance, simulation between FSM and deterministic PA is decidable [KM99]. It is also the case with the abovementioned [EBHH10] where refinement over "weakly deterministic" modal Petri nets is shown decidable.

Outline of the paper In Section 2, we introduce modal process rewrite systems formally and recall the refinement preorder. In Section 3 and 4, we show undecidability and decidability results for the refinement. Section 5 concludes.

2 Refinement Problems

In this section, we introduce modal transition systems generated by process rewrite systems and define the notion of modal refinement. We start with the usual definition of MTS.

2.1 Modal Transition Systems

Definition 2.1 (Modal transition system). *A* modal transition system (MTS*) over an action alphabet Act is a triple* $(\mathcal{P}, \dashrightarrow, \longrightarrow)$*, where* $\mathcal{P}$ *is a set of* processes *and* $\longrightarrow \subseteq \dashrightarrow \subseteq \mathcal{P} \times Act \times \mathcal{P}$ *are* must *and* may *transition relations, respectively.*

Observe that $\mathcal{P}$ is not required to be finite. We often use letters $s, t, \ldots$ for processes of MTS. Whenever clear from the context, we refer to processes without explicitly mentioning their underlying MTS.

We proceed with the standard definition of (modal) refinement.

Definition 2.2 (Refinement). *Let* $(\mathcal{P}_1, \dashrightarrow_1, \longrightarrow_1), (\mathcal{P}_2, \dashrightarrow_2, \longrightarrow_2)$ *be MTS over the same action alphabet and* $s \in \mathcal{P}_1$*,* $t \in \mathcal{P}_2$ *be processes. We say that* s refines t*, written* $s \leq_m t$*, if there is a relation* $\mathcal{R} \subseteq \mathcal{P}_1 \times \mathcal{P}_2$ *such that* $(s, t) \in \mathcal{R}$ *and for every* $(p, q) \in \mathcal{R}$ *and every* $a \in Act$*:*

1. *if* $p \overset{a}{\dashrightarrow}_1 p'$ *then there is a transition* $q \overset{a}{\dashrightarrow}_2 q'$ *s.t.* $(p', q') \in \mathcal{R}$*, and*
2. *if* $q \xrightarrow{a}_2 q'$ *then there is a transition* $p \xrightarrow{a}_1 p'$ *s.t.* $(p', q') \in \mathcal{R}$*.*

The ultimate goal of the refinement process is to obtain an *implementation*, i.e. an MTS with $\dashrightarrow = \longrightarrow$. Implementations can be considered as the standard labelled transition systems (LTS). Note that on implementations refinement coincides with strong bisimilarity, and on modal transition systems without any must transitions it corresponds to the simulation preorder, denoted by $\leq_{sim}$. Further, refinement has a game characterization [BKLS09] similar to (bi)simulation games, which we often use in the proofs. For reader's convenience, we recall the refinement game in Appendix A.

2.2 Modal Process Rewrite Systems

We now move our attention to infinite-state MTS generated by finite sets of rules. Let *Const* be a set of *process constants*. We define the set of process expressions $\mathcal{E}$ by the following abstract syntax:

$$E ::= \varepsilon \mid X \mid E \parallel E \mid E; E$$

where X ranges over *Const*. We often use Greek letters $\alpha, \beta, \ldots$ for elements of $\mathcal{E}$. The process expressions are considered modulo the usual structural congruence, i.e. the smallest congruence such that the operator ; is associative, $\parallel$ is associative and commutative and ε is a unit for both ; and $\parallel$. We often omit the ; operator.

Definition 2.3 (Modal process rewrite system)**.** *A* process rewrite system (PRS) *is a finite relation* $\Delta \subseteq (\mathcal{E} \setminus \{\varepsilon\}) \times Act \times \mathcal{E}$*, elements of which are called rewrite rules. A* modal process rewrite system (mPRS) *is a tuple* $(\Delta_{\text{may}}, \Delta_{\text{must}})$ *where* $\Delta_{\text{may}}, \Delta_{\text{must}}$ *are process rewrite systems such that* $\Delta_{\text{must}} \subseteq \Delta_{\text{may}}$.

An mPRS $\Delta = (\Delta_{\text{may}}, \Delta_{\text{must}})$ *induces an MTS* $\mathbb{MTS}(\Delta) = (\mathcal{E}, \dashrightarrow, \longrightarrow)$ *as follows:*

$$\frac{(E, a, E') \in \Delta_{\text{may}}}{E \overset{a}{\dashrightarrow} E'} \qquad \frac{E \overset{a}{\dashrightarrow} E'}{E; F \overset{a}{\dashrightarrow} E'; F} \qquad \frac{E \overset{a}{\dashrightarrow} E'}{E \parallel F \overset{a}{\dashrightarrow} E' \parallel F}$$

$$\frac{(E, a, E') \in \Delta_{\text{must}}}{E \overset{a}{\longrightarrow} E'} \qquad \frac{E \overset{a}{\longrightarrow} E'}{E; F \overset{a}{\longrightarrow} E'; F} \qquad \frac{E \overset{a}{\longrightarrow} E'}{E \parallel F \overset{a}{\longrightarrow} E' \parallel F}$$

We consider four distinguished classes of process expressions. Class **S** stands for expressions with no $\parallel$ (purely sequential expressions) and class **P** stands for expressions with no ; (purely parallel expressions). Further, we use **G** for the whole $\mathcal{E}$ (general expressions) and **1** for *Const* (one process constant and no operators). Now restricting the left and right sides of rules of PRS to these classes yields subclasses of PRS as depicted in Figure 1 using the standard shortcuts also introduced in Section 1. Each subclass $\mathcal{C}$ has a corresponding modal extension $\mathrm{m}\mathcal{C}$ containing all mPRS $(\Delta_{\text{may}}, \Delta_{\text{must}})$ with both Δ_{may} and Δ_{must} in $\mathcal{C}$. For instance, mFSM correspond to the standard finite MTS and mPN are modal Petri nets as introduced in [EBHH10]. An example of an mBPP and the resulting MTS are depicted in Figure 2.

For any classes $\mathcal{C}, \mathcal{D}$, we define the following decision problem $\mathrm{m}\mathcal{C} \leq_{\mathrm{m}} \mathrm{m}\mathcal{D}$.

Given mPRS $\Delta_1 \in \mathrm{m}\mathcal{C}, \Delta_2 \in \mathrm{m}\mathcal{D}$ and process terms δ_1, δ_2 conforming to left-hand side restrictions of $\mathcal{C}, \mathcal{D}$, respectively, does $\delta_1 \leq_{\mathrm{m}} \delta_2$ hold considering δ_1, δ_2 as processes of $\mathbb{MTS}(\Delta_1), \mathbb{MTS}(\Delta_2)$?

$$\begin{aligned}\Delta_{\mathrm{may}} = \{ \quad & (X, a, X \parallel Y), \\ & (X, c, \varepsilon), \\ & (Y, b, \varepsilon) \} \\ \Delta_{\mathrm{must}} = \{ \quad & (X, a, X \parallel Y), \\ & (Y, b, \varepsilon) \}\end{aligned}$$

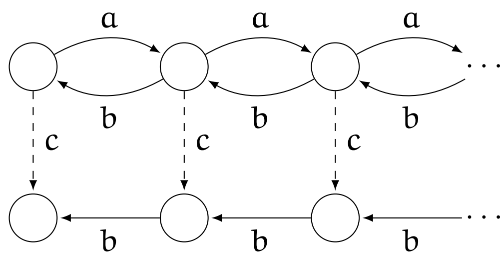

Figure 2: An example of a mBPP and its corresponding (infinite) MTS; the dashed arrows represent *may* transitions, the unbroken arrows represent *must* transitions; as $s \stackrel{a}{\longrightarrow} t$ implies $s \stackrel{a}{\dashrightarrow} t$ we omit the may transitions where must transitions are also present

3 Undecidability Results

In this section, we present all the negative results. As already discussed in Section 1, simulation—and thus refinement—is undecidable already on BPP [Hüt94] and BPA [GH94]. When considering the case where one of the two classes is mFSM, the undecidability holds for mPA [KM99]. Thus we are left with the problems mFSM$\leq_m$mPDA, mPDA$\leq_m$mFSM and mFSM$\leq_m$mPN, mPN$\leq_m$mFSM. On the one hand, the two former are shown decidable in Section 4 using non-modal methods for simulation of [KM02b]. On the other hand, the non-modal methods for simulation of [JM95] cannot be extended to the latter two problems. In this section, we show that (surprisingly) they are both undecidable and, moreover, even for mBPP.

Theorem 3.1. *The problem mBPP$\leq_m$mFSM is undecidable.*

Proof. We reduce the undecidable problem of simulation between two BPPs (even normed ones) to the problem mBPP$\leq_m$mFSM.

Let A, B be two BPP processes with underlying PRS Δ_A and Δ_B; w.l.o.g. $\Delta_A \cap \Delta_B = \emptyset$. We transform them as follows. We rename all actions of the underlying PRS of B from a to a'. Let Act' be the set of these renamed actions and let Δ'_B be the modification of Δ_B by renaming the actions. The mBPP is defined as $(\Delta_A \cup \Delta'_B, \Delta'_B)$, i.e. the transitions of A are just may, the (modified) transitions of B are both must and may.

We then build a finite mPRS as follows. The states are $\{s, u\} \cup \{s_a \mid a \in Act\}$.

- $s \stackrel{a}{\dashrightarrow} s_a$ and $s \stackrel{a'}{\dashrightarrow} u$ for all $a \in Act$
- $s_a \stackrel{a'}{\longrightarrow} s$ for all $a \in Act$ (with the corresponding may transition)

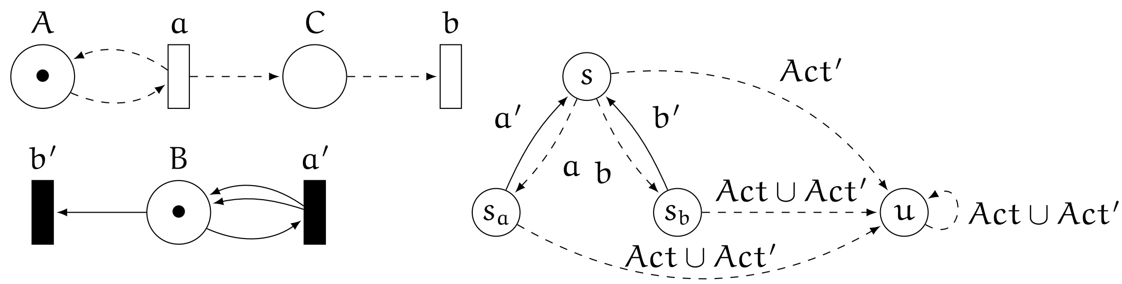

Figure 3: $A \parallel B \leq_m s$ where the original two BPPs are given by $A \xrightarrow{a} A \parallel C$, $C \xrightarrow{b} \varepsilon$, $B \xrightarrow{a} B \parallel B$, $B \xrightarrow{b} \varepsilon$.

- $s_a \overset{x}{\dashrightarrow} u$ for all $a \in Act$ and $x \in Act \cup Act'$
- $u \overset{x}{\dashrightarrow} u$ for all $x \in Act \cup Act'$

Clearly $q \leq_m u$ for any process q. The construction is illustrated in Figure 3.

We now show that $A \leq_{sim} B$ iff $A \parallel B \leq_m s$. In the following, α always denotes a process of Δ_A, while β denotes a process of Δ_B. Furthermore, we use the notation $\mathbb{LTS}(\Delta_A)$ to denote the LTS induced by Δ_A (similarly for Δ_B). We use the refinement game argumentation, see Appendix A.

$\Rightarrow$: Let $\mathcal{R} = \{(\alpha \parallel \beta, s) \mid \alpha \leq_{sim} \beta\}$. We show that $\mathcal{R}$ can be extended to be a modal refinement relation. Let $(\alpha \parallel \beta, s) \in \mathcal{R}$:

- If the attacker plays $\alpha \parallel \beta \overset{a'}{\dashrightarrow} \alpha \parallel \beta'$ (where $a' \in Act'$), the defender can play $s \overset{a'}{\dashrightarrow} u$ and obviously wins.
- If the attacker plays $\alpha \parallel \beta \overset{a}{\dashrightarrow} \alpha' \parallel \beta$ (where $a \in Act$), the defender has to play $s \overset{a}{\dashrightarrow} s_a$. There are two possibilities then:
  - if the attacker plays $\alpha' \parallel \beta \overset{x}{\dashrightarrow}$, the defender can play $s_a \overset{x}{\dashrightarrow} u$ and obviously wins;
  - if the attacker plays $s_a \xrightarrow{a'} s$, the defender can play $\alpha' \parallel \beta \xrightarrow{a'} \alpha' \parallel \beta'$ where β' is a process such that $\beta \xrightarrow{a} \beta'$ in $\mathbb{LTS}(\Delta_B)$ and $\alpha' \leq_{sim} \beta'$. Such β' obviously exists due to $\alpha \leq_{sim} \beta$. Thus $(\alpha' \parallel \beta', s) \in \mathcal{R}$.

$\Leftarrow$: We show that $\mathcal{R} := \{(\alpha, \beta) \mid \alpha \parallel \beta \leq_m s\}$ is a simulation. Let $(\alpha, \beta) \in \mathcal{R}$:

- If $\alpha \xrightarrow{a} \alpha'$ in $\mathbb{LTS}(\Delta_A)$ then $\alpha \parallel \beta \overset{a}{\dashrightarrow} \alpha' \parallel \beta$. This has to be matched by $s \overset{a}{\dashrightarrow} s_a$. Furthermore, $s_a \xrightarrow{a'} s$ has to be matched by $\alpha' \parallel \beta \xrightarrow{a'} \alpha' \parallel \beta'$. This means that $\beta \xrightarrow{a} \beta'$ in $\mathbb{LTS}(\Delta_B)$ and that $(\alpha', \beta') \in \mathcal{R}$. $\square$

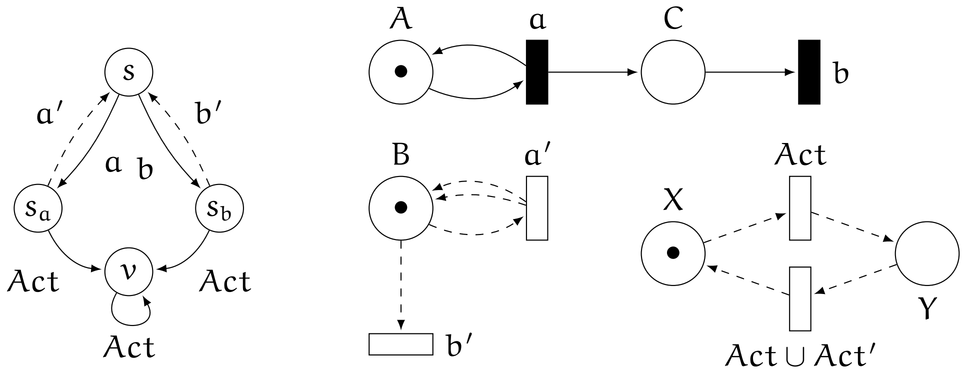

Figure 4: $s \leq_m A \parallel B \parallel X$ where the original two BPPs are again given by $A \xrightarrow{a} A \parallel C$, $C \xrightarrow{b} \varepsilon$, $B \xrightarrow{a} B \parallel B$, $B \xrightarrow{b} \varepsilon$.

Theorem 3.2. *The problem mFSM$\leq_m$mBPP is undecidable.*

Proof. We reduce the undecidable problem of simulation between two BPPs to the problem mFSM$\leq_m$mBPP. The proof is similar to the previous one. However, as the situation is not entirely symmetric (the requirement that $\Delta_{must} \subseteq \Delta_{may}$ introduces asymmetry), we need to modify the construction somewhat.

Let again A, B be two BPP processes with underlying PRS Δ_A and Δ_B; w.l.o.g. $\Delta_A \cap \Delta_B = \emptyset$. We rename all actions of Δ_B from a to a'. Let Act' be the set of these renamed actions and let Δ'_B be the modification of Δ_B. We further create a new PRS as follows:

$$\Delta_X = \{(X, a, Y) \mid a \in Act\} \cup \{(Y, x, X) \mid x \in Act \cup Act'\}$$

The mBPP is defined as $(\Delta_A \cup \Delta'_B \cup \Delta_X, \Delta_A)$, i.e. the (modified) transitions of B are just may, the transitions of A are both must and may, and the new transitions of Δ_X are may.

We then build a finite mPRS as follows. The states are $\{s, v\} \cup \{s_a \mid a \in Act\}$.

- $s \xrightarrow{a} s_a$ for all $a \in Act$ (with the corresponding may transitions)
- $s_a \overset{a'}{\dashrightarrow} s$ for all $a \in Act$
- $s_a \xrightarrow{a} v$ for all $a \in Act$ (with the corresponding may transitions)
- $v \xrightarrow{a} v$ for all $a \in Act$ (with the corresponding may transitions)

The construction is illustrated in Figure 4.

We now show that $A \leq_{sim} B$ iff $s \leq_m A \parallel B \parallel X$. As in the previous proof, α denotes a process of Δ_A while β denotes a process of Δ_B.

We first show that $\nu \leq_m \alpha \parallel \beta \parallel V$ for all $V \in \{X, Y\}$ and all processes α, β. Whenever the attacker plays a must transition of α, it is matched by $\nu \xrightarrow{a} \nu$. Whenever the attacker plays a may transition of ν, it is matched either by $X \overset{a}{\dashrightarrow} Y$ or by $Y \overset{a}{\dashrightarrow} X$ (α and β are unaffected).

$\Rightarrow$: Let $\mathcal{R} = \{(s, \alpha \parallel \beta \parallel X) \mid \alpha \leq_{sim} \beta\}$. We show that $\mathcal{R}$ can be extended to be a modal refinement relation. Let $(s, \alpha \parallel \beta \parallel X) \in \mathcal{R}$:

- If the attacker plays $s \overset{a}{\dashrightarrow} s_a$ then the defender can play $\alpha \parallel \beta \parallel X \overset{a}{\dashrightarrow} \alpha \parallel \beta \parallel Y$. The attacker then has two possibilities:
  - if the attacker plays $s_a \overset{a'}{\dashrightarrow} s$ then the defender can play $\alpha \parallel \beta \parallel Y \overset{a'}{\dashrightarrow} \alpha \parallel \beta \parallel X$ and the game is back in $\mathcal{R}$;
  - if the attacker plays $\alpha \parallel \beta \parallel Y \xrightarrow{a} \alpha' \parallel \beta \parallel Y$ then the defender can play $s_a \xrightarrow{a} \nu$ and win due to the fact above.
- If the attacker plays $\alpha \parallel \beta \parallel X \xrightarrow{a} \alpha' \parallel \beta \parallel X$ then the defender has to play $s \xrightarrow{a} s_a$. The attacker then has three possibilities:
  - if the attacker plays $\alpha' \parallel \beta \parallel X \xrightarrow{b} \alpha'' \parallel \beta \parallel X$ then the defender can play $s_a \xrightarrow{b} \nu$ and win due to the fact above.
  - if the attacker plays $s_a \overset{b}{\dashrightarrow} \nu$ then the defender can play $\alpha' \parallel \beta \parallel X \overset{b}{\dashrightarrow} \alpha' \parallel \beta \parallel Y$ and win due to the fact above.
  - if the attacker plays $s_a \overset{a'}{\dashrightarrow} s$ then the defender plays $\alpha' \parallel \beta \parallel X \overset{a'}{\dashrightarrow} \alpha' \parallel \beta' \parallel X$ where β' is a process such that $\beta \xrightarrow{a} \beta'$ in $\mathbb{LTS}(\Delta_B)$ and $\alpha' \leq_{sim} \beta'$. Such process has to exist due to $\alpha \leq_{sim} \beta$. Therefore, $(s, \alpha' \parallel \beta' \parallel X) \in \mathcal{R}$.

$\Leftarrow$: Let $\mathcal{R} = \{(\alpha, \beta) \mid s \leq_m \alpha \parallel \beta \parallel X\}$. We show that $\mathcal{R}$ is a simulation. Let $(\alpha, \beta) \in \mathcal{R}$.

- If $\alpha \xrightarrow{a} \alpha'$ then $\alpha \parallel \beta \parallel X \xrightarrow{a} \alpha' \parallel \beta \parallel X$. This has to be matched by $s \xrightarrow{a} s_a$. Furthermore, $s_a \overset{a'}{\dashrightarrow} s$ has to be matched by $\alpha' \parallel \beta \parallel X \overset{a'}{\dashrightarrow} \alpha' \parallel \beta' \parallel X$ (note that neither α' nor X can make an a'-transition). This means that $\beta \xrightarrow{a} \beta'$ in $\mathbb{LTS}(\Delta_B)$ and that $(\alpha', \beta') \in \mathcal{R}$. □

4 Decidability Results

We prove that the problems mFSM$\leq_m$mPDA and mPDA$\leq_m$mFSM are decidable and EXPTIME-complete like the corresponding simulation problems.

We modify the result of [KM02b] and show that, in certain classes of mPRS, refinement can be reduced to simulation.

We first need some auxiliary definitions. Let d be the maximal outdegree (we assume that the input MTS are finitely branching), let k be $|Act|$, the number of actions. We assume a fixed ordering of the actions, i.e. $Act = \{a_1, \ldots, a_k\}$. Let $\Lambda = \{\lambda_i^j \mid 1 \leq i \leq k, 1 \leq j \leq d\}$ and $\Delta = \{\delta_i^j \mid 1 \leq i \leq k, 1 \leq j \leq d\}$ be new actions. Let further γ be a new action. We define $Act' = Act \cup \Lambda \cup \Delta \cup \{\gamma\}$. Let further $mustsucc(s, a) = \{t \mid s \xrightarrow{a} t\}$. Assume arbitrary, but fixed linear ordering on $mustsucc(s, a)$. Then $s(a, j)$ denotes the jth process in $mustsucc(s, a)$.

An *A-translation* of $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ is an LTS $(\mathcal{P}', \longrightarrow)$ together with an injective mapping $f : \mathcal{P} \to \mathcal{P}'$ satisfying the following conditions:

- If $s \overset{a}{\dashrightarrow} s'$ then $f(s) \xrightarrow{a} f(s')$.
- $f(s) \xrightarrow{\lambda} f(s')$ for every $s \in \mathcal{P}$ and $\lambda \in \Lambda$.
- For all $s \in \mathcal{P}$, $1 \leq i \leq k$ and $1 \leq j \leq d$ we have the following:
  - if $j \leq |mustsucc(s, a_i)|$ then $f(s) \xrightarrow{\delta_i^j} f(s(a_i, j))$.
  - if $j > |mustsucc(s, a_i)|$ then $f(s) \xrightarrow{\delta_i^j} q_j$ where q_j is a state that can perform λ. There are no additional requirements on q_j.
- For every $s \in \mathcal{P}$ the state $f(s)$ has only the transitions admitted above.

For the second translation, we need the notion of a *universal process*. It is a process u such that $u \xrightarrow{a} u$ for all $a \in Act'$.

A *D-translation* of $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ is an LTS $(\mathcal{P}', \longrightarrow)$ together with an injective mapping $g : \mathcal{P} \to \mathcal{P}'$ satisfying the following conditions:

- If $t \overset{a}{\dashrightarrow} t'$ then $g(t) \xrightarrow{a} g(t')$.
- $g(t) \xrightarrow{\delta} u$ for every t and $\delta \in \Delta$, where u is an arbitrary universal process.
- For all $t \in \mathcal{P}$, all $1 \leq i \leq k$ and $1 \leq j \leq d$:
  - if $j \leq |mustsucc(t, a_i)|$ then $g(t) \xrightarrow{\lambda_i^j} r_{i,j,\ell}$ for each $1 \leq \ell \leq d$. Here, $r_{i,j,\ell}$ is a state of $\mathcal{P}'$ which has
    * exactly one δ_i^ℓ transition $r_{i,j,\ell} \xrightarrow{\delta_i^\ell} g(t(a_i, j))$

* for every $a \in \mathrm{Act}' \setminus \{\delta_i^\ell\}$ there is a universal process u and a transition $r_{i,j,\ell} \xrightarrow{a} u$.

– if $j > |\mathrm{mustsucc}(t, a_i)|$ then $g(t) \xrightarrow{\lambda_i^j} u$ where u is a universal process.

- For every $t \in \mathcal{P}$ the process $g(t)$ has only the transitions admitted above.

The following theorem can be proved similarly to the proof in [KM02b].

Theorem 4.1. *Let $(\mathcal{P}_1, \dashrightarrow_1, \longrightarrow_1)$, $(\mathcal{P}_2, \dashrightarrow_2, \longrightarrow_2)$ be two MTS, let $(\mathcal{P}_1', \longrightarrow_1)$ and $(\mathcal{P}_2', \longrightarrow_2)$ be their A and D-translations, respectively. Let $S \in \mathcal{P}_1$, $T \in \mathcal{P}_2$. We have that $S \leq_m T$ iff $f(S) \leq_{sim} g(T)$.*

The applicability of this method is the same (modulo the modal extension) as the applicability of the original method. Both the A-translation and the D-translation preserve the following subclasses of PRS: PDA, BPA, FSM, nPDA, nBPA and OC. Here, nPDA and nBPA are the normed variants (every process may be rewritten to ε in finite number of steps) of PDA and BPA, respectively. OC is the subclass of one-counter automata, i.e. PDA with only one stack symbol. Furthermore, the A-translation also preserves determinism.

As a direct corollary of the previous remark and the results of [KM02a], we obtain the following.

Theorem 4.2. *The problem mPDA$\leq_m$mFSM is EXPTIME-complete in both ways, even if the mFSM is of a fixed size. The problem mBPA$\leq_m$mFSM is EXPTIME-complete in both ways, but if the mFSM is of a fixed size, it is PTIME-complete.*

4.1 Visibly PDA

We have seen that the refinement relation is undecidable between any two infinite classes of the hierarchy depicted in Figure 1. However, there are other subclasses where the refinement is decidable. In this section, we show that the refinement between two modal visibly PDA is decidable.

Definition 4.3. *A PDA is a* visibly PDA (vPDA) *if there is a partitioning $Act = Act_c \uplus Act_r \uplus Act_i$ such that every rule $pX \xrightarrow{a} q\alpha$ satisfies the following:*

- *if $a \in Act_c$ then $|\alpha| = 2$ (call),*

- *if $a \in Act_r$ then $|\alpha| = 0$ (return),*

- *if* $a \in Act_i$ *then* $|\alpha| = 1$ *(internal).*

The modal extension (mvPDA) is straightforward; its subclass mvBPA can be defined similarly.

In order to prove decidability, we make use of the idea of [Srb06] for showing that simulation between two vPDA is decidable. We modify and simplify the method somehow, as the original method is used to prove decidability of various kinds of equivalences and preorders, while we are only considering the modal refinement.

Theorem 4.4. *The problem mvPDA$\leq_{\mathrm{m}}$mvPDA is decidable.*

Proof. Let $(\Delta_{\mathrm{may}}, \Delta_{\mathrm{must}})$ be a mvPDA with a stack alphabet Γ and a set of control states $\mathcal{Q}$. Let sA and tB be two processes of the mvPDA. Note that for simplicity we consider two processes of a single mvPDA. However, as a disjoint union of two mPRS is a mPRS, this also solves the case of two distinct mvPDA. Our goal is to transform the mvPDA into a PDA with a distinguished process such that this process satisfies certain μ-calculus formula if and only if $sA \leq_{\mathrm{m}} tB$.

We create a PDA Δ' with actions $\mathrm{Act}' = \{att, def\}$, stack alphabet $\Gamma' = \mathcal{G} \times \mathcal{G}$ where $\mathcal{G} = \Gamma \cup (\Gamma \times \Gamma) \cup (\Gamma \times \mathrm{Act}) \cup \{\varepsilon\}$, and control states $\mathcal{Q}' = \mathcal{Q} \times \mathcal{Q}$. We write Y_a instead of (Y, a) as an element of $\mathcal{G}$.

We use a (stack merging) partial mapping $[X\alpha, Y\beta] = (X, Y)[\alpha, \beta]$, $[\varepsilon, \varepsilon] = \varepsilon$. In the following, we abuse the notation of the rules, as we did in the introduction, and write e.g. $pX \overset{a}{\dashrightarrow} p'\alpha$ instead of $(pX, a, p'\alpha) \in \Delta_{\mathrm{may}}$.

The set of rules of Δ' is as follows:

- Whenever $pX \overset{a}{\dashrightarrow} p'\alpha$ then
  - $(p, q)(X, Y) \xrightarrow{att} (p', q)(\alpha, Y_a)$ for every $q \in \mathcal{Q}$ and $Y \in \Gamma$
  - $(q, p)(\beta, X_a) \xrightarrow{def} (q, p')[\beta, \alpha]$ for every $q \in \mathcal{Q}$ and $\beta \in \Gamma \times \Gamma \cup \Gamma \cup \{\varepsilon\}$
- Whenever $pX \xrightarrow{a} p'\alpha$ then
  - $(q, p)(Y, X) \xrightarrow{att} (q, p')(Y_a, \alpha)$ for every $q \in \mathcal{Q}$ and $Y \in \Gamma$
  - $(p, q)(X_a, \beta) \xrightarrow{def} (p', q)[\alpha, \beta]$ for every $q \in \mathcal{Q}$ and $\beta \in \Gamma \times \Gamma \cup \Gamma \cup \{\varepsilon\}$

Note that $[\alpha, \beta]$ and $[\beta, \alpha]$ is always well defined as $|\alpha| = |\beta|$ is guaranteed (the transition that created β has to have the same label as the transition that creates α – this is guaranteed via X_a).

We now claim that $sA \leq_m tB$ iff $(s,t)(A,B) \models \varphi$ where φ is an alternation-free μ-calculus formula: $\varphi = \nu Z.[att]\langle def\rangle Z$. Let us prove both implications.

$\Rightarrow$: let $\mathcal{F} = \{(p,q)[\alpha,\beta] \mid p\alpha \leq_m q\beta \text{ and } |\alpha| = |\beta|\}$. We show that $\mathcal{F}$ is a fixed point of the formula. We need to show that for every *att*-move $(p,q)[\alpha,\beta] \xrightarrow{att} c$ there is a *def*-move $c \xrightarrow{def} (p',q')[\alpha',\beta']$ such that $(p',q')[\alpha',\beta'] \in \mathcal{F}$. There are two possibilities.

- $(p,q)[\alpha,\beta] \xrightarrow{att} (p',q)(\alpha',Y_a)[\gamma,\delta]$ due to rule $pX \overset{a}{\dashrightarrow} p'\alpha'$ where $\alpha = X\gamma$ and $\beta = Y\delta$. This means that $p\alpha \overset{a}{\dashrightarrow} p'\alpha'\gamma$ and because $p\alpha \leq_m q\beta$, we have that $q\beta \overset{a}{\dashrightarrow} q'\beta'\delta$ due to some rule $qY \overset{a}{\dashrightarrow} q'\beta'$ such that $p\alpha'\gamma \leq_m q\beta'\delta$. Thus also $(p',q)(\alpha',Y_a)[\gamma,\delta] \xrightarrow{def} (p',q')[\alpha'\gamma,\beta'\delta]$ and $(p',q')[\alpha'\gamma,\beta'\delta] \in \mathcal{F}$.

- $(p,q)[\alpha,\beta] \xrightarrow{att} (p,q')(X_a,\beta')[\gamma,\delta]$. This situation is symmetric and is resolved similarly to the previous one (only this time, using must transitions instead of may transitions).

$\Leftarrow$: Let $\mathcal{R} = \{(p\alpha,q\beta) \mid (p,q)[\alpha,\beta] \models \varphi \text{ and } |\alpha| = |\beta|\}$. We show that $\mathcal{R}$ is a modal refinement. Let $(p\alpha,q\beta) \in \mathcal{R}$

- Let $p\alpha \overset{a}{\dashrightarrow} p'\alpha'$. This is due to a rule $pX \overset{a}{\dashrightarrow} p'\alpha''$, $\alpha = X\gamma$ and $\alpha' = \alpha''\gamma$ for some γ. In the PDA Δ' we thus have a transition $(p,q)[\alpha,\beta] \xrightarrow{att} (p',q)(\alpha'',Y_a)[\gamma,\beta']$ where $\beta = Y\beta'$. As $(p,q)[\alpha,\beta]$ satisfies φ, there has to be a *def*-transition $(p',q)(\alpha'',Y_a)[\gamma,\beta'] \xrightarrow{def} (p',q')[\alpha'',\beta''][\gamma,\beta']$ and also $(p',q')[\alpha'',\beta''][\gamma,\beta'] \models \varphi$. This means that $q\beta \overset{a}{\dashrightarrow} q'\beta''\beta'$ and $(p'\alpha',q'\beta''\beta') \in \mathcal{R}$.

- The other case is again, (almost) symmetric and can be resolved as the previous case (using must transitions instead of may transitions). $\square$

The following theorem can be proved using complexity bounds for μ-calculus model checking, as in [Srb06].

Theorem 4.5. *The problem mvPDA$\leq_m$mvPDA is EXPTIME-complete, the problem mvBPA$\leq_m$mvBPA is PTIME-complete.*

4.2 Birefinement

Since the refinement is often undecidable, the same holds for refinement equivalence $(\leq_m \cap \geq_m)$. Nevertheless, one can consider an even stronger relation that is still useful. We define the notion of birefinement as the modification of refinement where we require both conditions of Definition 2.2 to be satisfied in both directions, similarly as bisimulation can be defined as a symmetric simulation.

Definition 4.6 (Birefinement). *A birefinement is a symmetric refinement. We say that* α *birefines* β *(*$\alpha \sim_m \beta$*) if there exists a birefinement containing* (α, β)*.*

This notion then naturally captures the bisimilarity of modal transition systems. Furthermore, the birefinement problem on MTS can be reduced to bisimulation on LTS in the following straightforward way. Let $(\Delta_{\mathrm{may}}, \Delta_{\mathrm{must}})$ and $(\Gamma_{\mathrm{may}}, \Gamma_{\mathrm{must}})$ be two mPRS over the same action alphabet *Act*. We create a new action $\overline{a}$ for every $a \in Act$. We then translate the mPRS into ordinary PRS as follows. Let $\Delta = \Delta_{\mathrm{may}} \cup \{(\alpha, \overline{a}, \beta) \mid (\alpha, a, \beta) \in \Delta_{\mathrm{must}}\}$ and similarly for Γ. It is then clear that if we take two processes δ of $(\Delta_{\mathrm{may}}, \Delta_{\mathrm{must}})$ and γ of $(\Gamma_{\mathrm{may}}, \Gamma_{\mathrm{must}})$ then the following holds: δ birefines γ if and only if δ and γ are bisimilar when taken as processes of Δ and Γ, respectively.

The decidability and complexity of birefinement is thus identical to that of bisimulation in the non-modal case. Therefore, we may apply the powerful result that bisimilarity between *any* PRS and FSM is decidable [KŘS05] to get the following theorem.

Theorem 4.7. *Birefinement between an mFSM and any mPRS is decidable.*

This is an important result since it allows us to check whether we can replace an infinite MTS with a particular finite one, which in turn may allow for checking further refinements.

5 Conclusions

We have defined a generic framework for infinite-state modal transition systems generated by finite descriptions. We investigated the corresponding notion of modal refinement on important subclasses and determined the decidability border. Although in some classes it is possible to extend the decidability of simulation to decidability of refinement, it is not possible always. We have shown that somewhat surprisingly the parallelism is a great obstacle for deciding the refinement relation. Therefore, the future work will concentrate on identifying conditions leading to decidability. One of the best candidates is imposing determinism, which has a remarkable effect on the complexity of the problem in the finite case [BKLS09] as well as in the only infinite case considered so far, namely modal Petri nets [EBHH10]. Further, we leave the question whether the problem becomes decidable in some cases when the refining system is an implementation open, too. Finally, it remains open to what extent can verification results on finite MTS, such as [BČK11], be extended to infinite-state MTS.

Table 1: Summary of the decidability results

decidable	mFSM $\lesseqgtr_{\mathrm{m}}$ mPDA, mvPDA $\lesseqgtr_{\mathrm{m}}$ mvPDA, mFSM $\sim_{\mathrm{m}}$ mPRS
undecidable	mFSM $\lesseqgtr_{\mathrm{m}}$ mBPP, mBPA $\lesseqgtr_{\mathrm{m}}$ mBPA

References

[AFdFE+11] Luca Aceto, Ignacio Fábregas, David de Frutos-Escrig, Anna Ingólfsdóttir, and Miguel Palomino. Graphical representation of covariant-contravariant modal formulae. In *EXPRESS*, pages 1–15, 2011.

[AHKV98] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating refinement relations. In *CONCUR*, pages 163–178, 1998.

[AM04] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *STOC*, pages 202–211, 2004.

[BČK11] Nikola Beneš, Ivana Černá, and Jan Křetínský. Modal transition systems: Composition and LTL model checking. In *ATVA*, pages 228–242, 2011.

[BFJ+11] Sebastian S. Bauer, Uli Fahrenberg, Line Juhl, Kim G. Larsen, Axel Legay, and Claus R. Thrane. Quantitative refinement for weighted modal transition systems. In *MFCS*, volume 6907 of *LNCS*, pages 60–71. Springer, 2011.

[BG00] Glenn Bruns and Patrice Godefroid. Generalized model checking: Reasoning about partial state spaces. In *CONCUR*, pages 168–182, 2000.

[BHJ10] Sebastian S. Bauer, Rolf Hennicker, and Stephan Janisch. Interface theories for (a)synchronously communicating modal I/O-transition systems. In *FIT*, pages 1–8, 2010.

[BK10] Nikola Beneš and Jan Křetínský. Process algebra for modal transition systemses. In *MEMICS*, pages 9–18, 2010.

[BKL+11] Nikola Beneš, Jan Křetínský, Kim G. Larsen, Mikael H. Møller, and Jiří Srba. Parametric modal transition systems. In *ATVA*, pages 275–289, 2011.

[BKL+12] Nikola Beneš, Jan Křetínský, Kim G. Larsen, Mikael H. Møller, and Jiří Srba. Dual-priced modal transition systems with time durations. In *LPAR*, pages 122–137, 2012.

[BKLS09] Nikola Beneš, Jan Křetínský, Kim G. Larsen, and Jiří Srba. On determinism in modal transition systems. *Theor. Comput. Sci.*, 410(41):4026–4043, 2009.

[BL90] Gérard Boudol and Kim G. Larsen. Graphical versus logical specifications. In *CAAP*, pages 57–71, 1990.

[BLPR11] Nathalie Bertrand, Axel Legay, Sophie Pinchinat, and Jean-Baptiste Raclet. Modal event-clock specifications for timed component-based design. *Science of Computer Programming*, To appear, 2011.

[BLS95] Anders Børjesson, Kim G. Larsen, and Arne Skou. Generality in design and compositional verification using TAV. *Formal Methods in System Design*, 6(3):239–258, 1995.

[BML11] Sebastian S. Bauer, Philip Mayer, and Axel Legay. MIO workbench: A tool for compositional design with modal input/output interfaces. In *ATVA*, pages 418–421, 2011.

[Bru97] Glenn Bruns. An industrial application of modal process logic. *Sci. Comput. Program.*, 29(1-2):3–22, 1997.

[CDL+10] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. Compositional design methodology with constraint markov chains. In *QEST*, pages 123–132, 2010.

[ČGL93] Kārlis Čerāns, Jens Chr. Godskesen, and Kim G. Larsen. Timed modal specification - theory and tools. In *CAV*, pages 253–267, 1993.

[CGLT09] Alarico Campetelli, Alexander Gruler, Martin Leucker, and Daniel Thoma. *Don't Know* for multi-valued systems. In *ATVA*, pages 289–305, 2009.

[dAH01] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120, 2001.

[DFFU07] Nicolás D'Ippolito, Dario Fischbein, Howard Foster, and Sebastián Uchitel. MTSA: Eclipse support for modal transition systems construction, analysis and elaboration. In *ETX*, pages 6–10, 2007.

[DGG97] Dennis Dams, Rob Gerth, and Orna Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.

[DLL+10] Alexandre David, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. ECDAR: An environment for compositional design and analysis of real time systems. In *ATVA*, pages 365–370, 2010.

[DN04] Dennis Dams and Kedar S. Namjoshi. The existence of finite abstractions for branching time model checking. In *LICS*, pages 335–344, 2004.

[EBHH10] Dorsaf Elhog-Benzina, Serge Haddad, and Rolf Hennicker. Process refinement and asynchronous composition with modalities. In *ACSD/Petri Nets Workshops*, pages 385–401, 2010.

[FS08] Harald Fecher and Heiko Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. *J. Log. Algebr. Program.*, 77(1-2):20–39, 2008.

[GH94] Jan Friso Groote and Hans Hüttel. Undecidable equivalences for basic process algebra. *Inf. Comput.*, 115(2):354–371, 1994.

[GHJ01] Patrice Godefroid, Michael Huth, and Radha Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR*, pages 426–440, 2001.

[GNRT10] Patrice Godefroid, Aditya V. Nori, Sriram K. Rajamani, and SaiDeep Tetali. Compositional may-must program analysis: unleashing the power of alternation. In *POPL*, pages 43–56, 2010.

[HJS01] Michael Huth, Radha Jagadeesan, and David A. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In *ESOP*, pages 155–169, 2001.

[Hüt94] Hans Hüttel. Undecidable equivalences for basic parallel processes. In *TACS*, pages 454–464, 1994.

[JLS11] Line Juhl, Kim G. Larsen, and Jiří Srba. Modal transition systems with weight intervals. *Journal of Logic and Algebraic programming*, 2011. To appear.

[JM95] Petr Jancar and Faron Moller. Checking regular properties of petri nets. In *CONCUR*, pages 348–362, 1995.

[JP01] Bart Jacobs and Erik Poll. A logic for the java modeling language JML. In *FASE*, pages 284–299, 2001.

[KM99] Antonín Kučera and Richard Mayr. Simulation preorder on simple process algebras. In *ICALP*, pages 503–512, 1999.

[KM02a] Antonín Kučera and Richard Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *MFCS*, volume 2420 of *Lecture Notes in Computer Science*, pages 433–445. Springer, 2002.

[KM02b] Antonín Kučera and Richard Mayr. Why is simulation harder than bisimulation? In *CONCUR*, pages 594–610, 2002.

[KŘS05] Mojmír Křetínský, Vojtech Řehák, and Jan Strejček. Reachability of hennessy-milner properties for weakly extended PRS. In *FSTTCS*, pages 213–224, 2005.

[LNW07] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal I/O automata for interface and product line theories. In *ESOP*, pages 64–79, 2007.

[LT88] Kim G. Larsen and Bent Thomsen. A modal process logic. In *LICS*, pages 203–210, 1988.

[LW94] Barbara Liskov and Jeannette M. Wing. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, 16(6):1811–1841, 1994.

[LX90] Kim G. Larsen and Liu Xinxin. Equation solving using modal transition systems. In *LICS*, pages 108–117, 1990.

[Lyn88] Nancy Lynch. I/O automata: A model for discrete event systems. In *22nd Annual Conference on Information Sciences and Systems*, pages 29–38. Princeton University, 1988.

[May00] Richard Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.

[Nam03] Kedar S. Namjoshi. Abstraction for branching time properties. In *CAV*, pages 288–300, 2003.

[Nym08] Ulrik Nyman. *Modal Transition Systems as the Basis for Interface Theories and Product Lines*. PhD thesis, Institut for Datalogi, Aalborg Universitet, 2008.

[RBB+09] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benôit Caillaud, and Roberto Passerone. Why are modalities good for interface theories? In *ACSD*. IEEE Computer Society Press, 2009.

[RBB+11] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benôit Caillaud, Axel Legay, and Roberto Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 108(1-2):119–149, 2011.

[Srb06] Jiří Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In *CSL*, pages 89–103, 2006.

[UC04] Sebastián Uchitel and Marsha Chechik. Merging partial behavioural models. In *SIGSOFT FSE*, pages 43–52, 2004.

A Refinement Game

For reader's convenience, we recall the game characterization of modal refinement. As a matter of fact, it is a simple extension of the standard game-theoretic characterization of bisimilarity.

A *refinement game* on a pair of processes (s, t) is a two-player game between *Attacker* and *Defender*. The game is played in *rounds*. In each round the players change the *current pair of processes* (p, q) (initially $p = s$ and $q = t$) according to the following rule:

1. Attacker chooses an action $a \in Act$ and one of the processes p or q. If he chose p then he performs a move $p \overset{a}{\dashrightarrow} p'$ for some p'; if he chose q then he performs a move $q \overset{a}{\longrightarrow} q'$ for some q'.

2. Defender responds by choosing a transition under a in the other process. If Attacker chose the move from p, Defender has to answer by a move $q \overset{a}{\dashrightarrow} q'$ for some q'; if Attacker chose the move from q, Defender has to answer by a move $p \overset{a}{\longrightarrow} p'$ for some p'.

3. The new current pair of processes becomes (p', q') and the game continues with a next round.

The game is similar to standard bisimulation game with the exception that Attacker is only allowed to attack on the left-hand side using may transitions (and Defender answers by may transitions on the other side), while on the right-hand side Attacker attacks using must transitions (and Defender answers by must transitions in the left-hand side process).

Any *play* (of the modal game) thus corresponds to a sequence of pairs of processes formed according to the above rule. A play (and the corresponding sequence) is finite if and only if one of the players gets stuck (cannot make a move). The player who got stuck lost the play and the other player is the winner. If the play is infinite then Defender is the winner.

The following proposition is by a standard argument in analogy with strong bisimulation games.

Proposition A.1. *It holds that $s \leq_m t$ iff Defender has a winning strategy in the refinement game starting with the pair (s, t); and $s \not\leq_m t$ iff Attacker has a winning strategy.*