Fakultät für Informatik

Technische Universität München

Bachelor's thesis in Informatics

Algorithms for refinement of modal process rewrite systems

Philipp Meyer

Fakultät für Informatik

Technische Universität München

Bachelor's thesis in Informatics

Algorithms for refinement of modal process rewrite systems

Algorithmen zur Verfeinerung von modalen Prozessersetzungssystemen

Author:	Philipp Meyer
Supervisor:	Univ.-Prof. Dr. Dr. h.c. Javier Esparza
Advisor:	M. Sc. Jan Křetínský
Submission Date:	April 4, 2013

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

Munich, April 4, 2013

Philipp Meyer

Contents

1 Introduction

2 Theory

2.1 Basic definitions

process rewrite systems [May00, Esp01].

Definition 1 (Process). The set of processes $\mathcal{P}$ over a set of constants $Const$ is given by

$$\frac{}{\varepsilon \in \mathcal{P}}\,(0) \qquad \frac{X \in Const}{X \in \mathcal{P}}\,(1) \qquad \frac{p \in \mathcal{P} \qquad q \in \mathcal{P}}{p \cdot q \in \mathcal{P}}\,(S) \qquad \frac{p \in \mathcal{P} \qquad q \in \mathcal{P}}{p \| q \in \mathcal{P}}\,(P)$$

Processes are considered modulo the usual structural congruence, i.e. the smallest congruence such that the operator $\cdot$ is associative, $\|$ is associative and commutative and ε is a unit for both $\cdot$ and $\|$.

From here on we will denote processes by lowercase letters $p, q, \ldots$ and single constants by uppercase letters $P, Q, \ldots$.

The class of processes that can be produced just with rule 0, 1 and S, i.e. contain no $\|$, is the class of *sequential processes* **S**. The class of processes that can be produced just with rule 0, 1 and P, i.e. contain no $\cdot$, is the class of *parallel processes* **P**.

Definition 2 (Size of a process). The size $|p|$ of a process term p is defined by

$$\begin{aligned} |\varepsilon| &= 0 \\ |X| &= 1 \\ |p \cdot q| &= |p| + |q| \\ |p \| q| &= |p| + |q| \end{aligned}$$

2.2 Modal transition system

Modal transition system definition from [BK12]:

Definition 3 (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 $\longrightarrow \subseteq \dashrightarrow \subseteq \mathcal{P} \times Act \times \mathcal{P}$. An element $(p, a, q) \in \dashrightarrow$ is a *may transition*, also written as $p \overset{\text{a}}{\dashrightarrow} q$, and an element $(p, a, q) \in \longrightarrow$ is a *must transition*, also written as $p \xrightarrow{\text{a}} q$.

2.3 Modal refinement

Definition 4 (Refinement). Let $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ be an MTS and $p, q \in \mathcal{P}$ be processes. We say that p *refines* q, written $p \leq_m q$, if there is a relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ such that $(p, q) \in \mathcal{R}$ and for every $(p, q) \in \mathcal{R}$ and every $a \in Act$:

1. If $p \overset{\text{a}}{\dashrightarrow} p'$ then there is a transition $q \overset{\text{a}}{\dashrightarrow} q'$ s.t. $(p', q') \in \mathcal{R}$.
2. If $q \xrightarrow{\text{a}} q'$ then there is a transition $p \xrightarrow{\text{a}} p'$ s.t. $(p', q') \in \mathcal{R}$.

Modal refinement can also be seen as a refinement game from a pair of processes (p, q) where each side plays an attacking transition and the other a defending transition to reach a new state. The attacker wins if there is a strategy of attacking transitions where the defender always ends up in state where there are no defending transitions, otherwise the defender wins.

Definition 5 (Refinement game). Let $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ be an MTS and $p, q \in \mathcal{P}$ be processes.

We define the set of *attacking transitions* $Att = \{(p, q, p \overset{\text{a}}{\dashrightarrow} p') \mid p \overset{\text{a}}{\dashrightarrow} p'\} \cup \{(p, q, q \xrightarrow{\text{a}} q') \mid q \overset{\text{a}}{\dashrightarrow} q'\}$.

For an attacking transition $r \in Att$, the defending transitions are
we will make use of that notion

$$Def((p, q, r)) = \begin{cases} \{(q \overset{\text{a}}{\dashrightarrow} q', p', q') \mid q \overset{\text{a}}{\dashrightarrow} q'\} & \text{if } r = p \overset{\text{a}}{\dashrightarrow} p' \\ \{(p \xrightarrow{\text{a}} p', p', q') \mid p \xrightarrow{\text{a}} p'\} & \text{if } r = q \xrightarrow{\text{a}} q' \end{cases}$$

Then if $(p, q, r) \in Att$ and $(p', q') \in Def((p, q, r))$ we would get an attack transition $(p, q) \xrightarrow{\text{r,r'}}_a (p', q')$.

With that we can say that $p \leq_m q$ if there is a relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ such that $(p, q) \in \mathcal{R}$ and for every $(p, q, r) \in Att$ if $(p, q) \in \mathcal{R}$ then there is $(p', q') \in Def((p, q, r))$ such that $(p', q') \in \mathcal{R}$.

2.4 Modal process rewrite system

Definition 6 (Modal process rewrite system)**.** A *process rewrite system (PRS)* over an action alphabet Act is a finite relation $\Delta \subseteq \mathcal{P} \setminus \{\varepsilon\} \times Act \times \mathcal{P}$ Elements of Δ 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 $\Delta_{\text{may}} \subseteq \Delta_{\text{must}}$.

An mPRS $(\Delta_{\text{may}}, \Delta_{\text{must}})$ induces an MTS $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ as follows:

$$\frac{(p, a, p') \in \Delta_{\text{may}}}{p \overset{\text{a}}{\dashrightarrow} p'}\ (1) \qquad \frac{(p, a, p') \in \Delta_{\text{must}}}{p \xrightarrow{\text{a}} p'}\ (2)$$

$$\frac{p \overset{\text{a}}{\dashrightarrow} p'}{p \cdot q \overset{\text{a}}{\dashrightarrow} p \cdot q}\ (3) \qquad \frac{p \xrightarrow{\text{a}} p'}{p \cdot q \xrightarrow{\text{a}} p' \cdot q}\ (4) \qquad \frac{p \overset{\text{a}}{\dashrightarrow} p'}{p \| q \overset{\text{a}}{\dashrightarrow} p \| q}\ (5) \qquad \frac{p \xrightarrow{\text{a}} p'}{p \| q \xrightarrow{\text{a}} p' \| q}\ (6)$$

2.5 Attack tree

Definition 7 (Attack tree)**.** An *attack tree* over a set of processes $\mathcal{P}$ is a rooted tree where each node has two kinds of children. It is given by a triple (s, O, C), representing the tree with the root node labeled by $s \in \mathcal{P}^2$, the set of open edges O leading to states $s' \in \mathcal{P}^2$ and the set of closed edges C leading to the attack trees that are children of the root node.

The set of attack trees $\mathcal{T}$ constructable from an MTS $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ are defined inductively by

$$\frac{p, q \in \mathcal{P}, p \overset{\text{a}}{\dashrightarrow} p'}{((p, q), \{(p', q') \mid q \overset{\text{a}}{\dashrightarrow} q'\}, \emptyset) \in \mathcal{T}}\ (1)$$

$$\frac{p, q \in \mathcal{P}, q \xrightarrow{\text{a}} q'}{((p, q), \{(p', q') \mid p \xrightarrow{\text{a}} q'\}, \emptyset) \in \mathcal{T}}\ (2)$$

$$\frac{((p, q), O \uplus (p', q'), C) \in \mathcal{T} \qquad T = (p', q'), O', C') \in \mathcal{T}}{(s, O, C \cup T) \in \mathcal{T}}\ (3)$$

$$\frac{(s, O \uplus (r', p', q'), C) \in \mathcal{T} \qquad ((p', q', r''), O', C') \in \mathcal{T}}{(s, O, C \cup \{(r', (p', q', r''), O', C')\}) \in \mathcal{T}}\ (2)$$

Rules 1 and 2 specify an initial tree for an attacking rule with the possible defensive states while rule 3 replaces an open edge to a state with a tree with that state as its root. As we can see from the construction rules, every node (p, q) in the tree corresponds to an attacking transition applicable from that state, while the set of edges from that node corresponds exactly to the defending transition applicable from that state and attacking transition.

For an attack tree $T = ((p, q), O, C)$, the root is given by $root(T) = (p, q)$. We define the set of *open states* by the states that have an open edge to it, that is $open(((p, q, r), O, C)) = O \cup \bigcup_{T' \in C} open(T')$. We say that a tree is *closed* if it has no open states, that is $closed(T) \quad open(T) = \emptyset$.

Lemma 1. *If there are attack trees T with $root(T) = (p, q)$ and R with $root(R) = (p', q')$ and $(p', q') \in open(T)$, then there is an attack tree S with $root(S) = (p, q)$ and $open(S) = open(T) \setminus \{(p', q')\} \cup open(R)$*

Lemma 2. *If there are attack trees $T = ((p, q), O, C)$ and $R = ((p', q'), O', C')$ with $(p', q') \in open(T)$, then there is an attack tree S with $S = ((p, q), O' \setminus \{p', q'\}, C'')$ with $open(S) = open(T) \setminus \{(p', q')\} \cup open(R)$*

Proof. We prove the proposition by induction on T:

1. $T = ((p, q), O, \emptyset)$: Then $(p', q') \in O$ and we can construct $S = ((p, q), O \setminus \{(p', q')\}, \{R\}$ with $open(S) = open(T) \setminus \{(p', q')\} \cup open(R)$

2. $T = ((p, q), O, C \cup T'')$ from $T' = ((p, q), O \uplus \{(p'', q'')\}, C)$ and $T'' = (p'', q''), O'', C'')$. $open(T) = open(T') \setminus \{(p'', q'')\} \cup open(T'')$

   If $(p', q') \in open(T')$, by induction hypothesis we get $S' = ((p, q), (O \uplus \{(p'', q'')\}) \setminus \{(p', q'\}, C')$ with $open(S') = open(T') \setminus \{(p', q')\} \cup open(R)$.

   If $(p', q') \in open(T'')$, by induction hypothesis we get S'' with $root(S'') = (p'', q'')$ and $open(S'') = open(T'') \setminus \{(p', q')\} \cup open(R)$. else set $S'' = T''$.

   As $(p', q') \in open(T)$, if $(p', q') \in O$, then $(p', q') \neq (p'', q'')$. Therefore $S' = ((p, q), (O \setminus \{(p', q')\}) \uplus, C')$. Then from S' and S'' construct $S = ((p, q), O \setminus \{(p'', q'')\} \setminus \{(p', q'\}, C \cup S'')$ with $open(S) = open(S') \setminus \{(p'', q'')\} \cup open(S'')$. $= (open(T') \setminus \{(p', q')\} \setminus \{(p'', q'')\} \cup open(R)) \cup (open(T'') \setminus \{(p', q')\} \cup open(R)) = ((open(T') \setminus \{(p'', q'')\} \cup open(T'')) \setminus \{(p', q')\}) \cup open(R))$.

By replacing all nodes $((p'', q''), O', C')$ where $(p', q') \in O'$ with $((p'', q''), O \setminus \{(p', q')\}, C' \cup R)$. □

Theorem 1. *For an MTS $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ and processes $p, q \in \mathcal{P}$:*

$$(p \leq_m q) \iff \neg\exists T \in \mathcal{T} : root(T) = (p, q) \wedge closed(T)$$

Proof. $\Longrightarrow$: Assume $p \leq_m q$. Then there is a refinement relation $\mathcal{R}$. To show that for $(p, q) \in \mathcal{R}$ there is no closed tree from (p, q), we show for any T with $root(T) = (p, q)$, if T is closed, then $(p, q) \notin R$.

Let r be the attacking transition corresponding to (p, q) in T. For every fitting defending transition r' to (p', q') there is an edge from (p, q) to (p', q'). As T is closed, with $open(T) = O \cup \bigcup_{T' \in C} open(T') = \emptyset$ we get $O = \emptyset$ and every T' is closed.

Now we show the proposition by induction over the number of children $|C|$:

1. $|C| = 0$: Then there is an attacking transition, but no defending transition, therefore $(p, q) \notin \mathcal{R}$.

2. $|C| \geq 1$: Then there is an attacking rule, and for every defending transition leading to (p', q'), there is an edge to a closed tree T' with $root(T') = (p', q')$. By induction hypothesis we have $(p', q') \notin \mathcal{R}$ and therefore $(p, q) \notin \mathcal{R}$.

$\Longleftarrow$: Assume there is no closed attack tree T with $root(T) = (p, q)$. To show $p \leq_m q$, we show that $\mathcal{R} := \{(p', q') \mid \neg\exists T : root(T) = (p', q') \wedge closed(T)\}$ is a valid refinement relation with $(p, q) \in \mathcal{R}$.

For any attacking transition r and $(p, q) \in \mathcal{R}$, by inference rule 1 or 2 there exists an attacking tree $T = ((p, q), O, C)$. From all such T with $root(T) = (p, q)$, choose one where O is minimal with regard to the inclusion order. There exists $(p', q') \in O$ with $(p', q') \in \mathcal{R}$, because otherwise there would be a closed attack tree T' with $root(T') = (p', q')$ and by inference rule 3 we would get $T'' = ((p, q), O'', C \cup T')$ with $O'' = O \setminus \{(p', q')\} \subsetneq O$ in contradiction to the minimality of O. So for the attacking transition (p, q) there is a defending transition (p', q') with $(p', q') \in \mathcal{R}$. $\square$

2.6 Visibly pushdown automaton

Definition 8 (Visibly pushdown automaton)**.** A PRS Δ over the action alphabet Act is a *visibly pushdown automaton (vPDA)* if there is a partition $Act = Act_r \uplus Act_i \uplus Act_c$

such that every rule $(p, a, p') \in \Delta$ has the form

$$p = P \cdot S \qquad \text{and} \qquad p' = \begin{cases} Q & \text{if } a \in Act_r \quad \text{(return rule)} \\ Q \cdot T & \text{if } a \in Act_i \quad \text{(internal rule)} \\ Q \cdot T \cdot R & \text{if } a \in Act_c \quad \text{(call rule)} \end{cases}$$

A *modal visibly pushdown automaton (mvPDA)* is then an mPRS $(\Delta_{\text{may}}, \Delta_{\text{must}})$ such that Δ_{must} is a vPDA.

Definition 9 (Attack rule)**.** Let $(\Delta_{\text{may}}, \Delta_{\text{must}})$ be an mvPDA. An *attack rule* $(p, q) \longrightarrow_a S$ with $p, q \in \mathcal{P}$ and $S \subseteq \mathcal{P}$ is obtainable from the rewrite rules if it can be constructed by the following rules:

$$\frac{(p, a, p') \in \Delta_{\text{may}}}{(p, q) \longrightarrow_a \{(p', q') \mid (q, a, q') \in \Delta_{\text{may}}\}} \; (1)$$

$$\frac{(q, a, q') \in \Delta_{\text{must}}}{(p, q) \longrightarrow_a \{(p', q') \mid (p, a, p') \in \Delta_{\text{must}}\}} \; (2)$$

$$\frac{(p, q) \longrightarrow_a \{(p' \cdot P, q' \cdot Q)\} \uplus S \qquad (p', q') \longrightarrow_a S' \qquad \forall (p'', q'') \in S' : |p''| = 1}{(p, q) \longrightarrow_a S \cup \{(p'' \cdot P, q'' \cdot Q) \mid (p'', q'') \in S'\}} \; (3)$$

$$\frac{(p, q) \longrightarrow_a \{(p', q')\} \uplus S \qquad (p', q') \longrightarrow_a S' \qquad \forall (p'', q'') \in S' : |p''| = 1}{(p, q) \longrightarrow_a S \cup S'} \; (4)$$

Due to the constrainst on the rewrite rules of an mvPDA and the construction of the attack rules, we can see that for any element $(p, q) \longrightarrow_a S$ it holds that $|p| = |q| = 2$ and for any $(p', q') \in S$ that $1 \leq |p'| = |q'| \leq 3$.

When the rules 3 and 4 always combine a rule $(p, q) \longrightarrow_a S \uplus \{(p', q')\}$ on the left and rule $(p', q') \longrightarrow_a S'$ on right, it always holds that $|p'| = 2$ or $|p'| = 3$ and for all $(p'', q'') \in S'$ $|p''| = 1$. We will call a rule $p \longrightarrow_a S$ a *right-hand side* rule if $\forall (p', q') \in S : |p'| = 1$ and otherwise a *left-hand side* rule. This partitions the set of rules into two classes.

Lemma 3. *Given an MTS generated by a mvPDA, if there is an attack tree T with $root(T) = (p, q)$, then for any $s, t \in \mathcal{P}$, there is an attack tree R with $root(R) = (p \cdot s, q \cdot t)$ and $open(R) = \{(p' \cdot s, q' \cdot t) \mid (p', q') \in open(T)\}$.*

Proof. As p and q are the left-hand side of some rewrite rule, we have $|p| \geq 2$ and $|q| \geq 2$. Then by looking at the induction rules for an MTS from an mPRS, we have that if $p \overset{a}{\dashrightarrow} p'$, then $p \cdot s \overset{a}{\dashrightarrow} p' \cdot s$ and if $p \cdot s \overset{a}{\dashrightarrow} p' \cdot s$, then $p \overset{a}{\dashrightarrow} p'$, therefore $\{p \cdot s \overset{a}{\dashrightarrow} p' \cdot s \mid p \overset{a}{\dashrightarrow} p'\} = \{p \cdot s \overset{a}{\dashrightarrow} p' \cdot s\}$. The same holds for $\longrightarrow$.

We then prove the proposition by induction on T:

1. $T = ((p, q), O, \emptyset)$ from $p \overset{a}{\dashrightarrow} p'$ with $O = \{(p', q') \mid q \overset{a}{\dashrightarrow} q'\}$. Then also $p \cdot s \overset{a}{\dashrightarrow} p \cdot s'$ and with $O' = \{(p' \cdot s, q' \cdot t \mid q \cdot t \overset{a}{\dashrightarrow} q' \cdot t\} = \{(p' \cdot s, q' \cdot t \mid q \overset{a}{\dashrightarrow} q'\}$ $= \{(p' \cdot s, q' \cdot t \mid (p', q') \in O\}$ we can construct $R = ((p' \cdot s, q' \cdot t), O', \emptyset)$.

2. $T = ((p, q), O, \emptyset)$ from $q \overset{a}{\longrightarrow} q'$. This case is symmetric to the first one.

3. $T = ((p, q), O, C \cup T'')$ from $T' = ((p, q), O \uplus \{(p', q')\}, C)$ and $T'' = (p', q'), O', C')$. By induction hypothesis we get $R' = ((p \cdot s, q \cdot t), O' \uplus \{(p' \cdot s, q' \cdot t)\}, C')$ and $R'' = (p' \cdot s, q' \cdot t), O'', C')$. Then we can construct $R = ((p \cdot s, q \cdot t), O, C' \cup R'')$ with $open(R) = open(R') \setminus \{(p' \cdot s, q \cdot t)\} \cup open(R'') = \{(p' \cdot s, q' \cdot t) \mid (p', q') \in open(T')\} \setminus \{(p' \cdot s, q' \cdot t)\} \cup \{(p'' \cdot s, q'' \cdot t) \mid (p'', q'') \in open(T'')\}$ $= \{(p'' \cdot s, q'' \cdot t) \mid (p'', q'') \in open(T') \setminus \{p', q'\} \cup open(T'')\} = \{(p'' \cdot s, q'' \cdot t) \mid (p'', q'') \in open(T)\}$

□

Theorem 2. *For an mvPDA $(\Delta_{may}, \Delta_{must})$ with its induced MTS $(\mathcal{P}, \dashrightarrow, \longrightarrow)$, it holds that for any $P, S, Q, R \in Const$:*

$$\exists T : root(T) = (P \cdot S, Q \cdot R) \wedge closed(T) \iff (P \cdot S, Q \cdot T) \longrightarrow_a \emptyset$$

Proof. $\Rightarrow$: Assume T to be closed tree with $root(T) = (P \cdot S, Q \cdot T)$.

We show that for any partition of the tree $P = \{T_1, ..., T_n\}$ where every T_i is represented by a rule b_i, there is an attack rule representing the whole tree. Proof by induction over the length of the partition n:

1. $n = 1$: Then $T_1 = T$ and b_1 represents T.

2. $n > 1$: WLOG let $T_1 = (P \cdot S, Q \cdot T)$ be the part at the root of the tree T with succeding partitions C.

   As $n > 1$, there is $(p', q', r') \in C$. Then for $b_1 = (P \cdot S, Q \cdot T) \longrightarrow_a S$ we have by representation $(p', q') \in S$ and we have $|p'| = |q'| \geq 2$, as otherwise the rule r' would not be applicable from the state (p', q'). So b_1 is a left-hand side rule.

   For every partition T_i with no succeding partitions, we have $b_i = (p, q) \to \emptyset$, so that is a right-hand side rule.

Then by following the successors of the partitions from T_1, we will eventually come to a partition T_i followed by a partition T_j such that b_i is a left-hand side rule and b_j is a right-hand side rule.

Then by rule 3 or 4 we can combine b_i and b_j into a new rule b_0. This rule represents the partition T_i in $P' = P \setminus \{T_j\}$. Every other $T_k \in P'$ remains unchanged and is still represented by b_k. Then by induction hypothesis there is a b representing T.

$\Leftarrow$: We show that if $(p, q) \longrightarrow_a S$, then there is a tree T with $root(T) = (p, q)$ such that $open(T) = S$ by induction on the construction of $(p, q) \longrightarrow_a S$:

1. It was constructed by rule 1 from $(p, a, p') \in \Delta_{\text{may}}$. Then there is an attacking transition $p \overset{\text{a}}{\dashrightarrow} p'$ and for every $(q, a, q') \in \Delta_{\text{may}}$ there is an induced defending transition $q \overset{\text{a}}{\dashrightarrow} q'$. Then $S = \{(p', q') | q \overset{\text{a}}{\dashrightarrow} q'\}$ and by attack tree inference rule 1 there is $T = ((p, q), S, \emptyset)$ with $open(T) = S$.

2. It was constructed by rule 2 from $(q, a, q') \in \Delta_{\text{must}}$. Then there is an attacking transition $q \overset{\text{a}}{\longrightarrow} q'$ and for every $(p, a, p') \in \Delta_{\text{may}}$ there is an induced defending transition $p \overset{\text{a}}{\longrightarrow} p'$. Then $S = \{(p', q') | p \overset{\text{a}}{\dashrightarrow} p'\}$ and by attack tree inference rule 2 there is $T = ((p, q), S, \emptyset)$ with $open(T) = S$.

3. It was constructed by rule 3 from $(p, q) \longrightarrow_a \{(p' \cdot P, q' \cdot Q)\} \uplus S''$ and $(p', q') \longrightarrow_a S'$ with $S = S'' \cup S'''$ and $S''' = \{(p'' \cdot P, q'' \cdot Q) \mid (p'', q'') \in S'\}$. Then by induction hypothesis there is a tree T' with $root(T') = (p', q')$ and $open(T') = S'$ and a tree T'' with $root(T'') = (p, q)$ and $open(T'') = S'' \uplus \{(p' \cdot P, q' \cdot Q)\}$. By appying lemma 3 on T' there is a tree T''' with $root(T''') = (p' \cdot P, q' \cdot Q)$, $open(T''') = O''' \uplus \{(p' \cdot P, q' \cdot Q)\}$ and $O''' = \{(p'' \cdot P, q'' \cdot Q) \mid (p'', q'') \in S'\} = S'''$. By applying lemma 1 on T'' and T''' there is a tree T with $root(T) = (p, q)$ and $open(T) = S'' \cup S''' = S$.

4. It was constructed by rule 4 from $(p, q) \longrightarrow_a S'' \uplus \{(p', q')\}$ and $(p', q') \longrightarrow_a S'$ with $S = S'' \cup S'$. Then by induction hypothesis there is a tree T' with $root(T') = (p', q')$ and $open(T') = S'$ and a tree T'' with $root(T'') = (p, q)$ and $open(T'') = S'' \uplus \{(p', q')\}$. By applying lemma 1 on T' and T'' there is a tree T with $root(T) = (p, q)$ with $open(T) = S'' \cup S' = S$.

Therefore if $(P \cdot S, Q \cdot T) \longrightarrow_a \emptyset$, then there is a tree T with $root(T) = (P \cdot S, Q \cdot T)$ and $open(T) = \emptyset$. $\square$

3 The algorithm

3.1 Description

3.2 Implementation

Figure 3.1: Algorithm for calculating the attack rules on mvPDAs

function AttackRules($mvPDA = (\Delta_{\text{may}}, \Delta_{\text{must}})$)
  $rules \leftarrow \emptyset$
  for $P, Q, S, T \in Const(mvPDA), a \in Act(mvPDA), type \in \{\text{may}, \text{must}\}$ **do**
    if $type = \text{may}$ **then**
      $lhs \leftarrow (P \cdot S, Q \cdot T)$ ▷ Attack from left-hand side for may rules
    else
      $lhs \leftarrow (Q \cdot S, P \cdot Y)$ ▷ Attack from right-hand side for must rules
    end if
    for $(P \cdot S, a, p') \in \Delta_{type}$ **do**
      $rhs \leftarrow \emptyset$
      for $(Q \cdot T, a, q') \in \Delta_{type}$ **do**
        if $type = \text{may}$ **then**
          $newRhs \leftarrow (p', q')$
        else
          $newRhs \leftarrow (q', p')$
        end if
        $rhs \leftarrow rhs \cup \{newRhs\}$
      end for
      $rules \leftarrow rules \cup \{(lhs, rhs)\}$
    end for
  end for
  return $rules$
end function

Figure 3.2: Algorithm for combining attack rules

function $\textsc{Combine}(lhsRule = (lhs, lhsRhsSet), rhsRule = (rhsLhs, rhsSet))$
  $rules \leftarrow \emptyset$
  if $\forall rhs \in rhsSet : size(rhs) \leq 1$ **then**
    for $lhsRhs \in lhsRhsSet : lhsRhs = rhsLhs \cdot p$ **do**
      $newRhs \leftarrow (lhsRhsSet \setminus lhsRhs) \cup \{rhs \cdot p \mid rhs \in rhsSet\}$
      $rules \leftarrow rules \cup \{(lhs, newRhs)\}$
    end for
  end if
  return $rules$
end function

Figure 3.3: Refinement algorithm for mvPDAs

function $\text{VPDA}\textsc{Refinement}(P \cdot S, Q \cdot T, mvPDA)$ $\quad \triangleright\ P \cdot S \leq_m Q \cdot T$ given $mvPDA$
  $initial \leftarrow [P \cdot S, Q \cdot T]$
  $rules \leftarrow \textsc{AttackRules}(mvPDA)$
  while $\exists lhsRule, rhsRule \in rules : \textsc{Combine}(lhsRule, rhsRule) \not\subset rules$ **do**
    $rules \leftarrow rules \cup \textsc{Combine}(lhsRule, rhsRule)$
  end while
  return $(initial, \emptyset) \in rules$
end function

3.3 Soundness and completeness

Follows from theorem and theorem and

3.4 Runtime

3.5 Optimizations

3.6 Input and output

3.7 Performance evaluation

3.8 Example

Figure 3.4 and 3.5 define two mvPDA. The corresponding may transitions for the must transitions are implied. The problem is to decide whether $p \cdot S \leq_m q \cdot S$.

$$
\begin{aligned}
P \cdot S &\xrightarrow{\text{coin}} P \cdot M \cdot S \\
P \cdot M &\xrightarrow{\text{coin}} P \cdot M \cdot M \\
P \cdot M &\xrightarrow{\text{tea}} T \\
P \cdot M &\xrightarrow{\text{coffee}} c \\
T \cdot M &\xrightarrow{\text{tea}} T \\
T \cdot S &\xrightarrow{\text{coin}} P \cdot M \cdot S \\
c \cdot M &\xrightarrow{\text{coffee}} c \\
c \cdot S &\xrightarrow{\text{coin}} P \cdot M \cdot S
\end{aligned}
$$

Figure 3.4: mvPDA for process $P \cdot S$

$$
\begin{aligned}
Q \cdot S &\overset{\text{coin}}{\dashrightarrow} Q \cdot T \cdot S \\
Q \cdot S &\overset{\text{coin}}{\dashrightarrow} Q \cdot C \cdot S \\
Q \cdot T &\overset{\text{coin}}{\dashrightarrow} Q \cdot T \cdot T \\
Q \cdot C &\overset{\text{coin}}{\dashrightarrow} Q \cdot C \cdot C \\
Q \cdot T &\xrightarrow{\text{tea}} Q \\
Q \cdot T &\overset{\text{coffee}}{\dashrightarrow} Q \\
Q \cdot C &\overset{\text{tea}}{\dashrightarrow} Q \\
Q \cdot C &\xrightarrow{\text{coffee}} Q
\end{aligned}
$$

Figure 3.5: mvPDA for process $Q \cdot S$

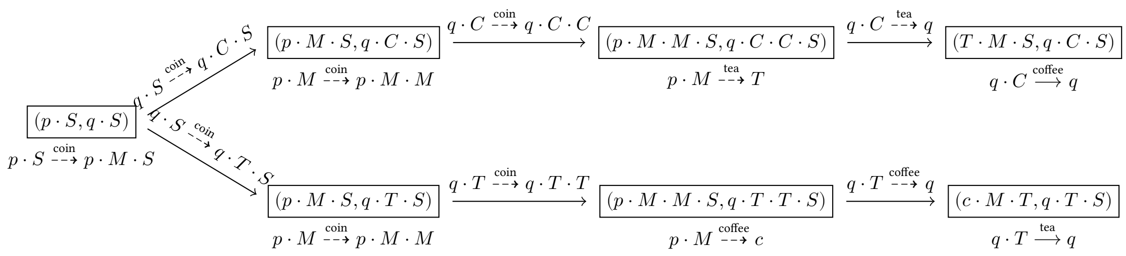

Figure 3.6: Tree for winning strategy

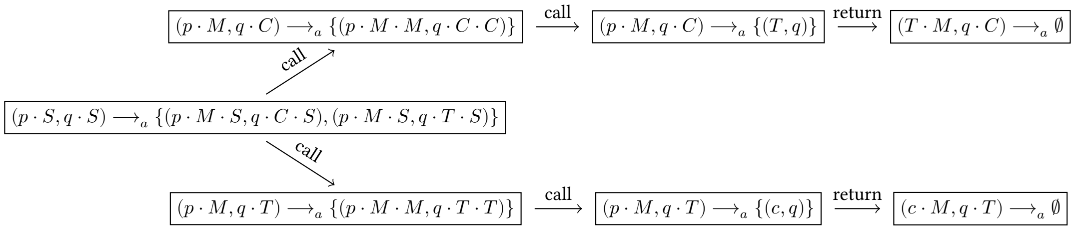

Figure 3.7: Tree for winning strategy with attack rules

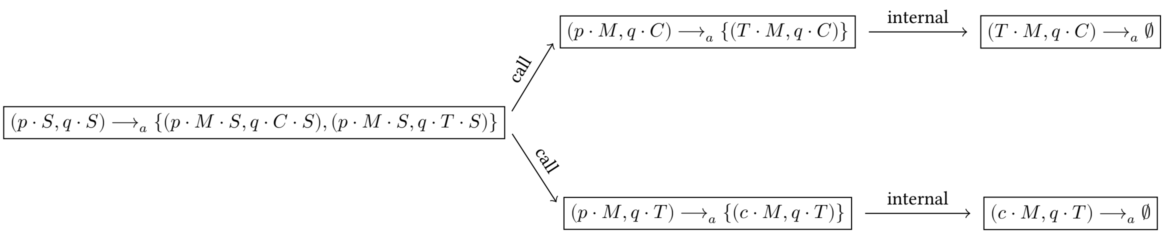

Figure 3.8: Merged tree for winning strategy with attack rules after one step

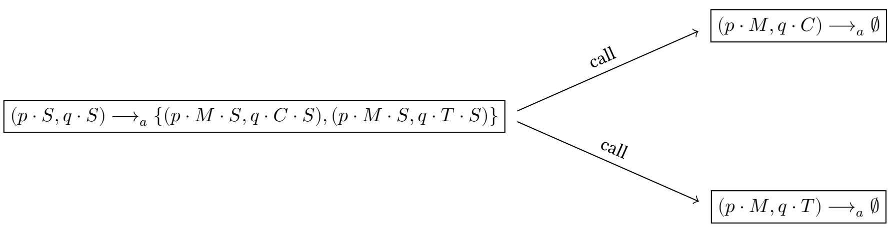

Figure 3.9: Merged tree for winning strategy with attack rules after two steps

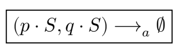

Figure 3.10: Final merged tree for winning strategy

4 Conclusion

Bibliography

[BK12] Nikola Benes and Jan Kretínský, *Modal process rewrite systems*, ICTAC (Abhik Roychoudhury and Meenakshi D'Souza, eds.), Lecture Notes in Computer Science, vol. 7521, Springer, 2012, pp. 120--135.

[BKLS09] Nikola Benes, Jan Kretínský, Kim Guldstrand Larsen, and Jirí Srba, *On determinism in modal transition systems*, Theor. Comput. Sci **410** (2009), no. 41, 4026--4043.

[Esp01] Javier Esparza, *Grammars as processes*, Lecture Notes in Computer Science **2300** (2001), 277--298.

[May00] Richard Mayr, *Process rewrite systems*, Inf. Comput **156** (2000), no. 1-2, 264--286.