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 2, 2013

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

Munich, April 2, 2013

Philipp Meyer

Contents

1 Introduction

2 Theory

2.1 Modal process rewrite system

Modal process rewrite systems [BK12] are a modal extension of process rewrite systems [May00, Esp01]. They induce a modal transition systems [BKLS09].

2.2 Basic definitions

Definition 1 (Process term)**.** The set of process terms over a set of constants $Const$ is given by

$$\frac{}{\varepsilon \in \mathcal{P}}\,(0) \qquad \frac{X \in Const}{X \in \mathcal{P}}\,(1)$$

$$\frac{p \in \mathcal{P} \qquad q \in \mathcal{P}}{p \cdot q \in \mathcal{P}}\,(S) \qquad \frac{p \in \mathcal{P} \qquad q \in \mathcal{P}}{p \| q \in \mathcal{P}}\,(P)$$

The processes expressions 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 $\|$.

Processes that can be produced just with rule 0, 1 and S, i.e. contain no $\|$, are called *sequential processes* and processes that can be produced just with rule 0, 1 and P, i.e. contain no $\cdot$, are called *parallel processes*

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

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

Process terms will be denoted by lowercase letters $p, q, \ldots$ while single constants are denoted by uppercase letters $P, Q, \ldots$.

Definition 3 (Constants of a process term)**.** The set of constants $Const(p)$ appearing in a process term p is inductively defined by

$$\begin{aligned} Const(\varepsilon) &= \emptyset \\ Const(X) &= \{X\} \\ Const(p \cdot q) &= Const(p) \cup Const(q) \\ Const(p \| q) &= Const(p) \cup Const(q) \end{aligned}$$

2.3 Modal transition system

Modal transition system definition from [BK12]:

Definition 4 (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 $\longrightarrow \subseteq \dashrightarrow \subseteq \mathcal{P} \times Act \times \mathcal{P}$. An element $(p, a, q) \in \dashrightarrow$ is a *may transition*, also written as $p \overset{a}{\dashrightarrow} q$, and an element $(p, a, q) \in \longrightarrow$ is a *must transition*, also written as $p \xrightarrow{a} q$.

2.4 Modal process rewrite system

Definition 5 (Modal process rewrite system)**.** A *process rewrite system (PRS)* over a set of constants $Const$ and action alphabet Act is a finite relation. $\Delta \subseteq \mathcal{P} \times Act \times \mathcal{P}$, 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 $\Delta_{\text{may}} \subseteq \Delta_{\text{must}}$.

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

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

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

2.5 Modal refinement

Definition 6 (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 7 (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

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

Then if $(p, q, r) \in Att$ and $(p, q, r') \in Def((p, q, r))$ we would get an attack transition $(p, q) \longrightarrow (p', q')$

With that we can define refinement by $p \leq_m q$, 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 $(p', q', r') \in Def((p, q, r))$ such that $(p', q') \in \mathcal{R}$.

2.6 Attack tree

Definition 8 (Attack transition and attack tree)**.** Let $(\mathcal{P}, \dashrightarrow, \longrightarrow)$ be an MTS. An *attack trees* is given by its node $(p, q, r) \in \mathcal{P}^2 \times (\dashrightarrow \cup \longrightarrow)$, a set of open edges $O \subseteq \mathcal{P}^2 \times (\dashrightarrow$

$\cup \longrightarrow$) and set of closed edges to children $C \subseteq (P^2 \times (\dashrightarrow \cup \longrightarrow)) \times \mathcal{T}$ where $\mathcal{T}$ is the set of trees. They are defined inductively by

$$\frac{(p,q,r) \in Att}{((p,q,r), Def(p,q,r), \emptyset)} \ (1)$$

$$\frac{((p,q,r), O \uplus (p',q',r'), C) \text{ atree} \qquad ((p',q',r'), O', C') \text{ atree}}{((p,q,r), O, C \cup \{((p',q',r'), O', C')\}) \text{ atree}} \ (2)$$

The root of an attack tree is $root((p,q,r), O, C) = (p,q)$

The set of open edges for an attack tree is given by $open(((p,q,r), O, C) \text{ atree}) = O \cup \bigcup_{T \in C} open(T)$. A tree T is a *closed tree*, written as $closed(T)$, iff $open(T) = \emptyset$. The set of edges from the root node of a tree is given by $edges((p,q,r), O, C) = O \cup \{(p',q',r') \mid ((p',q',r'), O', C') \in C$.

Intuitively, an attack transition $(p,q) \longrightarrow_a O$ means that from the state (p,q), there is a sequence of *attack transitions*, that is a may transition from the left side or a must transition from the right side, such that O is the set of reachable states by applying appropriate *defending transition*, that is a transition of the same type and with the same action symbol from the other side.

Lemma 1. *For two attack trees* $T = ((p,q,r), O, C)$ *atree and* $T' = ((p,q,r), O', C')$ *atree, we have* $edges(T) = edges(T')$.

Proof. By looking at the induction rules for attack trees, we see for a fixed (p,q,r) the open edges O are always initialised with $Def((p,q,r))$ in the base case and in the inductive rule, when element (p',q',r') is removed from O, it is added to C with a child tree. Therefore the set of edges as the union of edges in O and in edge elements in C always stays the same. □

Lemma 2. *For attack trees* $T = ((p,q,r), O, C)$ *atree and* $T' = ((p',q',r'), O', C')$ *atree with* $(p',q',r') \in open(T)$, *there exists a tree* $S = ((p,q,r), O'', C'')$ *with* $open(S) = (open(T) \setminus \{(p',q',r')\}) \cup open(T')$.

Proof. By induction on the attack tree T:

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

2. $T = ((p,q,r), O, C \cup T'')$: Then T was created from $T''' = ((p,q,r), O' \uplus \{(p'',q'',r'')\}, C)$ atree $T'' = ((p'',q'',r''), O'', C'')$ atree. As $open(T) = (open(T'') \setminus \{(p'',q'',r'')\} \cup open(T''')$, we could have the cases

   a) $(p',q',r') = (p'',q'',r'')$:

   b) $(p',q',r') \in open(T''')$: By induction hypothesis from T''' and T' we get a tree $S' = ((p,q,r), O''', C''')$ atree with $open(S') = (open(T''') \setminus \{(p',q',r')\}) \cup open(T') = (O' \uplus \{(p'',q'',r'')\} \cup (\bigcup_{T' \in C} open(C))) \{(p',q',r') = (O' \cup (\bigcup_{T' \in C} open(C))) \{(p',q',r')\} \uplus \{(p'',q'',r'')\}$. We can then combine S' and T'' to $S = ((p,q,r), O', C \cup S')$ atree with

   c) $(p',q',r') \notin open(T''')$: Then $(p',q',r') \in open(T'')$. By induction hypothesis from T'' and T' we get a tree $S' = ((p'',q'',r''), O''', C''')$ atree with $open(S') = (open(T'') \setminus \{(p',q',r')\}) \cup open(T')$. Then we can combine T''' and S' to $S = ((p,q,r), O', C \cup S')$ atree with $open(S) = O' \cup (\bigcup_{T' \in C} open(C)) \cup open(S') = O' \cup (\bigcup_{T' \in C} open(C)) \cup ((open(T'') \setminus \{(p',q',r')\}) \cup open(T')) = (O' \cup (\bigcup_{T' \in C} open(C)) \cup (open(T'')) \setminus \{(p',q',r')\}) \cup open(T')) = (open(T) \setminus \{(p',q',r')\}) \cup open(T'))$

   d) $(p',q',r') \in open(T''')$

□

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

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

Proof. $\Longrightarrow$: Assume $p \leq_m q$. Then there is a refinement relation $\mathcal{R}$. Let $T = ((p,q,r), O, C)$ atree be a closed attack tree. As $open(T) = \{(p',q') \mid \exists r' : (p',q',r') \in O\} \cup \bigcup_{T' \in C} open(T') = \emptyset$, we have $O = \emptyset$. and therefore $|O \sqcup C| = |C|$. We show by induction on $|C|$ that $(p,q) \notin R$:

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

2. $|C| \geq 0$: Then as $open(T) = \bigcup_{T' \in C} open(T') = \emptyset$, we have for every $T' = ((p',q',r'), O', C') \in C$ that $open(T') = \emptyset$ and by induction hypothesis $(p',q') \notin \mathcal{R}$. Then there is an attacking rule r, but for every defending rule r' leading to (p',q') we have $(p',q') \in \mathcal{R}$, therefore $(p,q) \notin \mathcal{R}$.

So for (p, q) there is no attack tree. $\Leftarrow$: Assume $\neg\exists T : root(T) = (p, q) \wedge closed(T)$ 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 $(p, q, r) \in Att$ with $(p, q) \in \mathcal{R}$, by inference rule 1 there exists $T = ((p, q, r), O, C)$ atree. From all such T, choose the one where O is minimal with regard to the inclusion order. There exists $(p', q', r') \in O : (p', q') \in \mathcal{R}$, because otherwise there would be a closed attack tree $T' = ((p', q', r'), O', C')$ atree and with this tree by inference rule 2 we would get $((p, q, r), O'', C \cup T')$ with $O'' = O \setminus \{(p', q', r')\} \subsetneq O$ in contradiction to the minimality of O. So for the attacking transition (p, q, r) there is a defending transition (p', q', r') with $(p', q') \in \mathcal{R}$.

With this refinement relation we have $p \leq_m q$. □

2.7 Visibly pushdown automaton

Definition 9 (Visibly pushdown automaton)**.** A PRS is a visibly pushdown automaton (vPDA) if all processes are sequential and there is a partition $Act = Act_r \uplus Act_i \uplus Act_c$ such that each rule $(p, a, p') \in \Delta$ has the form

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

The modal extension for a *modal visibly pushdown automaton (mvPDA)* is straightforward.

Definition 10 (Attack rules for mvPDA)**.** Let $(\Delta_{\text{may}}, \Delta_{\text{must}})$ be an mvPDA. We define a *attack rules* $(p, q) \longrightarrow_b S$ obtainable from the rewrite rules. For every $p, q \in \mathcal{P}$, we have:

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

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

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

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

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

Then we see that rules 3 and 4 always combine a rule where on the left-hand side, there is (p', q') in S with $|p'| = 2$ or $|p'| = 3$, while on the right-hand side we require for all $(p'', q'') \in S'$ that $|p''| = 1$. Therefore we will call an attack rule $(p, q) \longrightarrow_b S$ a *right-hand side* rule if $\forall (p', q') \in S : |p'| = 1$ and otherwise a *left-hand side* rule.

Lemma 3. *Given an MTS generated by a mvPDA, for an attack tree $((p, q, r), O, C)$ atree with $open(T) = S$ and any $s, t \in \mathcal{P}$, there is also a tree $T' = ((p \cdot s, q \cdot t, r'), O', C')$ atree with $open(T') = \{(p' \cdot s, q' \cdot t, r'') \mid (p', q', r') \in open(T)\}$.*

Proof. By the MTS induction rules, we have that for every $p \overset{a}{\dashrightarrow} p'$ is generated from a $(p, a, p') \in \Delta_{\text{may}}$ and for a mvPDA therefore $|p| = 2$. Then there is only one transition from $p \cdot s$, nameley $p \cdot s \overset{a}{\dashrightarrow} p' \cdot s$ generated by the MTS induction rule 1. Also for every $q \overset{a}{\longrightarrow} q'$ there is just $q \cdot t \overset{a}{\dashrightarrow} q' \cdot t$ from $q \cdot t$.

Then it is a single transition created from $p \overset{a}{\dashrightarrow} p'$ with $S = \{(p', q') \mid q \overset{a}{\dashrightarrow} q'\}$ and we get $p \cdot s \overset{a}{\dashrightarrow} p' \cdot s$ and $\{(p' \cdot s, q' \cdot t) \mid q \cdot t \overset{a}{\dashrightarrow} q' \cdot t\} = S'$. If the transition was created from $q \overset{a}{\longrightarrow} q'$ with $S = \{(p', q') \mid p \overset{a}{\longrightarrow} p'\}$ and we get $\{(p' \cdot s, q' \cdot t) \mid p \cdot t \overset{a}{\longrightarrow} p' \cdot t\} = S'$ Both cases yield $(p \cdot s, q \cdot t) \longrightarrow_a^* S'$. □

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

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

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

the linear form of a derivation of the attack sequence. Always a_1 has the form $(P \cdot S, Q \cdot R) \longrightarrow_a S$ and a_n the form $(p, q) \longrightarrow_a \emptyset$.

Our proposition is that if we can split up the sequence into subsequences which we can all compute seperately, we can also compute the whole sequence. More formally, we want to show that if there is a set of $k + 1$ indices $I = i_0, \ldots, i_k$ where

1. $0 = i_0 < i_1 < i_2 < \ldots < i_{k-1} < i_k = n$.

2. There is a sequence $(b_1, \ldots, b_k)$ where each b_i is an attack rule $(p, q) \longrightarrow_b S$.

3. For every $i, j \in I$ with $i < j$ the sequence $(a_{i+1}, ..., a_j)$, which generates the attack sequence $(p, q) \longrightarrow_a^* S$, the representing rule b_j is $(p, q) \longrightarrow_b S$ If $j < n$ then with $b_i = (p', q') \longrightarrow_b S'$ we require $(p', q') \in S'$. and if $j = n$ then $S = \emptyset$.

there is $(P \cdot S, Q \cdot T) \longrightarrow_b \emptyset$.

We prove this by induction on the number k:

1. $k = 1$: Then the indices are $0, n$ and the rule sequence (b_1) represents $(a_1, ..., a_n)$ generating $(P \cdot S, Q \cdot T) \longrightarrow_a^* \emptyset$. Then we have $(P \cdot S, Q \cdot T) \longrightarrow_b \emptyset$.

2. $k > 1$, and the induction hypothesis holds for any $k' < k$: Let $(b_1, ..., b_k)$ be the rule sequence. For the first rule $b_1 = (P \cdot S, Q \cdot T) \longrightarrow_b S$, there is be $(p', q') \in S$ with $|p'| = |q'| \geq 2$ because otherwise no more rules could be applied afterwards, in contradiction to $k > 1$. So this is a left-hand side rule. Also the last rule $b_k = (p', q') \longrightarrow_b \emptyset$ is a right-hand side rule.

$\Leftarrow$: We show that for $(p, q) \longrightarrow_b S$, there is a tree $T = ((p, q, r), O, C)$ atree such that $open(T) = S$. by induction on the inference of $(p, q) \longrightarrow_b S$:

1. If it was created by rule 1 or 2. If it was created from $(p, a, p') \in \Delta_{\text{may}}$, then there is an attacking transition $r = p \overset{\text{a}}{\dashrightarrow} p'$ and for every $(q, a, q') \in \Delta_{\text{may}}$ there is one induced defending transition $r' = q \overset{\text{a}}{\dashrightarrow} q'$. If it was created from $(q, a, q') \in \Delta_{\text{must}}$, then there is an attacking transition $r = p \overset{\text{a}}{\dashrightarrow} p'$ and for every $(q, a, q') \in \Delta_{\text{may}}$ there is one induced defending transition $r' = q \overset{\text{a}}{\dashrightarrow} q'$.

   Therefore $states(def(p, q, r)) = S$ and get the tree $T = ((p, q, r), def(p, q, r), \emptyset)$ atree with $open(T) = S$.

2. It was created by rule 3 from $(p, q) \longrightarrow_b \{(p' \cdot P, q' \cdot Q)\} \uplus S''$ and $(p', q') \longrightarrow_b S'$ 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 $T' = ((p', q', r'), O', C')$ atree with $open(T') = S'$ and a tree $T'' = ((p, q, r), O, C)$ atree with $open(T'') = S'' \uplus \{(p' \cdot P, q' \cdot Q, r'')\}$.

   By lemma 3 there is also a tree $T''' = ((p' \cdot P, q' \cdot Q, r''), O'', C'')$ atree with $open(T''') = O''' \uplus \{(p' \cdot P, q' \cdot Q, r'')\}$ and $O''' = \{(p'' \cdot P, q'' \cdot Q, r''') \mid (p'', q'', r'') \in S'\} = S'''$.

   By lemma 2 we then get a tree $T = ((p, q, r), O'''', C'''')$ with $open(T) = S'' \cup S''' = S$.

3. It was created by rule 4 from $(p, q) \longrightarrow_b \{(p', q')\} \uplus S''$ and $(p', q') \longrightarrow_b S'$ with $S = S'' \cup S'$. Then by induction hypothesis there is a tree $T' = ((p', q', r'), O', C')$ atree with $open(T') = S'$ and a tree $T'' = ((p, q, r), O, C)$ atree with $open(T'') = S'' \uplus \{(p', q')\}$. By lemma 2 we then get a tree $T = ((p, q, r), O'', C'')$ with $open(T) = S'' \cup S' = S$.

Then if $(P \cdot S, Q \cdot T) \longrightarrow_b \emptyset$ we have a tree $T = (P \cdot S, Q \cdot T, r), O, C)$ with $open(T) = \emptyset$. □

3 Algorithms

3.1 Description

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 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 VPDAREFINEMENT($P \cdot S, Q \cdot T, mvPDA$)        ▷ $P \cdot S \leq_m Q \cdot T$ given $mvPDA$
    $initial \leftarrow [P \cdot S, Q \cdot T]$
    $rules \leftarrow$ ATTACKRULES($mvPDA$)
    while $\exists lhsRule, rhsRule \in rules :$ COMBINE$(lhsRule, rhsRule) \not\subseteq rules$ do
        $rules \leftarrow rules \cup$ COMBINE$(lhsRule, rhsRule)$
    end while
    return $(initial, \emptyset) \in rules$
end function
```

3.2 Soundness and completeness

3.2.1 Soundness

3.2.2 Completeness

3.3 Runtime

3.4 Optimizations

3.5 Performance evaluation

3.6 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 $Q \cdot S$

$$\begin{tikzcd}
& \boxed{(p\cdot M\cdot S, q\cdot C\cdot S)} \arrow[r, "q\cdot C \overset{\text{coin}}{\dashrightarrow} q\cdot C\cdot C"] & \boxed{(p\cdot M\cdot M\cdot S, q\cdot C\cdot C\cdot S)} \arrow[r, "q\cdot C \overset{\text{tea}}{\dashrightarrow} q"] & \boxed{(T\cdot M\cdot S, q\cdot C\cdot S)} \\
& p\cdot M \overset{\text{coin}}{\dashrightarrow} p\cdot M\cdot M & p\cdot M \overset{\text{tea}}{\dashrightarrow} T & q\cdot C \overset{\text{coffee}}{\longrightarrow} q \\
\boxed{(p\cdot S, q\cdot S)} \arrow[uur, "q\cdot S \overset{\text{coin}}{\dashrightarrow} q\cdot C\cdot S"] \arrow[ddr, "q\cdot S \overset{\text{coin}}{\dashrightarrow} q\cdot T\cdot S"] & & & \\
p\cdot S \overset{\text{coin}}{\dashrightarrow} p\cdot M\cdot S & & & \\
& \boxed{(p\cdot M\cdot S, q\cdot T\cdot S)} \arrow[r, "q\cdot T \overset{\text{coin}}{\dashrightarrow} q\cdot T\cdot T"] & \boxed{(p\cdot M\cdot M\cdot S, q\cdot T\cdot T\cdot S)} \arrow[r, "q\cdot T \overset{\text{coffee}}{\dashrightarrow} q"] & \boxed{(c\cdot M\cdot T, q\cdot T\cdot S)} \\
& p\cdot M \overset{\text{coin}}{\dashrightarrow} p\cdot M\cdot M & p\cdot M \overset{\text{coffee}}{\dashrightarrow} c & q\cdot T \overset{\text{tea}}{\longrightarrow} q
\end{tikzcd}$$

Figure 3.6: Tree for winning strategy

$$\begin{tikzcd}
& \boxed{(p\cdot M, q\cdot C) \longrightarrow \{(p\cdot M\cdot M, q\cdot C\cdot C)\}} \arrow[r, "\text{call}"] & \boxed{(p\cdot M, q\cdot C) \longrightarrow \{(T, q)\}} \arrow[r, "\text{return}"] & \boxed{(T\cdot M, q\cdot C) \longrightarrow \emptyset} \\
\boxed{(p\cdot S, q\cdot S) \longrightarrow \{(p\cdot M\cdot S, q\cdot C\cdot S), (p\cdot M\cdot S, q\cdot T\cdot S)\}} \arrow[ur, "\text{call}"] \arrow[dr, "\text{call}"] & & & \\
& \boxed{(p\cdot M, q\cdot T) \longrightarrow \{(p\cdot M\cdot M, q\cdot T\cdot T)\}} \arrow[r, "\text{call}"] & \boxed{(p\cdot M, q\cdot T) \longrightarrow \{(c, q)\}} \arrow[r, "\text{return}"] & \boxed{(c\cdot M, q\cdot T) \longrightarrow \emptyset}
\end{tikzcd}$$

Figure 3.7: Tree for winning strategy with attack rules

$$\begin{tikzcd}
& \boxed{(p\cdot M, q\cdot C) \longrightarrow \{(T\cdot M, q\cdot C)\}} \arrow[r, "\text{internal}"] & \boxed{(T\cdot M, q\cdot C) \longrightarrow \emptyset} \\
\boxed{(p\cdot S, q\cdot S) \longrightarrow \{(p\cdot M\cdot S, q\cdot C\cdot S), (p\cdot M\cdot S, q\cdot T\cdot S)\}} \arrow[ur, "\text{call}"] \arrow[dr, "\text{call}"] & & \\
& \boxed{(p\cdot M, q\cdot T) \longrightarrow \{(c\cdot M, q\cdot T)\}} \arrow[r, "\text{internal}"] & \boxed{(c\cdot M, q\cdot T) \longrightarrow \emptyset}
\end{tikzcd}$$

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

$$\begin{tikzcd}
& \boxed{(p\cdot M, q\cdot C) \longrightarrow \emptyset} \\
\boxed{(p\cdot S, q\cdot S) \longrightarrow \{(p\cdot M\cdot S, q\cdot C\cdot S), (p\cdot M\cdot S, q\cdot T\cdot S)\}} \arrow[ur, "\text{call}"] \arrow[dr, "\text{call}"] & \\
& \boxed{(p\cdot M, q\cdot T) \longrightarrow \emptyset}
\end{tikzcd}$$

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

$$\boxed{(p\cdot S, q\cdot S) \longrightarrow \emptyset}$$

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.