DOI 10.1007/s00165-008-0081-8
BCS © 2008
Formal Aspects of Computing (2008) 20: 407–428

Formal Aspects of Computing

Reachability analysis of fragments of mobile ambients in AC term rewriting

Giorgio Delzanno and Roberto Montagna

Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, via Dodecaneso 35, 16143 Genova, Italy.
E-mail: giorgio@disi.unige.it

Dedicated to the memory of Prof. Nadia Busi

Abstract. In this paper, we investigate the connection between fragments of associative-commutative Term Rewriting and fragments of Mobile Ambients, a powerful model for mobile and distributed computations. The connection can be used to transfer decidability and undecidability results for important computational properties like reachability from one formalism to the other. Furthermore, it can be viewed as a vehicle to apply tools based on rewriting for the simulation and validation of specifications given in Mobile Ambients.

Keywords: Mobile ambients; Reachability; Term rewriting; Petri nets

1. Introduction

Models for mobile and distributed computation like the Mobile Ambients (MA) of Cardelli and Gordon [CG00] provide mechanisms for moving collections of processes across a network. The basic block of the MA model is the notion of ambient. Each ambient has a name, a collection of local agents, and a collection of sub-ambients. Local agents model the possible computations that can take place inside the ambient. Differently from standard process algebraic languages, MA agents have movement capabilities as well as primitives to dissolve the boundary of an ambient. Several variations of MA have been proposed in the literature. For instance, the Safe Ambients of [LS00] and the Boxed Ambients of [BCC01] provide more sophisticated mechanisms for controlling the access to an ambient. In this paper, we restrict our attention to *pure* and *public* versions of MA-like calculi. The pure public fragment of MA (ppMA) is obtained by forbidding the use of name restriction and of communication between local agents, two features inherited from the π-calculus. The ppMA fragment allows us to focus our attention on ambients and movement operations, the novel features with respect to the π-calculus. As shown in [MP05], ppMA is still Turing complete.

From an abstract point of view, the movement mechanisms of MA-like calculi can be viewed as update schemes for dynamically changing tree structures. Following this abstract view, in this paper, we investigate the connection between fragments of MA and associative-commutative (AC) Term Rewriting, a natural operational model for manipulating unordered finite trees. To be as close as possible to the MA syntax, we consider terms built via a constructor for representing parallel composition, a constructor for representing ambients (internal nodes), and a finite set of constants for representing local agents (leaves). We refer to this class of AC Term Rewriting as Tree Update Calculus (TUC).

The connection between the two computational models can be used to transfer properties from one formalism to the other. In this paper, we focus our analysis on the *reachability problem* for fragments and variations of ppMA.

Correspondence and offprint requests to: G. Delzanno, E-mail: giorgio@disi.unige.it

For two processes P_0 and P_1, the reachability problem consists in checking whether there exists a reduction from P_0 to P_1. The same decision problem can be formulated for a set of rewrite rules and two fixed ground terms t_0 and t_1.

As a first analysis, we show how to express a reachability problem between ppMA-processes as a reachability problem between terms in TUC. Resorting to results on ppMA [BT05, MP05], we can exploit the encoding to show that the reachability problem is undecidable in TUC. This negative result motivates further investigations in search of fragments of TUC that can still model ambients and movement operations and in which computational problems like reachability are decidable.

In this paper, we focus our attention on *structure preserving* rewriting rules (TUC^{sp}). Intuitively, a structure preserving rule is such that its application never removes ambients (internal nodes) of a tree, while it can produce and consume leaves without any limit. Thus, structure preserving rewrite rules are monotonic with respect to the number of internal nodes of terms, but not with respect to their size. By exploiting this property, we show that TUC^{sp}-reachability is decidable via a reduction to a reachability problem for a Petri net. The latter problem is known to be decidable [May84, Rao82]. It is important to remark that the TUC^{sp} fragment is not directly related to other fragments of Term Rewriting for which reachability is known to be decidable as those studied in [CDFV91, Jac96, Mayr00, Sal88].

By exploiting again the encoding from ppMA to TUC, we show that our decidability result for TUC^{sp} generalizes in an elegant way those proved in [BT05, BZ05a] for reachability in fragments of ppMA. Specifically, we show that the semantic restrictions for replication proposed in [BT05] (weak reduction) and the syntactic restrictions proposed in [BZ05a] (guarded replication) can be captured in a uniform way using the syntactic restrictions of TUC^{sp}. Furthermore, we apply our result to prove decidability of reachability for fragments of Safe and Boxed Ambients similar to those mentioned in [BT05, BZ05b]. It is important to remark that TUC^{sp} can express more general tree update schemes than those provided by the MA-fragments studied in [BT05, BZ05a, BZ05b]. Thus, TUC and TUC^{sp} can be viewed as operational models in which to reason on extensions of languages inspired to the Mobile Ambients model.

Plan of the paper In Sect. 2 we introduce ppMA. In Sect. 3 we introduce TUC. In Sect. 4 we encode the ppMA-reachability problem in reachability in TUC. In Sect. 5 we define the fragment TUC^{sp}. In Sect. 6 we study the decidability of reachability in TUC^{sp}. In Sect. 7 we study the relation between TUC^{sp} and fragments of ppMA. In Sect. 8 we discuss related work. Finally, in Sect. 9 we address some conclusions. In Appendix A we recall the main definition of Petri nets.

2. ppMA: Pure public mobile ambients

In this paper, we consider the pure (without communication) public (without name restriction) fragment of MA (ppMA for short), studied in [BT05, BZ05a, MP05], in which processes comply with the following grammar:

$$\begin{array}{lcl} P & ::= & \mathbf{0} \mid n[P] \mid M.P \mid P|P \mid !P \\ M & ::= & in\ n \mid out\ n \mid open\ n \end{array}$$

where n ranges over a denumerable set Amb of ambient names. The process $n[P]$ denotes an ambient with public name n. The process $M.P$ denotes sequential composition (action prefixing), while $P|Q$ denotes the parallel composition of P and Q. The replication $!P$ denotes an arbitrary number of parallel copies of P. Finally, $\mathbf{0}$ denotes the null process. Since ambients can be nested, the spatial structure of a process can be viewed as an unordered tree (with arbitrary width). Movement operators may change the ambient spatial structure of processes. The meaning of the operators becomes clearer by looking at the operational semantics defined in terms of a structural congruence $\equiv$ and of a reduction relation $\twoheadrightarrow$. The structural congruence $\equiv$ is the smallest one satisfying

$$P \mid Q \equiv Q \mid P \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R \quad P \mid \mathbf{0} \equiv P \quad !P \equiv !P \mid P$$

The reduction relation $\twoheadrightarrow$ is defined in Fig. 1. We use $\overset{*}{\twoheadrightarrow}$ to denote the reflexive and transitive closure of the relation $\twoheadrightarrow$.

$$\begin{array}{rcl} open\ m.P \mid m[Q] & \twoheadrightarrow & P \mid Q \\ n[in\ m.P \mid Q] \mid m[R] & \twoheadrightarrow & m[n[P \mid Q] \mid R] \\ m[n[out\ m.P \mid Q] \mid R] & \twoheadrightarrow & n[P \mid Q] \mid m[R] \end{array}$$

$$\frac{P \twoheadrightarrow Q}{P \mid R \twoheadrightarrow Q \mid R} \qquad \frac{P \twoheadrightarrow Q}{n[P] \twoheadrightarrow n[Q]} \qquad \frac{P' \equiv P \quad P \twoheadrightarrow Q \quad Q \equiv Q'}{P' \twoheadrightarrow Q'}$$

Fig. 1. Reduction semantics for ppMA

Definition 2.1 Given two ppMA processes P and Q, the *reachability problem* consists in deciding whether $P \overset{*}{\twoheadrightarrow} Q$ holds or not.

Example 2.1 Assume $P_0 = !P|pc[Q]$ and $P = trojan[in\ pc.virus[out\ trojan.P']]$. By using the congruence on $!P$ and the in capability, a copy of P can move inside the ambient pc yielding the process $P_1 = !P|pc[trojan[virus[out\ trojan.P']]|Q]$. By using the out capability, the $virus$ agent can now be released inside the ambient pc leading to $P_2 = !P|pc[trojan[\mathbf{0}]|virus[P']|Q]$. Thus, $P_0 \overset{*}{\twoheadrightarrow} P_2$ holds.

2.1. Safe and boxed ambients

In the Safe Ambients (SA) model [LS00] movement capabilities have co-capabilities used to regulate the access to an ambient. For instance, the SA reduction rule for the in capability is defined as:

$$n[in\ m.P \mid Q] \mid m[\overline{in}\ m.P' \mid R] \twoheadrightarrow m[n[P \mid Q] \mid P' \mid R]$$

i.e., ambient n enters m only if m grants the access to n using $\overline{in}\ m$.
In the Boxed Ambients (BA) model [BCC01] the open capability is replaced by a parent-child communication mechanism. As an example, let $(x)^{\uparrow}P$ denote a process ready to receive a sequence of capabilities from a process in the parent ambient (x is a variable that may occur in P), and $\langle M\rangle.R$ denote a process ready to send M to a child ambient. Then, the following reduction rule models parent-child communication:

$$n[(x)^{\uparrow}P \mid Q] \mid \langle M\rangle.R \twoheadrightarrow n[P\{x := M\} \mid Q] \mid R$$

where $P\{x := M\}$ denotes the substitution of x with M in P modulo α-renaming of bounded variables.

3. TUC: a fragment of AC term rewriting

In this section, we define a fragment of AC rewriting, called Tree Update Calculus (TUC), that can be viewed as a generalization of the tree update schemes underlying ppMA.
Given two *finite* sets of constants $\mathcal{N}$ and $\mathcal{Q}$ with $\mathcal{N} \cap \mathcal{Q} = \emptyset$, we use a constructor $n\langle \ldots \rangle$ to represent an ambient (internal node) with label $n \in \mathcal{N}$, an AC constructor | to build multisets of trees (the multiset of sons of an internal node), ϵ to represent the empty multiset, and the finite set of constants in $\mathcal{Q}$ to represent processes (leaves). For instance, given $\mathcal{N} = \{n, m\}$ and $\mathcal{Q} = \{a, b\}$ the term $n\langle a \mid a \mid n\langle\epsilon\rangle \mid m\langle a \mid b\rangle\rangle$ can be viewed as an abstract representation of an ambient n with two sub-processes of type a and two sub-ambients. Since ambients can be dynamically populated, we keep terms like $n\langle\epsilon\rangle$ (the empty ambient) distinguished from leaves in $\mathcal{Q}$. Formally, given a denumerable set of variables ranging over multiset of terms (multiset-variables) $\mathcal{V} = \{X, Y, \ldots\}$ the sets TR (tree terms) and MS (multiset terms) are the least sets satisfying:

- The empty multiset ϵ is in MS, every multiset-variable $X \in \mathcal{V}$ is in MS;
- Every constant in $\mathcal{Q}$ is in TR (leaf);
- If $m \in MS$ and $n \in \mathcal{N}$, then $n\langle m\rangle \in TR$ (internal node);
- If $t \in TR$, then $t \in MS$ (singleton multiset);
- If $m, m' \in MS$, then $m \mid m' \in MS$ (parallel composition).

A *ground* term is a term without variables. Notice that, with a little bit of overloading, we use the same notation for a term t and the singleton multiset containing t. The multiset constructor | is associative and commutative, i.e., $m_1 \mid (m_2 \mid m_3) = (m_1 \mid m_2) \mid m_3$, and $m_1 \mid m_2 = m_2 \mid m_1$ for $m_1, m_2, m_3 \in MS$. Furthermore, $m \mid \epsilon = m$ for any $m \in MS$.
From here on, we use the special symbol tuc to represent a forest $t_1 \mid \ldots \mid t_n$ as a single tree term $tuc\langle t_1 \mid \ldots \mid t_n\rangle$. We always assume that tuc never occurs in $t_1, \ldots, t_n$. The use of terms of the form $tuc\langle t_1 \mid \ldots \mid t_n\rangle$ simplifies the presentation and the proofs (we always work on trees).
Furthermore, we use the syntax $t[\ ]$ to indicate a tree term with one occurrence of the constant $\circ$, and $t[s]$ to indicate the term obtained by replacing the constant $\circ$ in $t[\ ]$ with s. We use $var(t)$ to denote the set of variables in t.
A TUC-rewrite rule $l \rightarrow r$ is such that $l, r \in MS$, $var(r) \subseteq var(l)$, the label tuc does not occur in l, r and every variable in l occurs within a tree term.[1] A TUC-theory $\mathcal{R}$ is a set of TUC rewrite rules.
The rules in the theory describe how the current tree configuration is updated during a computation in accord with the following operational semantics. For a fixed set of rules $\mathcal{R}$ and given two ground TR terms $t_1 = tuc\langle m_1\rangle$ and $t_2 = tuc\langle m_2\rangle$, $t_1 \Rightarrow t_2$ if and only if there exists a context $t[\ ]$, a rule $l \rightarrow r$ in $\mathcal{R}$, and a mapping σ from $var(l)$ to terms in MS (multiset of trees) such that $t_1 = t[\sigma(l)]$ and $t_2 = t[\sigma(r)]$. For a set of rules $\mathcal{R}$, we use $\overset{*}{\Rightarrow}$ to indicate the reflexive and transitive closure of $\Rightarrow$.

Definition 3.1 For a set of TUC rules $\mathcal{R}$ and two ground terms $t_1 = tuc\langle m_1\rangle$ and $t_2 = tuc\langle m_2\rangle$, the reachability problem consists in deciding if $t_1 \overset{*}{\Rightarrow} t_2$ holds.[2]

Example 3.1 Assume $\mathcal{N} = \{n\}$, $\mathcal{Q} = \{a, b\}$, and let $\mathcal{R}$ consist of the rules

$$\begin{array}{rcl} a & \rightarrow & a \mid a \\ a \mid a & \rightarrow & a \\ n\langle X\rangle & \rightarrow & n\langle a \mid n\langle X\rangle\rangle \end{array}$$

The first rule adds one occurrence of leaf a, while the second rule consumes one occurrence of a. The third rule inserts a new internal node n with leaf a on top of an existing n-rooted tree. The multiset-variable X represents the content of the node labelled by n. With these rules, the term $t = tuc\langle a \mid n\langle b\rangle\rangle$ can be rewritten into trees of arbitrary width and depth as shown by the following derivation:

$$t \overset{*}{\Rightarrow} tuc\langle a \mid a \mid n\langle a \mid n\langle b\rangle\rangle\rangle \overset{*}{\Rightarrow} tuc\langle a \mid n\langle a \mid a \mid n\langle a \mid n\langle b\rangle\rangle\rangle\rangle \overset{*}{\Rightarrow} \cdots$$

As another example, we can use the rule $n\langle X\rangle \rightarrow X$ to remove an internal node by moving its sons one level up in the tree, while the rule $n\langle X\rangle \rightarrow a$ replaces an entire n-rooted subtree with the leaf a.

In the next section, we will show that TUC is powerful enough to express reachability problems for ppMA.

4. From reachability in ppMA to reachability in TUC

In this section, we show how to reduce the reachability problem for ppMA to reachability in TUC. Before going into the details of the reduction, we need however some preliminary considerations on the semantics of MA. Let us first notice that we can work with a congruence relation applied only to context different from $!P$ (as for the reduction semantics). Let us now reformulate the axioms $P \mid \mathbf{0} \equiv P$ and $!P \equiv P \mid !P$ as the following reduction rules

$$P \twoheadrightarrow P \mid \mathbf{0} \quad P \mid \mathbf{0} \twoheadrightarrow P \quad !P \twoheadrightarrow !P \mid P \quad !P \mid P \twoheadrightarrow !P$$

In MA the empty ambient is represented by $n[\mathbf{0}]$. To maintain this property, we refine the reduction semantics of the out rule as follows

$$n[m[out\ n.P \mid R] \mid Q] \twoheadrightarrow m[P \mid R] \mid n[Q \mid \mathbf{0}]$$

[1] With this condition, we forbid rules like $X \rightarrow t$ where $X \in \mathcal{V}$.
[2] We assume that the label tuc does not occur in m_1 and m_2. We recall that, to simplify the proofs, the special symbol tuc is introduced to represent a forest $t_1 \mid \ldots \mid t_n$ as a single tree $tuc\langle t_1 \mid \ldots \mid t_n\rangle$.

$$\begin{array}{ll}
(open) & q_{open\ n.Q} \mid n\langle Y\rangle \ \rightarrow\ T(Q) \mid Y \\
(in) & m\langle q_{in\ n.Q} \mid Y\rangle \mid n\langle Z\rangle \ \rightarrow\ n\langle m\langle T(Q) \mid Y\rangle \mid Z\rangle \\
(out) & n\langle m\langle q_{out\ n.Q} \mid Y\rangle \mid Z\rangle \ \rightarrow\ m\langle T(Q) \mid Y\rangle \mid n\langle q_{\mathbf{0}} | Z\rangle \\
(copyt) & q_{!Q} \rightarrow q_{!Q} \mid T(Q) \\
(absorbt) & q_{!Q} \mid T(Q) \rightarrow q_{!Q} \\
(zero) & q \rightarrow q \mid q_{\mathbf{0}} \qquad q \mid q_{\mathbf{0}} \rightarrow q \\
 & n\langle X\rangle \rightarrow n\langle X\rangle \mid q_{\mathbf{0}} \qquad n\langle X\rangle \mid q_{\mathbf{0}} \rightarrow n\langle X\rangle \\
 & \qquad \text{For any } n, m \in \mathcal{N},\ q_{open\ n.Q}, q_{in\ n.Q}, q_{out\ n.Q}, q_{!Q} \in \mathcal{Q}
\end{array}$$

Fig. 2. TUC theory for ppMA

Several computation steps of the modified semantics may correspond to one computation or congruence step in the original semantics. Reachability is preserved by the modified semantics: If Q is reachable from P_0 in the standard semantics, then there exists Q' reachable from P_0 in the modified semantics such that Q' is equivalent modulo the congruences for $\mathbf{0}$ to Q, and Q' is obtained by replacing every occurrences of a process $!R$ in Q with an equivalent process $!R'$ occurring in P_0. Given a process term P, let us now define the set of replicated or sequential processes $Der(P)$ that may become active during a computation (the *derivatives* of P).

$$\begin{aligned}
Der(\mathbf{0}) &= \{\mathbf{0}\} \\
Der(!P) &= \{!P\} \cup Der(P) \\
Der(M.P) &= \{M.P\} \cup Der(P) \\
Der(n[P]) &= Der(P) \\
Der(P \mid Q) &= Der(P) \cup Der(Q)
\end{aligned}$$

It is easy to check that $Der(P)$ is a finite set. Furthermore, if $P \overset{*}{\twoheadrightarrow} Q$ using the modified reduction semantics, then $Der(Q) \subseteq Der(P) \cup \{\mathbf{0}\}$.

Let us now consider the reachability problem $P_0 \overset{*}{\twoheadrightarrow} P_1$. To encode it into TUC, we use tree terms in which internal nodes have labels in $\mathcal{N} = Amb$, the set of ambient names, and leaves range over the finite set of constants

$$\mathcal{Q} = \{q_R \mid R \in Der(P_0)\} \cup \{q_{\mathbf{0}}\}$$

Given a process Q derived from P_0, we define the ground term $T(Q)$ by induction on Q as follows:

$$\begin{aligned}
T(\mathbf{0}) &= q_{\mathbf{0}} \\
T(!Q_1) &= q_{!Q_1} \\
T(M.Q_1) &= q_{M.Q_1} \\
T(n[Q_1]) &= n\langle T(Q_1)\rangle \\
T(Q_1 \mid Q_2) &= T(Q_1) \mid T(Q_2)
\end{aligned}$$

Notice that we have chosen here to give a lazy definition of the mapping T, i.e., its application is not propagated inside labels in the rule for sequential composition. With our definition, TUC-constants are labelled by MA-expressions. An eager definition of the mapping of sequential composition could be $T(M.Q_1) = q_{M.T(Q_1)}$. This definition requires the definition of a new set of labels obtained by combining expressions of process algebra and terms of TUC. However, the lazy definition makes the presentation and the proofs simpler than the eager one.

The semantics of MA processes can then be simulated by the TUC-theory $\mathcal{R}(P_0)$ of Fig. 2. To make this claim formal, we need some preliminary definitions and some related properties.

Let us first notice that the reduction semantics can be defined by applying the reduction rules for !, $\mathbf{0}$, *open*, *in*, and *out* in a context with a hole $C[]$ defined by induction on the structure of processes, i.e., if $L \twoheadrightarrow R$, then $C[L] \twoheadrightarrow C[R]$ for any such context $C[]$.

The following property then holds.

Proposition 4.1 If $P_0 \twoheadrightarrow^* P_1$ then $tuc\langle T(P_0)\rangle \Rightarrow^* tuc\langle T(P_1)\rangle$ in $\mathcal{R}_{P_0}$.

Proof In order to make things work we have to prove the stronger property: if $P_0 \twoheadrightarrow^* P'$, then $T(P_0) \Rightarrow^* T(P')$. The proof is by induction of the length $L \geq 0$ of the derivation $P_0 \twoheadrightarrow^* P'$.
If $L = 0$ the thesis immediately holds.

Suppose that the thesis holds for derivation of lengths $L \geq 0$. Let us call this hypothesis IH_1. Consider now a derivation

$$P_0 \twoheadrightarrow^L P_1 \twoheadrightarrow P'$$

of length $L+1$. Thus, one of the basic reduction step is applied at some depth in P_1. The depth of a redex P of a MA-reduction is zero in the process P. If P has depth D in Q, then $P \mid Q$ has still depth D. If P has depth D in Q, then P has depth $D+1$ in $n\langle Q\rangle$ We assume here that P denotes a specific subexpression of a term Q. This can be made more precise by indexing all occurrences of subexpressions. We omit this to simplify the presentation. We proceed then by induction on the depth D at which a reduction is applied.
If $D = 0$ then a reduction step is applied at processes occurring at the top level in P_1. We now perform a case analysis on the type of reduction that can be applied to P_1.

1. Suppose that $P_1 = open\ n.P \mid n[R] \mid S$ and that we apply a reduction step for capability $open\ n$ obtaining the process $P' = P \mid R \mid S$.
   By definition,
   $$T(P_1) = q_{open\ n.P} \mid n\langle T(R)\rangle \mid T(S)$$
   By applying the TUC rule that defines the state $q_{open\ n.P}$,
   $$q_{open\ n.P} \mid n\langle X\rangle \ \twoheadrightarrow\ T(P) \mid X$$
   by taking the substitution $[X \mapsto T(R)]$, we obtain the term
   $$T(P) \mid T(R) \mid T(S) = T(P')$$
   By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
2. Suppose that $P_1 = m[in\ n.P \mid Q] \mid n[R] \mid S$ and that we apply a reduction step for capability $in\ n$ obtaining the process $P' = n[m[P \mid Q] \mid R] \mid S$.
   By definition,
   $$T(m[in\ n.P \mid Q] \mid n[R] \mid S) = m\langle q_{in\ n.P} \mid T(Q)\rangle \mid n\langle T(R)\rangle \mid T(S)$$
   By applying the TUC rule that defines the state $q_{in\ n.P}$, i.e.,
   $$m\langle q_{in\ n.P}\ \mid\ Y\rangle\ \mid\ n\langle Z\rangle\ \rightarrow\ n\langle m\langle T(P) \mid Y\rangle \mid Z\rangle\ \text{ for any } m \in \mathcal{N}, m \neq tuc$$
   by taking the substitution $[Y \mapsto T(Q), Z \mapsto T(R)]$, we obtain the term
   $$n\langle m\langle T(P) \mid T(Q)\rangle \mid T(R)\rangle \mid T(S) = T(P')$$
   By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
3. Suppose that we apply a reduction step for capability $out\ n$ in the process $P_1 = n[m[out\ n.P \mid Q] \mid R] \mid S$ obtaining the process $P' = m[P \mid Q] \mid n[\mathbf{0} \mid R] \mid S$.
   By definition,
   $$T(P_1) = n\langle m\langle q_{out\ n.P} \mid T(Q)\rangle \mid T(R)\rangle \mid T(S)$$
   By applying the TUC rule that defines the state $q_{out\ n.P}$, i.e.,
   $$n\langle m\langle q_{out\ n.P} \mid Y\rangle \mid Z\rangle \rightarrow m\langle T(P) \mid Y\rangle \mid n\langle q_{\mathbf{0}} \mid Z\rangle$$
   by taking the substitution $[Y \mapsto T(Q), Z \mapsto T(R)]$, we obtain the term
   $$m\langle T(P) \mid T(Q)\rangle \mid n\langle q_{\mathbf{0}} \mid T(R)\rangle \mid T(S) = T(P').$$
   By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
4. Suppose that we apply a reduction for $!P$ in the process $P_1 =!P \mid R$ and that we obtain the term $P' =!P \mid P \mid R$.
   By definition,
   $$T(P_1) = q_{!P} \mid T(R)$$
   By applying the TUC rule that defines the state $q_{!P}$, i.e.,
   $$q_{!P} \rightarrow q_{!P} \mid q_P$$

we get the term

$$q_{!P} \mid T(P) \mid T(R) = T(P')$$

By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.

5. Suppose that we apply a reduction for $!P \mid P$ in the process $P_1 = !P \mid P \mid R$ and that we obtain the term $P' = !P \mid R$.
   By definition,

   $$T(P_1) = q_{!P} \mid T(P) \mid T(R)$$

   By applying the TUC rule that defines the state $q_{!P}$, i.e.,

   $$q_{!P} \mid T(P) \to q_{!P}$$

   we get the term

   $$q_{!P} \mid T(R) = T(P')$$

   By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
6. Suppose that we apply a reduction $P \twoheadrightarrow P \mid \mathbf{0}$ in the process $P_1 = P \mid R$ and that we obtain the term $P' = P \mid \mathbf{0} \mid R$.
   Assume now that P is a parallel compositions $B_1 \mid \ldots \mid B_r$ of processes in which B_i is either $!A$, $\mathbf{0}$, $open\ n.A$, $in\ n.A$, or $out\ n.A$ for some name n and $i : 1, \ldots, r$. Now, by definition,

   $$T(P_1) = T(B_1) \mid \ldots \mid T(B_r) \mid T(R)$$

   Now let B_1 be either $\mathbf{0}$, $!A$ or $\alpha.A$ where α is a $in/out/open$ capability. Then, $T(B_1) = q_{B_1}$, i.e.,

   $$T(P_1) = q_{B_1} \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R)$$

   By applying one of the TUC rule named $zero$, namely, the rule

   $$q_{B_1} \to q_{B_1} \mid q_{\mathbf{0}}$$

   to state q_{B_1} we get the term

   $$q_{B_1} \mid q_{\mathbf{0}} \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R) = T(P')$$

   Now suppose that B_1 has the form $n\langle A\rangle$. Then, $T(B_1) = n\langle T(A)\rangle$, i.e.,

   $$T(P_1) = n\langle T(A)\rangle \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R)$$

   By applying one of the TUC rules named $zero$, namely

   $$n\langle X\rangle \to n\langle X\rangle \mid q_{\mathbf{0}}$$

   to state $n\langle T(A)\rangle$ with substitution $[X \mapsto T(A)]$, we get the term

   $$n\langle T(A)\rangle \mid q_{\mathbf{0}} \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R) = T(P')$$

   By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
7. Suppose that we apply a reduction $P \mid \mathbf{0} \twoheadrightarrow P$ in the process $P_1 = P \mid \mathbf{0} \mid R$ and that we obtain the term $P' = P \mid R$. Now, we can decompose P is a parallel compositions $B_1 \mid \ldots \mid B_r$ of processes in which B_i is either $!A$, $\mathbf{0}$, $in\ n.A$, or $out\ n.A$ for $i : 1, \ldots, r$. Now, by definition,

   $$T(P_1) = T(B_1) \mid \ldots \mid T(B_r) \mid \mathbf{0} \mid T(R)$$

   Now let B_1 be either $\mathbf{0}$, $!A$ or $\alpha.A$, Then, $T(B_1) = q_{B_1}$, i.e.,

   $$T(P_1) = q_{B_1} \mid q_{\mathbf{0}} \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R)$$

   By applying the TUC rule from the set $zero$

   $$q_{B_1} \mid q_{\mathbf{0}} \to q_{B_1}$$

   to state q_{B_1}, we get the term

   $$q_{B_1} \mid T(B_2) \mid \ldots \mid T(B_r) \mid T(R) = T(P')$$

Now suppose that B_1 has the form $n\langle A\rangle$. Then, $T(B_1) = n\langle T(A)\rangle$, i.e.,

$$T(P_1) = n\langle T(A)\rangle \mid q_0 \mid T(B_2) \mid \dots \mid T(B_r) \mid T(R)$$

By applying the TUC rule from the set *zero*

$$n\langle X\rangle \mid q_0 \to n\langle X\rangle$$

to state $n\langle T(A)\rangle$, with a substitution $[X \mapsto T(A)]$ we get the term

$$n\langle T(A)\rangle \mid T(B_2) \mid \dots \mid T(B_r) \mid T(R) = T(P')$$

By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.

Now suppose that $D > 1$ and let IH_2 be the inductive hypothesis for redex with depth $D' < D$. We proceed by a case analysis on the structure of P_1.

1. If P_1 is a parallel composition and $D > 1$, we can decompose P_1 as $B_1 \mid B_2$, where we have applied a reduction step to a subterm of B_1 (by hypothesis the depth of a top level redex is zero). Since $T(P_1) = T(B_1) \mid T(B_2)$ then we can apply the inductive hypothesis IH_2 to conclude that there exists a derivation going from t_1 to t'. By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$.
2. If $P_1 = n[B_1]$, since $D > 1$, then we have applied a reduction step to a subterm of B_1. Since $T(P_1) = n\langle T(B_1)\rangle$ then we can apply the inductive hypothesis IH_2 to conclude that there exists a derivation going from t_1 to t'. By applying IH_1, we have that there exists a derivation from $T(P_0)$ to $T(P')$. □

Given a term built on $\mathcal{Q}$ and $\mathcal{N}$, we now define the process $P(T)$ as follows:

$$\begin{array}{l} P(q_Q) = Q \ \text{ for any } q_Q \in \mathcal{Q} \\ P(m_1 \mid m_2) = P(m_1) \mid P(m_2) \\ P(n\langle m\rangle) = n[P(m)] \ \text{ for any } n \in Amb \end{array}$$

The following lemmas then hold.

Lemma 4.1 If t is a term built on $\mathcal{Q}$ and $\mathcal{N}$ in which there are no occurrences of terms of the form $n\langle\epsilon\rangle$ for some $n \in \mathcal{N}$, then $P(t)$ is an MA-process.

Proof By induction on the structure of t.

- Base:
  If $t = q_P$ where $P \in \{\mathbf{0}, in\ n.Q, out\ n.Q, open\ n.Q, !Q\}$ for some $n \in \mathcal{N}$, and MA process Q, then $P(t) = P$.
- Inductive step:
  - If $t = n\langle m'\rangle$ then $P(t) = n[P(m')]$. By inductive hypothesis, $P(m')$ is an MA process, so does $P(t)$.
  - If $t = t_1 \mid t_2$ then $P(t) = P(t_1) \mid P(t_2)$. By inductive hypothesis, $P(t_i)$ is an MA process for $i : 1, 2$, so does $P(t)$. □

Lemma 4.2 If Q is an MA process, $P(T(Q)) = Q$.

Proof By induction on the structure of Q.

- Base:
  If $Q \in \{\mathbf{0}, in\ n.Q', !Q'\}$ for some $n \in \mathcal{N}$ and MA process Q', then $T(Q) = q_Q$ and $P(q_Q) = Q$.
- Inductive step:
  - If $Q = n[Q']$ then $T(Q) = n\langle T(Q')\rangle$ and $P(n\langle T(Q')\rangle) = n[P(T(Q'))]$. By inductive hypothesis, $P(T(Q')) = Q'$, hence $P(T(Q)) = Q$.
  - If $Q = Q_1 \mid Q_2$ then $T(Q) = T(Q_1) \mid T(Q_2)$ and $P(T(Q_1) \mid T(Q_2)) = P(T(Q_1)) \mid P(T(Q_2))$. By inductive hypothesis, $P(T(Q_i)) = Q_i$ for $i \in \{1, 2\}$, hence $P(T(Q)) = Q$.

We are ready now to prove the following proposition.

Proposition 4.2 Given two ground multiset terms t and t' built on $\mathcal{Q}$ and $\mathcal{N}$ in which there are no subterms of the form $n\langle\epsilon\rangle$ for some $n \in \mathcal{N}$, if $tuc\langle t\rangle \Rightarrow^* tuc\langle t'\rangle$ in $\mathcal{R}_P$ then $P(t) \twoheadrightarrow^* P(t')$.

Proof In order to make things work we have to prove the stronger property: if $t_0 \to^* t'$, then $P(t_0) \twoheadrightarrow^* P(t')$. The proof is by induction of the length L of the derivation $t_0 \to^* t_1$.
IF $L = 0$ the thesis follows immediately.
Now suppose that the thesis holds for derivation of lengths L. Let's call this hypothesis IH_1. Consider now a derivation

$$t_0 \to^L t_1 \to t'$$

with length $L + 1$ such that $t_1 = s[m]$ and $t' = s[m']$ and $m \to m'$ is a ground instance of one of the rules used in the encoding of MA in TUC.
We proceed by induction on the nesting level D of $s[]$. The nesting level is defined by counting the number of tree terms (multisets do not increment it) needed to reach the placeholder [] in $s[]$.
If $D = 0$ then the rule is applied at the top level in t_1, we proceed then by a case analysis of the possible rewriting steps.

1. Suppose we apply a ground instance

$$q_{open\ n.Q} \mid n\langle s_1\rangle \to T(Q) \mid s_1$$

of the *open* TUC rule to $t_1 = q_{open\ n.Q} \mid n\langle s_1\rangle \mid m$. Then, by definition of $P(\cdot)$,

$$P(t_1) = q_{open\ n.Q} \mid n\langle s_1\rangle \mid m) = open\ n.Q \mid n\langle P(s_1) \mid P(m)\rangle$$

If we can apply the *open* reduction rule we obtain the process term

$$P_2 = Q \mid P(s_1) \mid P(m)$$

By Lemma 4.2, we have that $P(T(Q)) = Q$, thus

$$P(T(Q) \mid s_1 \mid m) = P_2$$

Thus, $P(t_1) = P(q_{open\ n.Q} \mid n\langle s_1\rangle \mid m) \twoheadrightarrow P_2 = P(t') = Q \mid P(s_1) \mid P(m)$. By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.

2. Suppose we apply a ground instance

$$m\langle q_{in\ n.Q} \mid s_2\rangle \mid n\langle s_3\rangle \to n\langle m\langle T(Q) \mid s_2\rangle \mid s_3\rangle$$

of the *in* TUC rule. Then, by definition of $P(\cdot)$,

$$P(m\langle q_{in\ n.Q} \mid s_2\rangle \mid n\langle s_3\rangle \mid r) = m[in\ n.Q \mid P(s_2)] \mid n[P(s_3)] \mid P(r)$$

If we can apply the *in* reduction rule we obtain the process term

$$P_2 = n[m[Q \mid P(s_2)] \mid P(s_3)] \mid P(r)$$

By Lemma 4.2, we have that $P(T(Q)) = Q$, thus

$$P_2 = P(n\langle m\langle T(Q) \mid s_2\rangle \mid s_3\rangle \mid r) = n\langle m\langle P(T(Q)) \mid P(s_2)\rangle \mid P(s_3)\rangle \mid P(r) = p(t')$$

By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.

3. Suppose we apply a ground instance

$$n\langle m\langle q_{out\ n.Q} \mid s_2\rangle \mid s_3\rangle \to m\langle T(Q) \mid s_2\rangle \mid n\langle q_{\mathbf{0}} \mid s_3\rangle$$

of the *out* TUC rule. Then, by definition of $P(\cdot)$,

$$P(n\langle m\langle q_{out\ n.Q} \mid s_2\rangle \mid s_3\rangle \mid r) = n[m[out\ n.Q \mid P(s_2)] \mid P(s_3)] \mid P(r)$$

If we can apply the *out* reduction rule we obtain the process term

$$P_2 = m[Q \mid P(s_2)] \mid n[\mathbf{0} \mid P(s_3)] \mid P(r)$$

By Lemma 4.2, we have that $P(T(Q)) = Q$, thus

$$P_2 = P(m\langle T(Q) \mid s_2\rangle \mid n\langle q_{\mathbf{0}} \mid s_3\rangle \mid r) = m[P(T(Q)) \mid P(s_2)] \mid n[\mathbf{0} \mid P(s_3)] \mid P(r) = P(t')$$

By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.

4. Suppose we apply the ground rule
$$q_{!Q} \;\rightarrow\; q_{!Q} \;\mid\; T(Q)$$
to a term $q_{!Q} \;\mid\; r$. By definition of $P(\cdot)$, we have that
$$P(q_{!Q} \;\mid\; r) = !Q \;\mid\; P(r)$$
Thus, we can apply the
$$!Q \twoheadrightarrow !Q \mid Q$$
reduction rule to $!Q \;\mid\; P(r)$ to obtain the process term
$$!Q \mid Q \mid P(r) = P(q_{!Q}) \mid P(T(Q)) \mid P(r) = P(t')$$
By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.
5. Suppose we apply the ground rule
$$q_{\mathbf{B}} \;\rightarrow\; q_{\mathbf{B}} \;\mid\; q_{\mathbf{0}}$$
to the term $q_{\mathbf{B}} \;\mid\; r$. Then, by definition of $P(\cdot)$,
$$P(q_{\mathbf{B}} \;\mid\; r) = B \;\mid\; P(r)$$
Thus, we can apply the $\mathbf{0}$ reduction rule to obtain the process term
$$B \mid \mathbf{0} \mid P(r) = P(q_{\mathbf{B}} \mid q_{\mathbf{0}} \mid r) = P(t')$$
By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.
6. The other cases are similar.

Now suppose that $D > 1$ and let IH_2 be the inductive hypothesis for redex with depth $D' < D$. We proceed by a case analysis on the structure of t_1.

1. If t_1 is a multiset term, then we can write it as a term $t_2 \mid t_3$ such that the redex of the reduction at depth $D > 1$ is in t_2 (by hypothesis the depth of a top level redex is zero). Since $P(t_1) = P(t_2) \mid P(t_3)$ then we can apply the inductive hypothesis IH_2 to conclude that there exists an MA-derivation going from $P(t_1)$ to $P(t')$. By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$.
2. If $t_1 = n\langle t_2\rangle$, since $D > 1$, then we have applied a reduction step to a subterm of t_2. Since $P(t_1) = n[P(t_2)]$ then we can apply the inductive hypothesis IH_2 to conclude that there exists a derivation going from $P(t_1)$ to $P(t')$. By applying IH_1, we have that $P(t) \twoheadrightarrow^* P(t')$. □

From Propositions 4.1 and 4.2, we obtain the following property that establishes the connection between reachability in MA and reachability in TUC.

Proposition 4.3 $P_0 \overset{*}{\twoheadrightarrow} P_1$ if and only if $tuc\langle T(P_0)\rangle \overset{*}{\Rightarrow} tuc\langle T(P_1)\rangle$ in $\mathcal{R}(P_0)$.

As a first consequence, Proposition 4.3 allows us to transfer the undecidability result for reachability in ppMA proved in [BT05] to our fragment of AC Term Rewriting.

Theorem 4.1 Reachability is undecidable in TUC.

Expressiveness of TUC In [MP05] it has been shown that ppMA is a Turing complete model. We can exploit this result and Prop. 4.3 to show that TUC is Turing complete as well. This property as well as the undecidability of reachability can be seen in a more direct way by defining an encoding of Two Counter Machines (2CMs) in TUC. Let us first define the instruction set of a 2CM with control states $s_1, \ldots, s_n$ and counters c_1 and c_2. When executed in state s_k, $INC_i(k, l)$ increments counter c_i and then moves to state s_l, while $DEC_i(k, l, m)$ decrements c_i and then moves to s_l if $c_i > 0$, or moves to state s_m if $c_i = 0$. A 2CM configuration consists of a control state and of the current values of the counters.

The TUC representation of a 2CM makes use of the following constants and labels

$$\mathcal{Q} = \{s_1, \ldots, s_n, zero_1, zero_2\}$$
$$\mathcal{N} = \{c_1, c_2\}$$

A 2CM configuration $C = (s_i, c_1 = n_1, c_2 = n_2)$, for $n_1, n_2 \geq 0$, is represented by the term

$$C^\bullet = tuc\langle s_i \mid c_1^{n_1}\langle zero_1\rangle \mid c_2^{n_2}\langle zero_2\rangle\rangle$$

where

- $c_i^0\langle t\rangle = t$, and
- $c_i^{n+1}\langle t\rangle = c_i\langle c_i^n\langle t\rangle\rangle$ for $n \geq 0$.

To encode $INC_i(k, l)$, we use the TUC rules

$$\begin{array}{ll} (inc) & s_k \mid c_i\langle X\rangle \to s_l \mid c_i^2\langle X\rangle \\ (incz) & s_k \mid zero_i \to s_l \mid c_i\langle zero_i\rangle \end{array}$$

To encode $DEC_i(k, l, m)$, we use the TUC rules

$$\begin{array}{ll} (dec) & s_k \mid c_i\langle X\rangle \to s_l \mid X \\ (test) & s_k \mid zero_i \to s_m \mid zero_i \end{array}$$

The latter rule encodes the zero-test on counter c_i. It is easy to verify that the 2CM configuration C_1 evolves into C_2 via instruction i if and only if the term $C_1^\bullet$ can be rewritten into $C_2^\bullet$ using the TUC rule encoding i. This encoding gives us a direct proof to show that TUC is a Turing complete model. In the following section we introduce a non trivial subclass of TUC that can still be used to model movement operations of MA-calculi and for which reachability is decidable.

5. The structure preserving fragment of TUC

The structure preserving fragment of TUC (TUC^{sp}) is obtained by restricting the syntax of rules so as to avoid that their application can remove internal nodes. This property does not imply that the resulting system is monotonic with respect to the size of terms. Indeed, although we forbid rules like $n\langle X\rangle \to X$ and $n\langle X\rangle \to a$ that delete internal nodes or subtrees, a rule that removes a leave like $a \mid a \to a$ is still definable in TUC^{sp}.

In order to define the syntax of TUC^{sp}-rules, we introduce two subclasses of tree terms, namely RT_L (restricted terms used in the left-hand side of rules) and RT_R (used in the right-hand side) with the following characteristics. Leaves in $\mathcal{Q}$ are the only ground terms we allow in RT_L. Every non ground tree term occurring in an RT_L term must be of the form $n\langle t_1 \mid \ldots \mid t_n \mid X\rangle$, where $n \in \mathcal{N}$, X is a variable and t_i is an RT_L term for $1 \leq i \leq n$ and $n \geq 0$. RT_R terms are slightly more general since they can have ground trees as subterms.

Formally, RT_L and RM_L are the least sets satisfying

- $\mathcal{Q} \subseteq RT_L$;
- if $t_1, \ldots, t_n \in RT_L$ and $X \in \mathcal{V}$, then $t_1 \mid \ldots \mid t_n \mid X \in RM_L$ for $n \geq 0$;
- if $m \in RM_L$ and $n \in \mathcal{N}$, then $n\langle m\rangle \in RT_L$.

RT_R and RM_R are the least sets satisfying

- $\mathcal{Q} \subseteq RT_R$ and $\epsilon \in RM_R$;
- if $t_1, \ldots, t_n \in RT_R$, then $t_1 \mid \ldots \mid t_n \in RM_R$ for $n \geq 1$;
- if $t_1, \ldots, t_n \in RT_R$ and $X \in \mathcal{V}$, then $t_1 \mid \ldots \mid t_n \mid X \in RM_R$ for $n \geq 0$;
- if $m \in RM_R$ and $n \in \mathcal{N}$, then $n\langle m\rangle \in RT_R$.

In the rest of the paper, we use $n\langle t_1, \ldots, t_n\rangle$ as an abbreviation for $n\langle t_1 \mid \ldots \mid t_n\rangle$ and $n\langle t_1, \ldots, t_n \mid X\rangle$ for $n\langle t_1 \mid \ldots \mid t_n \mid X\rangle$.

Given a term t let $IntNds(t)$ denote the number of occurrences of labels in $\mathcal{N}$ (internal nodes) in t. Formally, $IntNds(t)$ is defined by induction on t as follows:

- $IntNds(\epsilon) = IntNds(X) = IntNds(q) = 0$ for $X \in \mathcal{V}$ and $q \in \mathcal{Q}$;
- $IntNds(t_1 \mid \dots \mid t_k) = IntNds(t_1 \mid \dots \mid t_k \mid \ X) = IntNds(t_1) + \cdots + IntNds(t_k)$;
- $IntNds(n\langle m\rangle) = IntNds(m) + 1$.

For instance, if $t = n\langle n\langle q \mid q' \mid X\rangle \mid n'\langle q\rangle\rangle$, $X \in \mathcal{V}$, $q, q' \in \mathcal{Q}$, then $IntNds(t) = 3$.
We are ready now to define the class of structure preserving rules.

Definition 5.1 A structure preserving rule $l \to r$ is such that

1. $l = t_1 \mid \dots \mid t_n$, and $t_i \in RT_L$ for $i : 1, \dots, n$;
2. $r = t'_1 \mid \dots \mid t'_m$ and $t'_i \in RT_R$ for $i : 1, \dots, m$;
3. l and r have the same set V of variables;
4. each variable in V occurs once in l and once in r;
5. $IntNds(l) \leq IntNds(r)$.

A TUC^{sp}-theory is a set of structure preserving rules.

By definition of RT_L and RT_R, with the first condition we associate one and only one variable to each node, whereas in the right-hand side we also admit internal nodes with ground subtrees. The last three conditions ensure that the tree structure of terms, involved in a rewriting step can only get larger: condition (3) and (4) ensure that subtrees can neither be eliminated nor duplicated, condition (5) ensures that internal nodes can never be eliminated. Since $IntNds(t)$ only counts the number of occurrence of labels of internal nodes, these conditions do not imply monotonicity with respect to the size of a term.
Some examples of TUC^{sp} rules are:

- $a \mid a \to a$ and $a \to a \mid a$ (resp. deletion and insertion of a leaf);
- $n\langle X\rangle \mid m\langle Y\rangle \to n\langle Y\rangle \mid m\langle X\rangle$ (swapping of subtrees);
- $n\langle q \mid X\rangle \mid m\langle q \mid Y\rangle \to n\langle r \mid n\langle X\rangle\rangle \mid m\langle r \mid m\langle Y\rangle\rangle \mid n\langle\epsilon\rangle$ (insertion of nodes).

As opposite, the following rules are not in the TUC^{sp} fragment:

- $n\langle X\rangle \to n\langle X\rangle \mid n\langle X\rangle$ (duplication of a subtree),
- $n\langle X\rangle \to X$ (removal of a node),
- $n\langle\epsilon\rangle \to n\langle\epsilon\rangle \mid q$ (ground term on the left-hand side),
- $n\langle X\rangle \to a$ (removal of a complete subtree).

Despite all restrictions of Definition 5.1, TUC^{sp} is still a powerful computational model.
As a first example, it is easy to check that Petri nets can be encoded in TUC^{sp} by using terms with one nesting level only. For instance, if $Q = \{a, b, c\}$, then the ground term $tuc\langle a \mid a \mid b\rangle$ can be used to represent a marking in which places a, b and c have two, one, and zero tokens, respectively. The rule $a \mid a \to c$ represents a transition that removes two tokens from place a, and adds one token to place c. From this observation, it follows that TUC^{sp} reachability is at least as hard as Petri net reachability.
Actually, in the next section we show that reachability between terms in a TUC^{sp}-theory can always be reduced to a reachability problem for a Petri net extracted from the terms and from the rules (Proposition 6.1). From this property and from the decidability of Petri net reachability [May84, Rao82], we obtain that TUC^{sp}-reachability is decidable (Theorem 6.1).

6. Decidability of reachability in TUC^{sp}

6.1. Rationale

Given a TUC^{sp}-theory $\mathcal{R}$, and two ground tree terms t_0 and t_f, let us consider the reachability problem $t_0 \overset{*}{\Rightarrow} t_f$. By definition, if there exist $t_1, \dots, t_k$ such that $t_0 \Rightarrow t_1 \Rightarrow \cdots \Rightarrow t_k \Rightarrow t_f$, then $IntNds(t_i) \leq IntNds(t_f)$ for $i : 0, \dots, k$. This property gives us an upper bound on the number of internal nodes—but not on the number of leaves—occurring in each intermediate tree t_i. As an example, consider the two TUC^{sp} rules $a \to a \mid a$ and $a \mid a \to a$ and the tree term $t = tuc\langle a\rangle$. In a derivation from t to t we can only find terms with one nesting level but with any number of occurrences of a.

To keep track of the number of leaves we can resort to a Petri net [Rei85], see also Appendix A. The construction is based on the following key observations. Firstly, we can exploit the above mentioned upper bound on the number of internal nodes to isolate the finite set of possible *tree structures* (tree terms without leaves) that can occur in a derivation leading to t_f. The Petri net has a finite set of places (T-places) each one denoting one of these tree structures. Only one T-place can be marked in a reachable marking. Another finite set of places (L-places) is used to keep track of the current number of occurrences of leaves (constants in $\mathcal{Q}$) at every level of the tree structure denoted by the currently marked T-place. The association between an internal node of a T-place and the set of its leaves is maintained via a finite set of *position labels*. To make the intuition clearer, let us consider the following example.

6.1.1. An example

Consider the TUC^{sp}-theory consisting of the following rules

$$\begin{array}{l} a \to n\langle a\rangle \\ n\langle X\rangle \ \mid \ m\langle Y\rangle \to n\langle m\langle Y\rangle \ \mid \ X\rangle \end{array}$$

The first rule replaces a constant a with a tree with root n and a single leaf a. Repeated applications of this rule may give rise to trees of unbounded depth of the form

$$n\langle n\langle \dots n\langle a\rangle \dots\rangle\rangle$$

The second rule defines an interaction between two adjacent trees. When a tree t with root m is at the same level of a tree t' with root n, then t m can move and become a son of n. Let us now consider the reachability problem $t \Rightarrow t'$ for the two terms

$$t = tuc\langle n\langle a\rangle \ \mid \ m\langle a\rangle\rangle \quad t' = tuc\langle n\langle n\langle m\langle a\rangle \ \mid \ a\rangle\rangle\rangle$$

Now since the rules are in TUC^{sp}, every tree term that occurs in a derivation from t to t' must have at most three internal nodes with labels in $\mathcal{N} = \{n, m\}$ (TUC^{sp} rules are monotonic w.r.t. the nesting level of terms). As an example, although the term $t_1 = tuc\langle n\langle n\langle n\langle m\langle a\rangle \ \mid \ a\rangle\rangle\rangle\rangle$ is derivable from t, it cannot occur in a derivation from t to t' (in TUC^{sp} we cannot write rules that delete internal nodes of a tree).

The Petri net associated to the reachability problem for t and t' has a set of places, called T-places, associated to the set of of tree structures with root tuc and at most three internal nodes with labels in $\mathcal{N} = \{n, m\}$. Notice that this set is finite. To distinguish occurrences of the same label in different nodes we can add positions $\bullet_1, \bullet_2, \dots$ to the internal nodes and we use the syntax $a\langle t \mid \bullet_i\rangle$ to associate the position i to a node with label $a \in \mathcal{N} \cup \{tuc\}$ and subtree t. As an example,

- $tuc\langle \bullet_1\rangle$ represents the tree structure with no labels from $\mathcal{N}$; $\bullet_1$ represent the top-level position;
- $tuc\langle n\langle \bullet_2\rangle \mid \bullet_1\rangle$ represents the tree structure with a single node n,
- $tuc\langle n\langle n\langle \bullet_3\rangle \mid \bullet_2\rangle \mid \bullet_1\rangle$ represents the tree structure with a node n whose son has again label n; here we use different position labels (2 and 3) to distinguish them;
- $tuc\langle n\langle n\langle \bullet_3\rangle \mid m\langle \bullet_4\rangle \mid \bullet_2\rangle \mid \bullet_1\rangle$ represents the tree structure with a node n and two sons: n with position 3 and m with position 4.

In Fig. 3 we illustrate the tree structures associated to the above described four terms. Each one of these terms is associated to a T-place. Since a term has only one possible tree structure, only one T-place is marked in every reachable marking of our Petri net encoding. As an example, the tree structure

$$T = tuc\langle n\langle \bullet_2\rangle \ \mid \ n\langle \bullet_3\rangle \ \mid \ \bullet_1\rangle$$

can be associated to the term t: tuc has position 1 (top-level), node n has position 2, and node m has position 3. Furthermore, the tree structure

$$S = tuc\langle n\langle n\langle m\langle \bullet_3\rangle \mid \bullet_4\rangle \ \mid \ \bullet_2\rangle \ \mid \ \bullet_1\rangle$$

can be associated to the term t'. Here we have two occurrences of label n. Notice that we associate a new label, namely 4, to the node with the second occurrence of n. In our example the node with label m moves together with its subtree represented here by label 3. It is also important to remark that we have to consider all possible tree structures and permutations of positions. Thus, several T-places are associated to a term t (e.g. we could

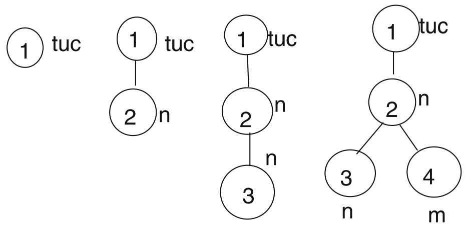

Fig. 3. Example of tree structures with position labels

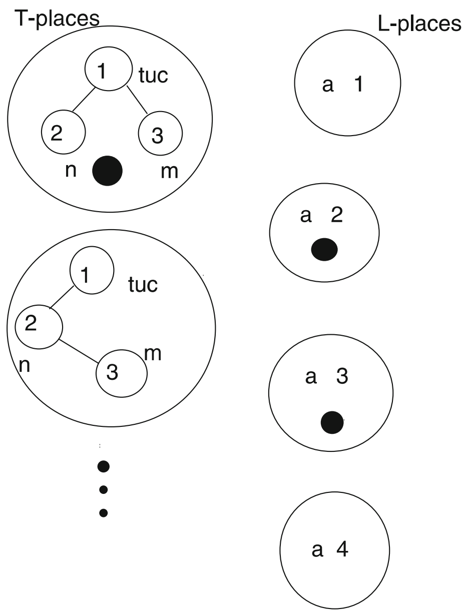

Fig. 4. Initial marking

swap position 1 with 2 in the tree structure T for t). We show later that this amounts to consider a finite set of Petri net reachability problems, one for each choice of positions labels.

T-places are only useful to remember the tree structure of a term that occurs in a derivation leading to t'. We still need to keep track of the number of constants that may appear at each level of a tree structure (the currently marked T-place). To distinguish between occurrences of the same constant at different levels we introduce a set of L-places, consisting of pairs $\langle a, \bullet_i \rangle$ where i is one of the four possible positions in a T-place (i.e. $i \in 1, \ldots, 4\}$).

Since in t we only have one occurrence of a inside n and one inside m, we can associate to it a marking m in which the only marked places are: the T-place $T = tuc\langle n\langle \bullet_2 \rangle \mid n\langle \bullet_3 \rangle \mid \bullet_1 \rangle$ and the L-places $\langle a, \bullet_2 \rangle$ and $\langle a, \bullet_3 \rangle$. All other T- and L-places are empty. For instance, since there are no constants at the top level, $\langle a, \bullet_1 \rangle$ has no tokens. The initial marking is also depicted in Fig. 4, black dots correspond to tokens (for brevity we show only two T-places). Similarly, in the target marking m' associated to t' we put one token in the T-place S, one token in the L-place $\langle a, 4 \rangle$, and one token in the L-place $\langle a, 3 \rangle$ Following this idea, we can reduce the original reachability problem from t to t' to a reachability problem from marking m to marking m'. Let us now describe the transitions of the Petri net. The rule $a \to n\langle a \rangle$ modifies the tree structure of the term to which it is applied. Thus, in our Petri net for each T-place we add a set of transitions that mimic the effect of applying the transitions at different levels of the corresponding tree structure. As an example, the transition in Fig. 5 mimic the effect of the application of the rule on the tree structure associated to t under the hypothesis that a occurs inside n (i.e. that there is at least one token in the L-place $\langle a, 2 \rangle$ where 2 is the position of node n). The transition marks a T-place in which there is a new node with label n in position 4 that corresponds to the tree structure of the newly generated term, and adds a token to the L-place $\langle a, 4 \rangle$ (i.e. a constant a inside the new subtree with root n). For the rule $n\langle X \rangle | m\langle Y \rangle \to n\langle m\langle Y \rangle | X \rangle$ we have to introduce transitions that update the current T-place with

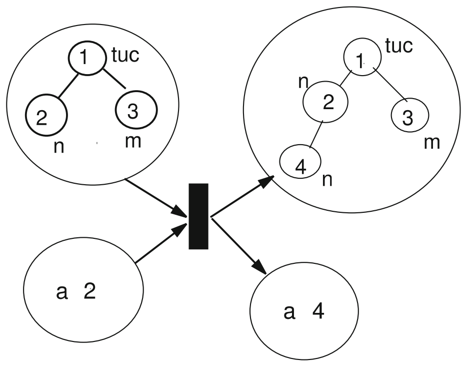

Fig. 5. Example of Petri net encoding of the rule $a \to n\langle a\rangle$

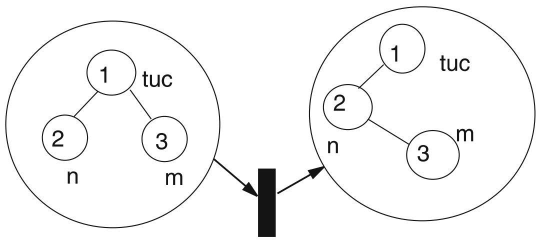

Fig. 6. Example of Petri net encoding of the rule $n\langle X\rangle \mid m\langle Y\rangle \to n\langle m\langle Y\rangle \mid X\rangle$

no modifications of L-places. As an example, the transition in Fig. 6 mimic the effect of the application of the rule on the tree structure associated to t. It is important to notice that the movement of a subtree is obtained by simply updating the T-place to a T-place with an adequate rearrangement of position labels. For instance, in Fig. 6 the T-place in the postset of the transition maintains position 3 for node m. This way, all L-place associated to position 3 are automatically transferred to the level in which m has moved.

The importance of using a Petri net can be understood when considering rules that generate or consume multiple occurrences of constants. As an example the rule $a \to a \mid a$ is represented by a finite set of Petri net transitions, each one associated to one of the four positions in a T-place. The transition for position i checks if there is a token in the L-place $\langle a, i\rangle$ and then adds a token to the same place. Similarly, we associate to the rule $a \mid a \to a$ a transition for each position in a T-place. The transition for position i checks if there are at least two tokens in the L-place $\langle a, i\rangle$ and then removes one of them.

This kind of rules may give rise to Petri nets with an unbounded number of tokens in the set of reachable markings. This is the feature that makes the reachability problem for TUC^{sp}-theories not trivial at all.

6.2. Formal definitions

In the rest of the section we formalize the encoding of a TUC^{sp}-reachability problem as Petri net reachability.

6.2.1. Tree structures with position labels

Let us first introduce the set of terms we use to represent T-places, i.e., *tree structures* with position labels. Let $\mathcal{P}$ be the set of position labels $\{\bullet_1, \bullet_2, \dots, \}$.
Then TS (tree structures) and MTS (multisets of TS terms) are the least set satisfying:

- if $t_1, \dots, t_r \in TS$ and $\bullet_i \in \mathcal{P}$, then $t_1, \dots, t_r \mid \bullet_i \in MTS$ for $r \geq 0$ and $i \geq 1$;
- if $m \in MTS$ and $n \in \mathcal{N}$, then $n\langle m\rangle \in TS$.

We still use $IntNds(t)$ to denote the number of occurrences of labels of internal nodes in t (we assume that $tuc \notin \mathcal{N}$). Furthermore, we use $Pos(t)$ to denote the multiset that keeps track of the number of occurrences in

t of position labels in $\mathcal{P}$. As an example, let $\mathcal{N} = \{n, p, q\}$, then $t = tuc\langle n\langle \bullet_4\rangle, p\langle p\langle \bullet_5\rangle \mid \bullet_3\rangle, \bullet_8\rangle$ is a TS term with $IntNds(t) = 3$. Furthermore, we have that $Pos(t) = \{\bullet_3, \bullet_4, \bullet_5, \bullet_8\}$. Notice that a tree structure represents a skeleton for an infinite set of tree terms, all those obtained by populating internal nodes with multisets of leaves.

Given a natural number $N \geq 1$, $t \in TS$ is said to be N-*well-formed* if and only if $Pos(t) = \{\bullet_1, \ldots, \bullet_{k+1}\}$ for $k \leq N$, i.e., t has k internal nodes and $k+1$ position labels numbered $1, \ldots, k+1$ (the top level label in a multiset term is not associated to an internal node). We use WS_K to denote the set of MTS terms that are K-well-formed. As an example, WS_3 consists of MTS-terms like $n_1\langle \bullet_{i_1}\rangle \mid n_2\langle \bullet_{i_2}\rangle \mid n_3\langle \bullet_{i_3}\rangle \mid \bullet_{i_4}$, $n_1\langle n_2\langle \bullet_{i_1}\rangle, n_3\langle \bullet_{i_2}\rangle \mid \bullet_{i_3}\rangle \mid \bullet_{i_4}$, $\ldots$, where $n_i \in \mathcal{N}$ for $i : 1, \ldots, 3$, and (i_1, i_2, i_3, i_4) is a permutation of $(1, 2, 3, 4)$.

6.2.2. Petri net: T-places and L-places

Let $\mathcal{R}$ be a TUC^{sp}-theory, t_0 and t_f be two ground tree terms and K be the number of internal nodes of t_f, i.e., $K = IntNds(t_f)$. The Petri net N associated to the reachability problem $t_0 \overset{*}{\Rightarrow} t_f$ has two kind of places. The set of *T-places* is defined as $\{tuc\langle w\rangle \mid w \in WS_K\}$ (the set of tree structures with at most K internal nodes with labels in $\mathcal{N}$). The set of *L-places* is defined as $\{\langle q, \bullet_i\rangle \mid q \in \mathcal{Q}\ and\ i \in \{1, \ldots, K+1\}\}$. Transitions have in their preset and postset a single T-place together with a set of L-places.

6.2.3. Petri net: transitions

To define the marking and transitions of the Petri net associated to t_0, t_f, and $\mathcal{R}$, we introduce two functions called *match* and *extract*. The intuition behind their definitions is as follows. Given a rule $l \to r$ and a T-place s, we first try to match the left-hand side l with a multiset term m occurring in s. The matching procedure works by a parallel inspection of the structure of l and m and returns a substitution θ for the variables in l. Every variable is associated to a multiset term labelled by a position. During the matching phase, we collect in a set S all the leaves occurring in l. Leaves are labelled with the position of the corresponding internal nodes. If the matching between l and m succeeds then we can build a transition t with $\{s\} \cup S$ as input places. To determine the output places, we first have to apply the substitution θ, computed during the matching phase, to the right-hand side r, and then extract the new tree structure m' and the corresponding leaves from $\theta(r)$. Since r can have ground subterms, during the visit of $\theta(r)$ we may need to introduce new position labels, distinct from those in s. New labels are collected in a set N. Finally, we have to replace m with m' in s to obtain the output T-place s' of the transition.

For instance, suppose that $s = n\langle m\rangle \mid \bullet_1$, $m = n'\langle n\langle \bullet_3\rangle \mid \bullet_2\rangle$, and that the rule $l \to r$ is such that $l = n'\langle q_1, q_2 \mid X\rangle$ and $r = n'\langle q_3 \mid X\rangle \mid n'\langle q_4\rangle$. The mapping $\theta = [X \mapsto n\langle \bullet_3\rangle \mid \bullet_2]$ can be used to unify m and l. The set $S = \{\langle q_1, \bullet_2\rangle, \langle q_2, \bullet_2\rangle\}$. contains the leaves in l partitioned according to the position labels in m. Furthermore, from $\theta(r) = n'\langle q_3, n\langle \bullet_3\rangle \mid \bullet_2\rangle \mid n'\langle q_4\rangle$ we can extract the labelled term (without leaves) $m' = n'\langle n\langle \bullet_3\rangle \mid \bullet_2\rangle \mid n'\langle \bullet_4\rangle$ after having associated the new position label $\bullet_4$ to the ground tree $n'\langle q_4\rangle$. The set $S' = \{\langle q_3, \bullet_2\rangle, \langle q_4, \bullet_4\rangle\}$ contains the leaves of the new tree structure (again partitioned according to the position labels). Finally, let s' be the term obtained by replacing m with m' in s. Then, we can define a transition with input places $\{s\} \cup S$ and output places $\{s'\} \cup S'$.

Match Since TUC has an AC constructor, we need to work with sets of matching substitutions. The function *match* takes in input a (multiset) term $s \in WS_{\mathcal{M}}$ and a term $t \in RM_L$ and returns a set of pairs $\langle \theta, S\rangle$, such that θ is a mapping from the variables in t to multisets of TS terms, and S is a multiset of L-places.

Formally, $\langle \theta, S\rangle \in match(s, t)$ iff there exist $u, v, w \geq 0$ and $k \geq 1$ such that

- $s = n_1[s_1], \ldots, n_v[s_v], r_1, \ldots, r_u \mid \bullet_k$,
- $t = q_1, \ldots, q_w, n_1[t_1], \ldots, n_v[t_v] \mid X$ where $q_i \in \mathcal{Q}$ for $1 \leq i \leq w$
- $\langle \theta_i, S_i\rangle \in match(s_i, t_i)$ for $1 \leq i \leq v$,
- $\theta = \{X \mapsto (r_1, \ldots, r_u \mid \bullet_k)\} \cup \bigcup_{i=1}^{v} \theta_i$,
- $S = \{\langle q_1, \bullet_k\rangle, \ldots, \langle q_w, \bullet_k\rangle\} \oplus \bigoplus_{i=1}^{v} S_i$.

Here $\oplus$ denotes *multiset union*. Notice that if $v = u = 0$, then $\theta = \{X \mapsto \bullet_k\}$, and that if $v = u = w = 0$ then $S = \emptyset$. For instance, consider

$$s = a\langle b\langle a\langle \bullet_3\rangle \mid \bullet_2\rangle, c\langle \bullet_1\rangle, a\langle \bullet_4\rangle \mid \bullet_5\rangle \mid \bullet_6 \quad t = a\langle b\langle q_1 \mid X\rangle, a\langle q_2 \mid Y\rangle \mid Z\rangle \mid W$$

```
function ComputeTransitions($\mathcal{R}$: set of $\mathsf{TUC}^{sp}$-rules): set of Petri net transitions
begin
 let $\tau_\mathcal{R} = \emptyset$;
 for every rule $l \to r \in \mathcal{R}$
  for every T-place $v$ and term $t = n\langle m \mid \bullet_k\rangle$ such that $v = v'[t]$
   let $t' = n\langle l \mid Z\rangle$ for a variable $Z \notin var(l)$;
   for every $\langle \theta, S\rangle \in match(m \mid \bullet_k, l \mid Z)$
    for every $\langle s \mid \bullet_{tmp}, S'\rangle \in extract(\theta(r))$ s.t. $w = v'[n\langle s \mid \theta(Z)\rangle]$ is $K$-well-formed
     add to $\tau_\mathcal{R}$ a transition with preset $\{v\} \oplus S$ and postset $\{w\} \oplus S'$;
 return $\tau_\mathcal{R}$.
end.
```

Fig. 7. Algorithm for computing Petri net transitions

Then, $match(s, t)$ returns

$$\theta = \{X \mapsto (a\langle \bullet_3\rangle \mid \bullet_2),\ Y \mapsto \bullet_4,\ Z \mapsto (c\langle \bullet_1\rangle \mid \bullet_5),\ W \mapsto \bullet_6\}$$
$$S = \{\langle q_1, \bullet_2\rangle, \langle q_2, \bullet_4\rangle\}$$

As another example, if $s = n\langle \bullet_1\rangle \mid n\langle \bullet_2\rangle \mid \bullet_3$ and $t = n\langle X\rangle \mid n\langle Y\rangle \mid Z$ then $\{X \mapsto \bullet_1, Y \mapsto \bullet_2, Z \mapsto \bullet_3\}$ and $\{X \mapsto \bullet_2, Y \mapsto \bullet_1, Z \mapsto \bullet_3\}$ are two possible substitutions returned by $match(s, t)$ (in both cases $S = \emptyset$).

Extract Let θ be a substitution computed by $match(s, t_1)$. Suppose that t_2 is an RM_R term with the same variables as t_1. The term $\theta(t_2)$ obtained by applying θ to t_2 is such that all variables in t_2 are replaced by MTS terms (multiset of tree structures with position labels). The *extract* function extracts the tree structure of $\theta(t_2)$ and associates to each leaf in t_2 a position label accordingly to those injected by θ in t_2. The function *extract* returns a set of pairs $\langle s, S\rangle$, where s is an MTS term, and S is a multiset of L-places.
Formally, $\langle s, S\rangle \in extract(t)$ iff there exist $v, w \geq 0$, and $k \geq 1$ such that

- $t = p_1, \ldots, p_w, n_1\langle m_1\rangle, \ldots, n_v\langle m_v\rangle \mid \bullet_k$ or $t = p_1, \ldots, p_w, n_1\langle m_1\rangle, \ldots, n_v\langle m_v\rangle$ with $p_j \in \mathcal{Q}$ for $0 \leq j \leq w$,
- $(s_i, S_i) \in extract(m_i)$ for $1 \leq i \leq v$,
- $s = n_1\langle s_1\rangle, \ldots, n_v\langle s_v\rangle \mid \bullet_k$, and
- $S = \{\langle p_1, \bullet_k\rangle, \ldots, \langle p_w, \bullet_k\rangle\} \oplus \bigoplus_{i=1}^{v} S_i$.

Notice that if a certain level of t there is no position label, then we choose one non-deterministically to label the corresponding level of s. As an example, given $t = a\langle q_1, b\langle q_2, a\langle \bullet_3\rangle \mid \bullet_2\rangle, a\langle \bullet_4\rangle\rangle$, $extract(t)$ contains the pair $s = a\langle b\langle a\langle \bullet_3\rangle \mid \bullet_2\rangle, a\langle \bullet_4\rangle \mid \bullet_5\rangle$, $S = \{\langle q_1, \bullet_5\rangle, \langle q_2, \bullet_2\rangle\}$, where $\bullet_5$ is a new label.

Transitions The set $\tau_\mathcal{R}$ of Petri net transitions associated to rules in $\mathcal{R}$ is computed via the algorithm of Fig. 7. Notice that, well-formedness ensures that all position labels introduced in w by *extract* must be distinct and new with respect to those in $Pos(t)$.

6.2.4. Petri Net: markings

Given a ground tree term t, the set of Petri net markings $marking(t)$ associated to t is the set $marking(t) = \{M \mid \langle s, S\rangle \in extract(t),\ M = \{s\} \oplus S\}$. Intuitively, the marking associated to t has one token in the T-place s representing the tree structure of t, and k tokens in place $\langle q, \ell\rangle$ if q occurs k times at position ℓ in s.

Example 6.1 Given

$$t = a\langle q_1, b\langle q_2, q_2, q_3, a\langle q_2\rangle\rangle, a\langle q_1, b\langle q_1\rangle\rangle, a\langle q_1, q_2\rangle\rangle$$

$extract(t)$ returns $s = a\langle s'\rangle \mid \bullet_7$ where

$$s' = b\langle a\langle \bullet_1\rangle \mid \bullet_2\rangle, a\langle b\langle \bullet_3\rangle \mid \bullet_4\rangle, a\langle \bullet_5\rangle \mid \bullet_6$$

and

$$S = \{\langle q_1, \bullet_6\rangle, \langle q_2, \bullet_2\rangle, \langle q_2, \bullet_2\rangle, \langle q_3, \bullet_2\rangle, \langle q_2, \bullet_1\rangle, \langle q_1, \bullet_4\rangle, \langle q_1, \bullet_3\rangle, \langle q_1, \bullet_5\rangle, \langle q_2, \bullet_5\rangle\}$$

We obtain the marking M_t associated to $\langle s, S\rangle$ by putting 1 token in the T-place s, 1 token in the L-place $\langle q_1, \bullet_6\rangle$, 2 tokens in the L-place $\langle q_2, \bullet_2\rangle$, etc.

Now consider the rule $l \to r$ such that $l = a\langle q_2 \mid X\rangle \mid b\langle q_3 \mid Y\rangle$ and $r = a\langle q_1, b\langle q_1 \mid X\rangle \mid Y\rangle$. Then, we have that $t' = a\langle l \mid Z\rangle$ can be matched against s.

Indeed $match(s',\ l \mid Z)$ returns among other possible solutions

$$\begin{array}{l}\theta = \{X \mapsto \bullet_5,\quad Y \mapsto (a\langle \bullet_1\rangle \mid \bullet_2),\quad Z \mapsto (a\langle b\langle \bullet_3\rangle \mid \bullet_4\rangle \mid \bullet_6)\} \\ U = \{\langle q_2, \bullet_5\rangle, \langle q_3, \bullet_2\rangle\}\end{array}$$

Another possibility is to swap the terms associated to Y and Z. Furthermore, $\theta(r) = a\langle q_1, b\langle q_1 \mid \bullet_5\rangle, a\langle \bullet_1\rangle \mid \bullet_2\rangle$ and $extract(\theta(r))$ contains the pair $\langle s', S'\rangle$, where $s' = a\langle b\langle \bullet_5\rangle, a\langle \bullet_1\rangle \mid \bullet_2\rangle \mid \bullet_{new}$ and $S' = \{\langle q_1, \bullet_2\rangle, \langle q_1, \bullet_5\rangle\}$. Thus, we build a transition τ with input places $\{s\} \cup U$ and output places $\{s'''\} \cup S'$ where

$$s''' = a\langle s', a\langle b\langle \bullet_3\rangle \mid \bullet_4\rangle \mid \bullet_6\rangle = a\langle a\langle b\langle \bullet_5\rangle, a\langle \bullet_1\rangle \mid \bullet_2\rangle, a\langle b\langle \bullet_3\rangle \mid \bullet_4\rangle \mid \bullet_6\rangle$$

Furthermore, since $U \subseteq S$ we have that τ is enabled at M_T. Its firing leads to the marking obtained by removing one token from the T-place s and from the L-places $\langle q_2, \bullet_5\rangle$ and $\langle q_3, \bullet_2\rangle$, and by adding one token to the T-place s''' and to the L-places $\langle q_1, \bullet_2\rangle$ and $\langle q_1, \bullet_5\rangle$. Namely, the newly marked L-places is

$$V = \{\langle q_1, \bullet_2\rangle, \langle q_1, \bullet_5\rangle, \langle q_1, \bullet_6\rangle, \langle q_2, \bullet_2\rangle, \langle q_2, \bullet_2\rangle, \langle q_2, \bullet_1\rangle, \langle q_1, \bullet_4\rangle, \langle q_1, \bullet_3\rangle, \langle q_1, \bullet_5\rangle\}$$

From the resulting marking we can reconstruct the term obtained by applying the rule. We just have to populate s''' with the corresponding leaves in V to obtain the term:

$$a\langle q_1, a\langle q_1, b\langle q_1\rangle, a\langle q_2, q_2\rangle\rangle, a\langle q_1, b\langle q_1\rangle\rangle\rangle$$

6.3. Formal properties of the Petri net

In this section we formally study the relation between a TUC^{sp}-theory and the Petri nets extracted using the algorithm explained in the previous section. Let us first recall that if $t \Rightarrow^* t'$, then the number of internal nodes in t is less or equal than the number of internal nodes K of t'. We now prove that $t \Rightarrow t'$ if and only if there exist $M \in markings(t)$ and $M' \in markings(t')$ and a transition in $N_{\mathcal{M},\mathcal{R}}$ such that M evolves into M'.
Let $\preccurlyeq$ denote multiset inclusion, i.e., $S \preccurlyeq S'$ if, for every symbol a, S' has at least the same number of occurrences of a as S', e.g., $[a, b] \preccurlyeq [a, a, b, c]$ while $[a, b] \not\preccurlyeq [b, b, b, c]$. The following lemma then holds.

Lemma 6.1 Let m be an RM_L-term occurring in the left-hand side of a TUC^{sp} rule, and m' a ground MS term. If there exists a ground substitution σ such that $\sigma(m) = m'$, then there exists $\langle m_1, S_1\rangle \in extract(m')$, $\langle \theta, S\rangle \in match(m_1, m)$, and $\langle m_2, S_2\rangle \in extract(\theta(m))$ such that $m_2 = m_1$, and $S = S_2 \preccurlyeq S_1$.

Proof The proof is by structural induction on the structure of m and m_1.
Let m be the multiset

$$m = q_1, \ldots, q_k, n_1\langle w_1\rangle, \ldots, n_r\langle w_r\rangle \mid X$$

(every RM_L term has one and only variable) for $q_i \in \mathcal{Q}\ i : 1, \ldots, k$, and m' be the multiset

$$m' = p_1, \ldots, p_l, n'_1\langle w'_1\rangle, \ldots, n'_z\langle w'_z\rangle$$

for $p_i \in \mathcal{Q}\ i : 1, \ldots, l$.
Since, by hypothesis $\sigma(m) = m'$, then we have that $k \leq l$, $r \leq z$, $p_i = q_i$ for $i : 1, \ldots k$, $n_i = n'_i$ and $\sigma(w_i) = w'_i$ for $i : 1, \ldots, r$. Furthermore, $\sigma(X) = p_{k+1}, \ldots, p_l, n_{r+1}\langle w_{r+1}\rangle, \ldots, n_z\langle w_z\rangle$. Notice that, by definition of TUC^{sp}-rules, X cannot not occur in any subterm w'_i.
Now, by definition $extract(m')$ contains a pair $\langle m_1, S_1\rangle$, where

$$m_1 = n'_1\langle v_1\rangle, \ldots, n'_z\langle v_z\rangle \mid \bullet_u$$

where $\langle w_i, Z_i\rangle \in extract(v_i)$ for $i : 1, \ldots, z$, and $S_1 = \{\langle p_1, u\rangle, \ldots, \langle p_l, u\rangle\} \oplus \bigoplus_{i=1}^{z} Z_i$ for some $u \geq 0$.
(IH) Since $\sigma(w_i) = w'_i$, we can apply now the inductive hypothesis to infer that there exists $\langle b_i, B_i\rangle \in extract(w'_i)$, $\langle \theta_i, A_i\rangle \in match(b_i, w_i)$, and $\langle c_i, C_i\rangle \in extract(\theta_i(w_i))$ such that $c_i = b_i$, and $A_i = C_i \preccurlyeq B_i$ for $i : 1, \ldots, r$.
We can now apply the definition of $match$ to infer that there exists a pair $\langle \theta, S\rangle \in match(m_1, m)$ such that

$$\begin{array}{l}\theta = \{X \mapsto n_{r+1}\langle v_{r+1}\rangle, \ldots, n_z\langle v_z\rangle \mid \bullet_u\} \cup \bigcup_{i=1}^{r} \theta_i \\ S = \{\langle q_1, \bullet_u\rangle, \ldots, \langle q_k, \bullet_u\rangle\} \oplus \bigoplus_{i=1}^{r} A_i\end{array}$$

Now we have that

$$\theta(m) = q_1, \dots, q_k, n_1\langle\theta_1(w_1)\rangle, \dots, n_r\langle\theta(w_r)\rangle, n_{r+1}\langle v_{r+1}\rangle, \dots, n_z\langle v_z\rangle \mid \bullet_u$$

Finally, from (IH) and by definition of *extract*, we have $\langle m_2, S_2\rangle \in extract(\theta(m))$, where

$$m_2 = n_1\langle c_1\rangle, \dots, n_r\langle c_r\rangle, n_{r+1}\langle v_{r+1}\rangle, \dots, n_z\langle v_z\rangle \mid \bullet_u$$

and

$$S_2 = \{\langle q_1, \bullet_u\rangle, \dots, \langle q_k, \bullet_u\rangle\} \cup \bigcup_{i=1}^{r} C_i$$

By applying again IH, namely $c_i = b_i$ and $A_i = C_i \preccurlyeq B_i$ for $i : 1, \dots, r$, we have that $m_2 = m_1$ and $S = S_2 \preccurlyeq S_1$. □

The previous property shows that if a a rule $l \to r$ can be applied at a subterm m of a term t, then there exists a transition associated to the rule $l \to r$ that is fireable on the marking associated to t.
Specifically, let $t = tuc\langle m\rangle$ and suppose that there exists $l \to r$, a subterm $n\langle m' \mid rest\rangle$ of t, and a mapping σ such that $\sigma(l) = m'$.
Furthermore, let $\langle t_1, T_1\rangle \in extract(t)$, $\langle s_1, V_1\rangle \in extract(n\langle m' \mid rest\rangle)$ ($s_1 = n\langle m_1 \mid rest'\rangle \mid \bullet_{new}$), and $\langle m_1, S_1\rangle \in extract(m')$. Then t_1 is a T-place with s_1 and m_1 as subterms. Furthermore, by Lemma 6.1 we have that there exists $\langle\theta, S\rangle \in match(l, m_1)$ such that $\theta(m)$ has the same structure of m_1 and a submultiset of the leaves in S_1.
Thus, we have that $n\langle l \mid Z\rangle$ and $n\langle m' \mid rest'\rangle$ can be matched via $\theta' = \theta \cup \{Z \mapsto rest'\}$.
This implies that the transition τ extracted from $l \to r$ by considering the T-place t_1 and the subterm m', can be fired on the marking associated to t.
The first effect of firing τ is that of consuming the leaves in S. Furthermore, it changes the tree structure t_1 and produces new leaves according to the structure of $\theta(r)$.
The following lemma formalizes this property

Lemma 6.2 Let $l \to r$ be a TUC^{sp}rule such that $\sigma(l) = m$ and $\sigma(r) = m'$, $\langle m_1, S_1\rangle \in extract(m)$, and $\langle\theta, S\rangle \in match(m_1, l)$. Then, there exists $\langle m_2, S_2\rangle \in extract(m')$ and $\langle m_3, S_3\rangle \in extract(\theta(r))$ such that $Pos(m_1) \preccurlyeq Pos(m_3) = Pos(m_2)$, $m_3 = m_2$ and $S_3 \preccurlyeq S_2$.

Proof The proof is by induction on the structure of l and r. □

Thus, when fired on a marking in $marking(t)$ such that the T-place t_1 is marked, the transition τ puts a token in the T-place obtained by replacing m_1 with m_3. Furthermore, it subtracts a token to every place in S and it adds a token to every place in S_3. These last steps corresponds to subtracting the leaves in m_1 and adding the leaves in m_3 so as to obtain term $\sigma(r)$. The aforementioned properties give us the following proposition.

Proposition 6.1 Let N be the Petri net associated to the reachability problem $t_0 \overset{*}{\Rightarrow} t_f$ and $\mathcal{R}$. Then, $t_0 \overset{*}{\Rightarrow} t_f$ holds if and only if there exist $M \in marking(t_0)$ and $M' \in marking(t_f)$ such that M' is reachable from M in N.

Proposition 6.1 allows us to reduce reachability in a TUC^{sp}-theory to a finite set of reachability problems for a Petri net. Since the latter problem is decidable [May84, Rao82], we obtain the following result.

Theorem 6.1 Reachability is decidable in TUC^{sp}.

7. Reachability analysis of fragments of ppMA in TUC^{sp}

In this section, we will exploit the results obtained for reachability in TUC^{sp}-theories in order to give a uniform presentation of positive results for reachability in fragments of ppMA taken from the literature [BT05, BZ05a].

As investigated in [BT05, BZ05a], the undecidability of the reachability problem in ppMA has two different sources: the use of the *open* capability and the use of the congruence $!P \equiv !P \mid P$. To overcome this problem, in [BT05] Boneva and Talbot proposed a weaker semantics for the open-free fragment of ppMA: the original semantics of $!P$ is replaced by the oriented reduction rule

$$!P \twoheadrightarrow P \mid !P$$

thus dropping the *absorbtion* law $!P \mid P \twoheadrightarrow !P$ from the calculus. In [BT05], the authors prove that reachability is decidable in this semantic fragment of open-free ppMA.

In Busi and Zavattaro [BZ05a] introduced instead a "syntactic" fragment of open-free ppMA for which reachability is decidable. In this fragment replication is allowed only when guarded by an action, i.e., in terms like $!in\ n.P$ and $!out\ n.P$. Differently from the Boneva-Talbot fragment, the semantics of replication is not modified with respect to the original MA model. In Busi and Zavattaro [BZ05b] proved also the decidability of reachability for the public BA with guarded replication and with parent-child communication defined only for finite sequences of capabilities.

Consistently with the above mentioned results, we first notice that, due to the presence of the rules $open$ and $absorbt$, the TUC-theory that models a reachability problem in ppMA of Fig. 2 is not structure preserving. The semantic restriction of Boneva-Talbot can be mimicked here by dropping $open$ and $absorbt$ from the calculus.

We now observe that the following TUC^{sp} rules naturally model the semantics of guarded replication:

$$\begin{array}{l} q_{!M.P} \to q_{!M.P} \mid q_{M.P} \\ q_{!M.P} \mid q_{M.P} \to q_{!M.P} \end{array}$$

for any $!M.P$ occurring in the set $Der(P_0)$ associated to the initial process P_0.

The syntactic restriction of Busi-Zavattaro can be mimicked by removing the rules $open$, $copyt$, and $absorbt$ from the TUC theory in Fig. 2, and by adding the new rules for $q_{!M.P}$. Furthermore, under the same assumptions taken in [BZ05b], we can use a TUC^{sp}-theory to model parent-child communication. For instance, the rule

$$n\langle q_{(x)\uparrow P} \mid X\rangle \mid q_{\langle M\rangle .R} \to n\langle T(P\{x := M\}) \mid X\rangle \mid T(R)$$

models the transmission of a finite sequence of capabilities M to a child ambient n. In all the above cases the resulting rewrite rules are structure preserving.

Therefore, we can apply Theorem 6.1 to obtain a uniform and elegant way to prove the decidability of reachability in these fragments of MA and BA. A similar reasoning can be applied for other variations of the MA model. For instance, it is immediate to check that reachability in open-free pure public SA can be encoded as TUC^{sp} reachability. Indeed, safe movement operations like safe in can be modelled via the TUC^{sp} rules

$$m\langle q_{in\ n.Q} \mid Y\rangle \mid n\langle q_{\overline{in}\ n.R} \mid Z\rangle \to n\langle m\langle T(Q) \mid Y\rangle \mid T(R) \mid Z\rangle$$

which is structure preserving.

It is interesting to notice that in TUC^{sp} we can model tree update schemes that are not present as primitive operations in MA. For instance, the rule that swaps the sons of two ambients, i.e.,

$$n\langle X\rangle \mid m\langle Y\rangle \to n\langle Y\rangle \mid m\langle X\rangle$$

is structure preserving. Thus, when such a tree update scheme is added to ppMA, it does not break the good property of the above mentioned fragments.

8. Related work

Reachability is known to be decidable for ground Term Rewriting Systems (TRSs) [DT90, MR98]. In our setting we consider however reachability problems with substitutions and rewrite rules with infinite sets of ground instances.

Ground AC TRSs are equivalent to Process Rewrite Systems (PRS) a combination of prefix rewrite systems and Petri nets introduced in [Mayr00]. TUC^{sp} seems to be not directly related to PRS. Indeed, PRS does not allow synchronization rules of the form

$$n\langle X\rangle \mid m\langle Y\rangle \to n'\langle X\rangle \mid m'\langle Y\rangle$$

which are expressible in TUC^{sp}. Using these kind of rules, it is easy to see that TUC^{sp}-rules can generate non-regular tree languages when viewing rewrite rules as grammar rules.

Despite of the non regularity of the generated languages, the decidability of reachability gives us a way to decide the membership problem of a given term t in the set of terms reachable from a certain term via a set of TUC^{sp} rewrite rules. This property seems a distinguished feature from decidability results of TRSs based on tree automata like those obtained for right-linear and monadic TRSs [Sal88], linear and semi-monadic TRSs

[CDFV91], and decreasing TRSs [Jac96]. More specifically, the rule used to encode movement capabilities in Fig. 2 violate all the syntactic restrictions proposed in [Sal88, Jac96, CDFV91] since they contain a variables in a nested term in the right-hand side that also occurs in the left hand side.

The completion algorithm for tree automata presented in [FGV05] could be a useful heuristic for testing non-reachability of a given (set of) term(s).

Concerning the reachability problem for fragments of ppMA, we are only aware of the work in [BZ05a, BT05, BZ05b]. In Boneva and Talbot [BT05] propose a weaker semantics for the open-free fragment of ppMAwhere the original semantics of $!P$ is replaced by the oriented reduction rule $!P \twoheadrightarrow P \mid !P$. In the fragmnent of Mobile Ambients considered in [BZ05a] by Busi and Zavattaro replication is allowed only when guarded by an action, i.e., in terms like $!in\ n.P$ and $!out\ n.P$. In Busi and Zavattaro [BZ05b] proved also the decidability of reachability for the public BA with guarded replication and with parent-child communication defined only for finite sequences of capabilities. As discussed in detail in Sect. 7, the decidability result for TUC^{sp} gives us a uniform and more general view of these results.

Several interesting expressiveness results for Mobile Ambients, its variations, and fragments of the full model are given in the original paper of Cardelli and Gordon [CG00] and by Maffeis and Phillips in [MP05].

The present paper extends the paper published in the proceedings of ICTAC 2006 [DM06] with complete proofs of the main results (TUC vs ppMA, reachability in TUC^{sp} and Petri nets) and with expanded definitions and examples.

9. Conclusions

We have investigated the relation between fragments of Mobile Ambients and AC Term Rewriting. Our investigations show that a large class of tree update mechanisms that includes the movement capabilities of Mobile Ambients can be naturally expressed in simple fragments of Term Rewriting in which it is possible to decide important computational properties like reachability. Although in this paper, we have focused our attention to the relation between term rewriting and the model of Mobile Ambients, it could be very interesting to further investigate in the direct use of languages like TUC for the specification of computational models with hierarchical structures. Indeed, it is important to remark that term rewriting allows us to move well beyond the *in* and *out* movement operations of Mobile Ambients. The use of term variables allows us to define arbitrary movement operations on subterms. As an example, the rule

$$n\langle X\rangle \mid m\langle Y\rangle \rightarrow n\langle Y\rangle \mid m\langle X\rangle$$

can be used to swap the subtrees of two given nodes. This operation cannot be executed (in one step) in Mobile Ambients. Interestingly, this kind of tree operation is still expressible in the good fragment of TUC, i.e., TUC^{sp}. We also believe that theoretical results on Petri net can be applied to extend the decidability result for TUC^{sp} to other properties like coverability and to more general term rewriting rules, e.g., with several multiset variables at the same level in a tree term (i.e. to mix different subtrees inside the same level).

On the practical side, we are currently investigating two main directions: the use of tools for manipulating AC Term Rewriting Systems for simulation and analysis of specifications given in Mobile and BioAmbients, and of approximation techniques based on unfoldings for exploiting in an effective way the reduction of reachability in TUC^{sp} to Petri net reachability.

Acknowledgments

We thank Cristiano Calcagno for fruitful discussions and the anonymous reviewers of ICTAC 2006 for suggestions and criticism that helped us to improve the paper.

Appendix A: Basics of Petri Nets

A Petri net is a pair (P, T) where P is the set of *places* and T is the set of *transitions* ($T \subseteq \mathcal{P}_{fin}(P) \times \mathcal{P}_{fin}(P)$). If both P and T are finite then also the P/T net is finite.
Markings is the term used to identify multiset over P. Assume a marking m and a place p, we say that the place p contains $\mu_m(p)$ *tokens*.

A P/T system is a triple $N = (P, T, m_0)$ where (P, T) is a P/T net and m_0 is the initial marking.
A *transition* $t = (i, o)$ (usually written $i \rightarrow o$) is enabled in the current marking m if $i \preccurlyeq m$ (where $\preccurlyeq$ denotes the multiset inclusion) and its execution (usually written $m \xrightarrow{t} m'$) produces a new set of marking $m' = (m \setminus i) \oplus o$ (where $\oplus$ denotes the multiset union).
Finally, we say that m' is reachable from m_0 if there exists a sequence of transitions $t_1, \ldots, t_n$ amd markings $m_1, \ldots, m_n$ such that $m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots m_{n-1} \xrightarrow{t_n} m_n$ and $m_n = m'$. Given a Petri net $\mathcal{N}$ with initial marking m_0 and a marking m, the reachability problem consists in checking if m is reachable from m_0 by firing a finite sequence of transitions of $\mathcal{N}$. This problem has been proved to be decidable in [May84, Rao82].

References

[BT05] Boneva I, Talbot J-M (2005) When ambients cannot be opened. Theor Comput Sci 333(1–2):127–169
[BCC01] Bugliesi M, Castagna G, Crafa S (2001) Boxed ambients. In: Proceeding of TACS 2001, pp 38–63
[BZ05a] Busi N, Zavattaro G (2005) Deciding reachability in mobile ambients. In: Proceedings of ESOP 2005, pp 248–262
[BZ05b] Busi N, Zavattaro G (2005) Deciding reachability in boxed ambients. In: Proceedings of ICTCS 2005, pp 143–159
[CG00] Cardelli L, Gordon AD (2000) Mobile ambients. Theor Comput Sci 240(1):177–213
[CDFV91] Coquidé J-L, Dauchet M, Gilleron R, Vágvölgyi S (1991) Bottom-Up tree pushdown automata and rewrite systems. In: Proceedings of RTA 1991, pp 287–298
[DM06] Delzanno G, Montagna R (2006) Reachability analysis of mobile ambients in fragments of AC term rewriting. In: Proceedings of ICTAC 2006. pp 302–316
[DT90] Dauchet M, Tison S (1990) The theory of ground rewrite systems is decidable. In: Proceedings of LICS 1990, pp 242–248
[FGV05] Feuillade G, Genet T, Viêt Triêm Tông V (2005) Reachability analysis over term rewriting systems. Theor Comput Sci 330(3):501–551
[Jac96] Jacquemard F (1996) Decidable approximations of term rewriting systems. In: Proceedings of RTA 1996, pp 362–376
[LS00] Levi F, Sangiorgi D (2003) Mobile safe ambients. ACM Trans Program Lang Syst 25(1):1–69
[MP05] Maffeis S, Phillips I (2005) On the Computational Strength of Pure Ambient Calculi. Theor Comput Sci 330(3):501–551
[May84] Mayr EW (1984) An algorithm for the general petri net reachability problem. SIAM J Comput
[MR98] Mayr R, Rusinowitch M (1998) Reachability is decidable for ground AC rewrite systems. In: Proceedings of Infinity 1998
[Mayr00] Mayr R (2000) Process rewrite systems. Inf Comput 156(1-2):264–286
[Rao82] Rao Kosaraju S (1982) Decidability of reachability in vector addition systems. In: Proceeding of Fourteenth Annual ACM Symposium on Theory of Computing, pp 267–281
[Rei85] Reisig W (1985) Petri nets—an introduction, eatcs monographs on theoretical computer science, vol 4. Springer, Heidelberg
[Sal88] Salomaa K (1988) Deterministic tree pushdown automata and monadic tree rewriting systems. J Comput Syst Sci 37(3): 367–394

Received 3 May 2007
Accepted in revised form 3 April 2008 by K. Barkaoui, M. Broy, A. Cavalcanti and A. Cerone
Published online 7 May 2008