Simplification and inclusion of unranked tree automata

LC

May 1, 2019

Abstract

We show simplification techniques for finite automata on (ordered) unranked trees. We also provide optimized algorithms for important problems like universality and inclusion based on subsumption techniques. This has applications in checking containment of Relax NG specifications.

1 Introduction

The introduction of the shuffle and the shuffle closure operation leads to undecidability of the inclusion problem.

Idea:

1. Translate Relax NG into hedge automata.
2. Convert hedge automata to stepwise automata, in order to allow sharing of common parts.
3. Aggressively simplify stepwise automata.
4. Convert simplified stepwise automata into ranked automata, and check universality/inclusion using known methods.
5. Check universality/inclusion directly on stepwise automata.

The encoding into ranked automata can be performed either by a first-child/next-sibling encoding or by currying.

Idea: Use top-down lookahead simulation with different lookahead in different branches (more lookahead in the horizontal level).

Another application: validating translators: create a random RNG schema, convert to DTD, convert back to RNG and check equivalence.

The question is whether representing Relax NG as stepwise automata brings any benefit w.r.t. encoding into ranked automata.

If one uses automata to represent types [4], then the subtyping relation amounts to a language inclusion check.

We do not discuss neither shuffle regular expressions nor attribute-element constraints. While these facilities do not increase the expressive power of Relax NG, they do make these specifications more succinct, thus increasing the complexity of checking inclusion. Attributes are unordered, while elements are ordered: Common generalization: Unranked trees with a partially commutative alphabet.

Application of inclusion: checking exhaustiveness of pattern matching in XDuce.

Directions. Patterns and marking tree automata. Intersection types and alternating tree automata: Study removing transitions from alternating finite tree automata.

Other topic: simulation-based simplification of top-down tree transducers. Cf. also forest transducers (directly handling unranked trees).

Type-checking problem for top down tree transducers (EXPTIME-complete), or even macro tree transducers. Backward type inference.

Literature review. [4] proposes XDuce (cf. also [3]), an explicit top-down algorithm for non-deterministic ranked tree automata, thanks to an simple but elegant observation. They also give an algorithm converting XML Schemas to automata (cf. also [2]). External types vs. internal types: expressions

[1] proposes an (inefficient) inclusion checking algorithm via complementation.

[7] discusses bottom-up inclusion checking of non-deterministic ranked tree automata using semi-implicit techniques (BDDs).

[5] notion of subsumption between types (compare with simulation?).

Hedge automata [6].

Universality/language inclusion for unranked trees can be translated to language inclusion for ranked trees.

2 Stepwise automata

Let Σ be a finite alphabet. A *nondeterministic stepwise automaton* is a tuple $\mathcal{A} = (\Sigma, Q, I, \Delta_0, \Delta)$ where Q is a finite set of states, of which those in $I \subseteq Q$ are *initial*, $\Delta_0 \subseteq \Sigma \times Q$ specifies the set of initial states for each input symbol, and $\Delta \subseteq Q \times Q \times Q$ is the nondeterministic transition relation. We write $p \xrightarrow{q} r$ instead of $(p, q, r) \in \Delta$, and similarly we write $a \xrightarrow{\varepsilon} p$ (or just $a \to p$) instead of $(a, p) \in \Delta_0$, which is extended to nonempty sequences of states of the form $wq \in Q^*$ as: $a \xrightarrow{wq} p$ if there exists an intermediate state r s.t. $a \xrightarrow{w} r$ and $r \xrightarrow{q} p$. The *a-language* $\mathcal{L}_a(p)$ of a state p is the set of words $w \in Q^*$ s.t. $a \xrightarrow{w} p$. For a tree $a(t_1, \ldots, t_n)$ we write $t \to p$ if there are states $p_1, \ldots, p_n$ s.t. $t_1 \to p_1, \ldots, t_n \to p_n$ and $p_1 \cdots p_n \in \mathcal{L}_a(p)$. The *language* $\mathcal{L}(p)$ of a state p is the set of trees t s.t. $t \to p$, and the language of a NSA $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \bigcup_{p \in I} \mathcal{L}(p)$.

Proposition 1. *A tree $t = a(t_1, \ldots, t_n)$ is in $\mathcal{L}(p)$ if, and only if, either*

- $n = 0$ *and* $a \to p$, *or*
- $n > 0$ *and there exists a transition* $r \xrightarrow{q} p$ *s.t.* $a(t_1, \ldots, t_{n-1}) \in \mathcal{L}(r)$ *and* $t_n \in \mathcal{L}(q)$.

Downward simulation. A *downward simulation* $\sqsubseteq_{\mathsf{dw}} \subseteq Q \times Q$ is the largest relation on states s.t., whenever $p \sqsubseteq_{\mathsf{dw}} q$, we have

1. For every input symbol $a \in \Sigma$, if $a \xrightarrow{\varepsilon} p$, then $a \xrightarrow{\varepsilon} q$.
2. For every transition $p'' \xrightarrow{p'} p$ there exists a transition $q'' \xrightarrow{q'} q$ s.t. $p' \sqsubseteq_{\mathsf{dw}} q'$ and $p'' \sqsubseteq_{\mathsf{dw}} q''$.

As expected, downward simulation entails inclusion of both the language of trees and the a-languages of states.

Lemma 1. *If* $p \sqsubseteq_{\mathsf{dw}} q$, *then* $\mathcal{L}(p) \subseteq \mathcal{L}(q)$ *and* $\mathcal{L}_a(p) \subseteq \mathcal{L}_a^{\downarrow}(q)$ *for every* $a \in \Sigma$.

Proof. We first prove that $\mathcal{L}_a(p) \subseteq \mathcal{L}_a^{\downarrow}(q)$ for every symbol a and states p, q s.t. $p \sqsubseteq_{\mathsf{dw}} q$. We proceed by induction on the length of words in $Q^*_{\approx}$. For the base case, let $\varepsilon \in \mathcal{L}_a(p)$, i.e., $a \xrightarrow{\varepsilon} p$. Since $p \sqsubseteq_{\mathsf{dw}} q$, there exists a transition $a \xrightarrow{\varepsilon} q$, thus $\varepsilon \in \mathcal{L}_a(q)$, as required. For the inductive step, let $wp' \in \mathcal{L}_a(p)$, i.e., $a \xrightarrow{wp'} p$. There exists an intermediate state p'' and a transition $p'' \xrightarrow{p'} p$ s.t. $a \xrightarrow{w} p''$, i.e., (1) $w \in \mathcal{L}_a(p'')$. Since $p \sqsubseteq_{\mathsf{dw}} q$, there exists a transition (2) $q'' \xrightarrow{q'} q$ s.t. $p' \sqsubseteq_{\mathsf{dw}} q'$ and $p'' \sqsubseteq_{\mathsf{dw}} q''$. By (1) and the induction hypothesis, $w \in \mathcal{L}_a^{\downarrow}(q'')$, therefore there exists a word $w' \sqsupseteq_{\mathsf{dw}} w$ s.t. (3) $w' \in \mathcal{L}_a(q'')$. By (2) and (3), $w'q' \in \mathcal{L}_a(q)$, and thus $wq \in \mathcal{L}_a^{\downarrow}(q)$, as required.

We now prove $\mathcal{L}(p) \subseteq \mathcal{L}(q)$ for every pair of states p, q s.t. $p \sqsubseteq_{\mathsf{dw}} q$. We proceed by induction on the height of trees. Let $t = a(t_1, \ldots, t_n) \in \mathcal{L}(p)$, that is, $t \to p$. There are states $p_1, \ldots, p_n$ s.t. $t_1 \in \mathcal{L}(p_1), \ldots, t_n \in \mathcal{L}(p_n)$ and $p_1 \cdots p_n \in \mathcal{L}_a(p)$. Since $p \sqsubseteq_{\mathsf{dw}} q$ and the first part of the proof, there are states $q_1, \ldots, q_n$ s.t. (2) $p_1 \sqsubseteq_{\mathsf{dw}} q_1, \ldots, p_n \sqsubseteq_{\mathsf{dw}} q_n$ and (3) $q_1 \cdots q_n \in \mathcal{L}_a(q)$. By (2) and the induction hypothesis, $t_1 \in \mathcal{L}(q_1), \ldots, t_n \in \mathcal{L}(q_n)$. Thus, by (3) we have $t = a(t_1, \ldots, t_n) \to q$, i.e., $t \in \mathcal{L}(q)$, as required. □

Corollary 1. *If for every initial state* $p \in I_{\mathcal{A}}$ *there exists an initial state* $q \in I_{\mathcal{B}}$ *s.t.* $p \sqsubseteq_{\mathsf{dw}} q$, *then* $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$

Upward simulation. An *upward simulation* $\sqsubseteq_{\mathsf{up}} \subseteq Q \times Q$ is the largest relation on states s.t., whenever $p \sqsubseteq_{\mathsf{up}} q$, we have

1. If $p \in I$, then $q \in I$.
2. For every transition $p \xrightarrow{p'} p''$ there exists a transition $q \xrightarrow{q'} q''$ s.t. $p' \sqsubseteq_{\mathsf{dw}} q'$ and $p'' \sqsubseteq_{\mathsf{up}} q''$.
3. For every transition $p' \xrightarrow{p} p''$ there exists a transition $q' \xrightarrow{q} q''$ s.t. $p' \sqsubseteq_{\mathsf{dw}} q'$ and $p'' \sqsubseteq_{\mathsf{up}} q''$.

A *context* over Σ is a tree over $\Sigma \cup \{\Box\}$ with a unique occurrence of $\Box \notin \Sigma$ as a leaf. If s is a context and t is a tree (or a context), we denote by $s \circ t$ the tree (or context, resp.) obtained by replacing the unique hole of s with t. Consider the NSA $\mathcal{A}_\Box = (\Sigma_\Box, Q, I, \Delta_{\Box,0}, \Delta)$ over the extended alphabet $\Sigma_\Box = \Sigma \cup \{\Box_p \mid p \in Q\}$ where $\Delta_{\Box,0} = \Delta_0 \cup \{(\Box_p, p) \mid p \in Q\}$. The *context language* $\mathcal{L}_\Box(p)$ is the set of contexts t s.t. $t \circ \Box_q \in \mathcal{L}(\mathcal{A}_\Box, p)$ for some $\Box_q$. The *upward language* $\mathcal{L}_{\mathsf{up}}(p)$ of a state p is the set of contexts t s.t. $t \circ \Box_p \in \mathcal{L}(A_\Box)$.

Lemma 2. *Let $p \sqsubseteq_{\mathsf{up}} q$. For every label $a \in \Sigma$, state $p' \in Q$, and word $upv \in \mathcal{L}_a(p')$, there exists a state $q' \in Q$ and a word $xqy \in \mathcal{L}_a(q')$ s.t. $u \sqsubseteq_{\mathsf{dw}} x$, $v \sqsubseteq_{\mathsf{dw}} y$, and $p' \sqsubseteq_{\mathsf{up}} q'$.*

Proof. □

Lemma 3. *If for every transition $a \xrightarrow{\varepsilon}_{\mathcal{A}} p$ there exists a corresponding transition $a \xrightarrow{\varepsilon}_{\mathcal{B}} q$ with $\mathcal{L}_{\mathsf{up}}(p) \subseteq \mathcal{L}_{\mathsf{up}}(q)$, then $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.*

Proof. □

Lemma 4. *If $p \sqsubseteq_{\mathsf{up}} q$, then $\mathcal{L}_{\mathsf{up}}(p) \subseteq \mathcal{L}_{\mathsf{up}}(q)$.*

Proof. We proceed by induction on the depth of the unique $\Box$. If $t = \Box \in \mathcal{L}_{\mathsf{up}}(p)$, then $p \in I$, and we immediately have $t \in \mathcal{L}_{\mathsf{up}}(q)$ since there is a transition $\Box_q \to q$, and $q \in I$ by the definition of upward simulation. For the inductive step, let $t \in \mathcal{L}_{\mathsf{up}}(p)$. There exists an intermediate state p' and two contexts t', t'' s.t. $t = t' \circ t''$, $t' \in \mathcal{L}_{\mathsf{up}}(p')$, and $t'' \in \mathcal{L}_\Box(p')$. Then t'' has the form $a(t_1, \ldots, \Box, \ldots, t_n)$ with the hole at position i, and there exists a word $upv \in \mathcal{L}_{p'}()$ s.t. $t_1 \ldots t_{i-1} \in \mathcal{L}(u)$ and $t_{i+1} \ldots t_n \in \mathcal{L}(v)$. By Lemma 2, there exists a state q' and a word $xqy \in \mathcal{L}_a(q')$ s.t. $u \sqsubseteq_{\mathsf{dw}} x$, $v \sqsubseteq_{\mathsf{dw}} y$, and $p' \sqsubseteq_{\mathsf{up}} q'$. By Lemma 1, $t_1 \ldots t_{i-1} \in \mathcal{L}(x)$ and $t_{i+1} \ldots t_n \in \mathcal{L}(y)$, and thus (1) $t'' \in \mathcal{L}_\Box(q')$. By $p' \sqsubseteq_{\mathsf{up}} q'$ and the induction hypothesis, (2) $t' \in \mathcal{L}_{\mathsf{up}}(q')$. Since $t = t' \circ t''$, by (1) and (2) we conclude that $t \in \mathcal{L}_{\mathsf{up}}(q)$, as required. □

Corollary 2. *If for every transition $a \xrightarrow{\varepsilon}_{\mathcal{A}} p$ there exists a corresponding transition $a \xrightarrow{\varepsilon}_{\mathcal{B}} q$ with $p \sqsubseteq_{\mathsf{up}} q$, then $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.*

Quotienting. For an equivalence relation on states $\approx \subseteq Q \times Q$, the *quotient* NSA is defined as $\mathcal{A}_\approx = (\Sigma, Q_\approx, I_\approx, \Delta_{\approx,0}, \Delta_\approx)$ by taking as the set of states $Q_\approx = \{[q] \mid q \in Q\}$ the set of equivalence classes of Q, as the set of final states $I_\approx = \{[q] \mid q \in I\}$ the set of equivalence classes of initial states in I, and $(a, [q]) \in \Delta_{\approx,0}$ if $(a, q) \in \Delta_0$, and finally $([p], [q], [r]) \in \Delta_\approx$ if $(p, q, r) \in \Delta$. We say that an equivalence is *good for quotienting* (GFQ) if $\mathcal{L}(\mathcal{A}_\approx) = \mathcal{L}(\mathcal{A})$. For a preorder $\preceq$, we say it is GFQ if the induced equivalence $\preceq \cap (\preceq)^{-1}$ is GFQ.

Lemma 5. *Downward simulation $\sqsubseteq_{\mathsf{dw}}$ is GFQ.*

Proof. We show $[p] \sqsubseteq_{\mathsf{dw}} p$, from which the claim follows by Lemma 1. It suffices to show that the relation $[p] \preceq p'$ defined as $\hat{p} \sqsubseteq_{\mathsf{dw}} p'$ for some $\hat{p} \in [p]$ is a downward simulation relation in the quotient automaton. Notice that if

$[p] \preceq p'$, then indeed $\hat{p} \sqsubseteq_{\mathsf{dw}} p'$ *for every* $\hat{p} \in [p]$. Let $[p] \preceq p'$. If $a \xrightarrow{\varepsilon} [p]$, then by the definition of quotient there exists a transition $a \xrightarrow{\varepsilon} \hat{p}$ with $\hat{p} \in [p]$. Since $[p] \preceq p'$, we have $\hat{p} \sqsubseteq_{\mathsf{dw}} p'$, and thus there exists a transition $a \xrightarrow{\varepsilon} p'$, as required. Now let $[r] \xrightarrow{[q]} [p]$. By the definition of quotient, there are states $\hat{r} \in [r], \hat{q} \in [q], \hat{p} \in [p]$ s.t. (1) $\hat{r} \xrightarrow{\hat{q}} \hat{p}$. Since $[p] \preceq p'$, and thus $\hat{p} \sqsubseteq_{\mathsf{dw}} p$, by the definition of simulation there exists a transition $r' \xrightarrow{q'} p'$ s.t. $\hat{r} \sqsubseteq_{\mathsf{dw}} r'$ and $\hat{q} \sqsubseteq_{\mathsf{dw}} q'$, i.e., $[\hat{r}] \preceq r', [\hat{q}] \preceq q'$. Thus, $\preceq$ is a downward simulation, which concludes the proof. □

On the other hand, upward simulation is not GFQ. However, we can modify its definition to obtain a GFQ relation. A *stable upward simulation* is any relation $\sqsubseteq_{\mathsf{sup}} \subseteq Q \times Q$ s.t. whenever $p \sqsubseteq_{\mathsf{sup}} q$,

1. If $p \in I$, then $q \in I$.
2. For every transition $p \xrightarrow{p'} p''$ there exists a transition $q \xrightarrow{q'} q''$ s.t. $p'' \sqsubseteq_{\mathsf{up}} q''$ and $\hat{p} \sqsubseteq_{\mathsf{dw}} q'$ for every $\hat{p} \approx_{\mathsf{sup}} p'$.
3. For every transition $p' \xrightarrow{p} p''$ there exists a transition $q' \xrightarrow{q} q''$ s.t. $p'' \sqsubseteq_{\mathsf{up}} q''$ and $\hat{p} \sqsubseteq_{\mathsf{dw}} q'$ for every $\hat{p} \approx_{\mathsf{sup}} p'$.

Note that there is no unique maximal stable upward simulation.

Lemma 6. *Stable upward simulations* $\sqsubseteq_{\mathsf{sup}}$ *are GFQ.*

Proof. Let $[p] \preceq p'$ if $p \sqsubseteq_{\mathsf{sup}} p'$. We show that $\preceq$ is an upward simulation. □

Transition pruning. For two transitions $t = p \xrightarrow{q} r$ and $t' = p' \xrightarrow{q'} r'$, let $t \preceq t'$ if $p \sqsubseteq_{\mathsf{dw}} p'$, $q \sqsubseteq_{\mathsf{dw}} q'$, and $r \sqsubseteq_{\mathsf{up}} r'$. Then $\preceq$ is a preorder, and we consider its two smaller strict variants $\prec_1, \prec_2$ defined as $t \prec_1 t'$ if either $p \sqsubset_{\mathsf{dw}} p'$ or $q \sqsubset_{\mathsf{dw}} q'$, and $t \prec_2 t'$ if $r \sqsubset_{\mathsf{up}} r'$.

Given an automaton $\mathcal{A} = (\Sigma, P, I, \Delta_0, \Delta)$ and a strict preorder on its transitions $\prec \subseteq \Delta \times \Delta$, let $\mathsf{Prune}(\mathcal{A}, \prec)$ be the automaton $(\Sigma, P, I, \Delta_0, \Delta')$ where, for every transition $t \in \Delta$, Δ' contains a transition $t' \in \Delta$ s.t. $t \prec t'$ and t' is maximal with this property. We say that a strict preorder $\prec$ is *good for pruning* (GFP) if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{Prune}(\mathcal{A}, \prec))$.

Lemma 7. *The strict preorder* $\prec_1$ *is GFP.*

Lemma 8. *The strict preorder* $\prec_2$ *is GFP.*

3 Downward universality testing

We call a set of states $R \subseteq P$ a *macrostate*. A *clause* is a set of macrostates $C = \bigwedge_{i \in I} R_i$ interpreted conjunctively. The language $\mathcal{L}(R)$ of a macrostate R is $\bigcup_{r \in R} \mathcal{L}(R)$, and the language $\mathcal{L}(C)$ of a clause C is $\bigcap_{R \in C} \mathcal{L}(R)$. Consequently, C is universal iff every $R \in C$ is universal.

Our universality algorithm is based on the following decomposition. Let

$$\mathsf{Down}(R) = \{(p,q) \mid \exists (p \xrightarrow{q} r) \cdot r \in R\},$$

and, for a subset thereof $S \subseteq \mathsf{Down}(R)$, let $S_1 = \pi_1(S) = \{p \mid (p,q) \in S\}$ and $S_2 = \pi_2(R \backslash S) = \{q \mid (p,q) \notin S\}$.

Lemma 9. *A macrostate R is universal if, and only if,*

1. *for every $a \in \Sigma$, $a \to R$, and*
2. *for every $S \subseteq \mathsf{Down}(R)$, either S_1 or S_2 is universal.*

Proof. Let $t = a(t_1, \ldots, t_n)$. Then, $t \in \mathcal{L}(R)$ iff there exists a state $r \in \mathcal{L}(R)$ s.t. $t \in \mathcal{L}(r)$. By Proposition 1, either $n = 0$ and $a \to r$, or $n > 0$ and there exists a transition $p \xrightarrow{q} r$ s.t. $t' = a(t_1, \ldots, t_{n-1}) \in \mathcal{L}(p)$ and $t_n \in \mathcal{L}(q)$. Since t was arbitrary, this is equivalent to say that $\mathcal{L}(\mathsf{Down}(R))$ contains all pairs of trees. However,

$$\begin{aligned}
\mathcal{L}(\mathsf{Down}(R)) &= \bigcup_{(p,q) \in \mathsf{Down}(R)} \mathcal{L}(p,q) = \\
&= \bigcup_{(p,q) \in \mathsf{Down}(R)} \mathcal{L}(p) \times \mathcal{L}(q) = \\
&= \bigcup_{(p,q) \in \mathsf{Down}(R)} (\mathcal{L}(p) \times \mathcal{T}(\Sigma)) \cap (\mathcal{T}(\Sigma) \times \mathcal{L}(q)) = \\
&= \bigcap_{S \subseteq \mathsf{Down}(R)} \left(\bigcup_{(p,q) \in S} \mathcal{L}(p) \times \mathcal{T}(\Sigma) \right) \cup \left(\bigcup_{(p,q) \notin S} \mathcal{T}(\Sigma) \times \mathcal{L}(q) \right) = \\
&= \bigcap_{S \subseteq \mathsf{Down}(R)} \mathcal{L}(S_1) \times \mathcal{T}(\Sigma) \cup \mathcal{T}(\Sigma) \times \mathcal{L}(S_2).
\end{aligned}$$

Therefore, $\mathcal{L}(\mathsf{Down}(R))$ contains all pairs of trees if, for every $S \subseteq \mathsf{Down}(R)$, either S_1 or S_2 is universal. □

Since $\mathcal{A}$ is universal iff the initial macrostate I is universal, the previous characterisation immediately yields a recursive algorithm exploring an AND/OR tree of exponential size labelled with macrostates rooted at I. In order to speed up the computation, one can apply the following optimisations:

1. If a macrostate R contains two states $p, q \in R$ s.t. $\mathcal{L}(p) \subseteq \mathcal{L}(q)$, then it is safe to replace R by $S := R \backslash \{p\}$ since $\mathcal{L}(R) = \mathcal{L}(S)$.
2. If a clause C contains two macrostates $R, S \in C$ s.t. $\mathcal{L}(R) \subseteq \mathcal{L}(S)$, then we can replace C with $D := C \backslash \{S\}$, since $\mathcal{L}(C) = \mathcal{L}(D)$.

In order to implement the first two optimisations above, let $\sqsubseteq_1$ be any relation on states s.t., if $p \sqsubseteq_1 q$, then $\mathcal{L}(p) \subseteq \mathcal{L}(q)$, and let $\sqsubseteq_2$ be any relation on macrostates s.t. $R \sqsubseteq_2 S$ implies $\mathcal{L}(R) \subseteq \mathcal{L}(S)$. We keep maximal states in a

macrostate, and minimal macrostates in a clause. I.e., for a macrostate R and a clause C, let

$$\max R := \{r \in R \mid \forall s \in R \cdot r \sqsubseteq_1 s \text{ implies } s \sqsubseteq_1 r\},$$
$$\min C := \{R \in C \mid \forall S \in C \cdot S \sqsubseteq_2 R \text{ implies } R \sqsubseteq_2 S\}.$$

Moreover, we can further enhance the algorithm by maintaining a global set NU of macrostates that have already been shown to be non-universal. If a macrostate S is non-universal and $R \sqsubseteq_2 S$ then R is non-universal as well, therefore it suffices to maintain only $\sqsubseteq_2$-maximal macrostates in NU. Notice that the macrostate

$$J := \{p \in Q \mid \exists a \in \Sigma \cdot \neg(a \to p)\}$$

is clearly non-universal and it can be used to initialise the set NU. With a little preprocessing one can quickly build an antichain of non-universal states. For a set of pairs of macrostates $S \subseteq Q \times Q$ let

$$\mathsf{Up}(S) = \{r \mid \forall (p \xrightarrow{q} r) \cdot (p, q) \in S\}.$$

Proposition 2. *The function* $\mathsf{Up}(\_)$ *satisfies the following properties:*

1. *If* S *is non-universal, then also* $\mathsf{Up}(S)$ *is non-universal.*
2. *For any set of states* R, $R \subseteq \mathsf{Up}(\mathsf{Down}(R))$.
3. *If* S *is* $(\sqsubseteq_1 \times \sqsubseteq_1)$*-downward closed, then also* $\mathsf{Up}(S)$ *is* $\sqsubseteq_1$*-downward closed.*

As a combination of the first two points above, we have that, if R is not universal, then the coarser $\mathsf{Up}(\mathsf{Down}(R))$ is not universal either. The third point allows us to store just $\sqsubseteq_1$-maximal elements Since the $J \times J$ is $(\sqsubseteq_1 \times \sqsubseteq_1)$-downward closed, it suffices to store the $\sqsubseteq_1$-maximal elements in macrostates in NU. We can thus use the following function $\widehat{\mathsf{Up}}(\_)$ manipulating only maximal elements:

$$\widehat{\mathsf{Up}}(S) = \max\{r \mid \forall (p \xrightarrow{q} r) \cdot \exists (p', q') \in S \cdot p \sqsubseteq_1 p' \text{ and } q \sqsubseteq_1 q'\}.$$

Finally, we can also store a set of macrostates UN which have already been proved to be universal. It thus suffices to keep $\sqsubseteq_2$-minimal elements in this set, and it can be used to prune the search even more.

4 Upward universality testing

5 Encoding unranked trees as ranked trees

There are two standard ways to encode an unranked tree into a ranked—in fact, binary—while preserving recognisability.

Algorithm 1: Determine whether the automaton is universal.

Input: An NSA $\mathcal{A} = (\Sigma, Q, I, \Delta_0, \Delta)$.
Output: **true** iff $\mathcal{A}$ is universal.

```
1  NU := {max(J)} = {max{p ∈ Q | ∃a ∈ Σ · ¬(a → p)}};
2  UN := {};
3  return IsUniversal(max(I), {});

4  Function IsUniversal(R, C)
5      if ∃S ∈ C ∪ UN · S ⊑₂ R then return true;
6      if ∃S ∈ NU · R ⊑₂ S then return false;
7      C := min(C ∪ {R});
8      foreach S ⊆ max(Down(R)) do
9          S₁ := max(π₁(S));
10         if ¬IsUniversal(S₁, C) then
11             S₂ := max(π₂(R\S));
12             if ¬IsUniversal(S₂, C) then
13                 NU := max(NU ∪ {R});
                   /* Or the coarser R ⊆ Up^(max(Down(R)))   */
14                 return false;
15     UN := min(UN ∪ {R});
16     return true;
```

First-child next-sibling encoding. For every $a \in \Sigma$, we interpret it as a ranked binary symbol a, and we have additionally have a leaf symbol ε. We encode a sequence of trees $u = t_1, \cdots, t_n$ as a binary tree as follows:

$$\begin{aligned} \mathsf{fcns}(\varepsilon) &= \varepsilon, \\ \mathsf{fcns}(a(u), v) &= a(\mathsf{fcns}(u), \mathsf{fcns}(v)). \end{aligned}$$

Currying encoding. For every $a \in \Sigma$ we interpret it as a leaf symbol a, and we have an additional binary symbol @.

$$\begin{aligned} \mathsf{curry}(a) &= a, \\ \mathsf{curry}(a(t_1, \ldots, t_n)) &= @(\mathsf{curry}(a(t_1, \ldots, t_{n-1})), \mathsf{curry}(t_n)) \quad \text{if } n \geqslant 1. \end{aligned}$$

References

[1] H. Hosoya and M. Murata. Boolean operations for attribute-element constraints. In *Proc. of CIAA'03*, CIAA'03, pages 201–212, Berlin, Heidelberg, 2003. Springer-Verlag.

[2] H. Hosoya and B. C. Pierce. Regular expression pattern matching for xml. *J. Funct. Program.*, 13(6):961–1004, Nov. 2003.

[3] H. Hosoya and B. C. Pierce. Xduce: A statically typed xml processing language. *ACM Trans. Internet Technol.*, 3(2):117–148, May 2003.

[4] H. Hosoya, J. Vouillon, and B. C. Pierce. Regular expression types for xml. *ACM Trans. Program. Lang. Syst.*, 27(1):46–90, Jan. 2005.

[5] G. M. Kuper and J. Siméon. Subsumption for xml types. In *Proc. of ICDT'01*, ICDT '01, pages 331–345, London, UK, UK, 2001. Springer-Verlag.

[6] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of xml schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4):660–704, Nov. 2005.

[7] A. Tozawa and M. Hagiya. Xml schema containment checking based on semi-implicit techniques. In *Proc. of CIAA'03*, CIAA'03, pages 213–225, Berlin, Heidelberg, 2003. Springer-Verlag.