One-variable context-free hedge automata

Florent Jacquemard, Michael Rusinowitch

To cite this version:

Florent Jacquemard, Michael Rusinowitch. One-variable context-free hedge automata. Journal of Computer and System Sciences, Elsevier, 2016, 10.1016/j.jcss.2016.10.006 . hal-01426626

HAL Id: hal-01426626

https://hal.inria.fr/hal-01426626

Submitted on 4 Jan 2017

One-Variable Context-Free Hedge Automata[☆]

Florent Jacquemard[a,*], Michael Rusinowitch[b,*]

[a]*INRIA Paris–Rocquencourt & Ircam UMR – 1 place Igor Stravinsky, 75004 Paris, France.*
[b]*INRIA Nancy–Grand Est & LORIA UMR*
615 rue du Jardin Botanique, 54602 Villers-les-Nancy, France.

Abstract

We introduce an extension of hedge automata called One-Variable Context-Free Hedge Automata. The class of unranked ordered tree languages they recognize has polynomial membership problem and is preserved by rewrite closure with inverse-monadic rules. We also propose a modeling of primitives of the W3C XQuery Update Facility by mean of parameterized rewriting rules, and show that the rewrite closure of a context-free hedge language with these extended rewriting systems is a context-free hedge language. This result is applied to static analysis of XML access control policies expressed with update primitives.

Keywords: Hedge automata, rewrite closure, verification, static type checking, XQuery Update, access control.

Introduction

The verification of systems and programs involving tree-like structures motivates the study of rewrite closures for various kind of term rewriting systems (TRS) and the identification of computable cases. Automata for ranked trees (terms over a signature) provide a good formalism for characterizing the closure of languages *w.r.t.* term rewriting systems. Indeed, tree-automata (TA) transitions can be defined as rewrite rules [1], given a TRS $\mathcal{R}$ and an initial TA $\mathcal{A}$, the rewrite closure of the language of $\mathcal{A}$ by $\mathcal{R}$ can be computed in many interesting cases by *tree automata completion*, a procedure that iteratively computes critical pairs between the TA transitions (which are rewrite rules) of $\mathcal{A}$ and the TRS rules of $\mathcal{R}$.

Unranked trees are natural representations for XML documents, where the number of children of a node is not fixed. Hedge Automata (HA) [2] embeds most of the formalisms [3] for specifying the structural information (or type) of XML documents. Some procedures have been proposed for the construction of rewrite closure of HA languages [4]. There are however two problems in the context of unranked trees. The first one is technical, and stems from the different natures of rewrite rules and HA transitions. Indeed, performing automata computations in unranked trees requires vertical navigation from children to parent (in case

[☆]This work has been partly supported by the grant ANR-12-CORD-0009 "Inedit".

[*]Corresponding author

Email addresses: `florent.jacquemard@inria.fr` (Florent Jacquemard), `michael.rusinowitch@inria.fr` (Michael Rusinowitch)

of bottom-up computations) and also horizontal navigation between children, since their number is unbounded. The original HA is an hybrid model where horizontal and vertical navigation rules are not specified in the same way, and tree automata completion is more difficult to define in this case than for ranked tree automata. The second difficulty is that rewrite closures are often non regular in the case of unranked trees, *i.e.* they cannot be captured by HA. For instance the closure of HA languages by a flat linear unranked TRS is not always HA whereas the closure of TA languages by flat linear TRS is TA [5]. There are mainly two solutions to this problem: the construction of (over)-approximations (as in [4]) or the extension of the HA model.

Regarding applications of these results, one may cite for instance regular tree model checking, type checking of XML transformations, and consistency verification of access control policies (ACP) for XML updates. We briefly survey these three typical applications.

In formal verification of infinite state systems, several regular model checking approaches represent sets of configurations by regular languages, transitions by rewrite rules and (approximations of) reachable configurations as rewrite closure of regular languages [6]. *Regular tree model checking* extends the approach from string to tree regular languages and string rewriting to term rewriting (see *e.g.* [7, 8]). It has been extended later from ranked tree to hedge rewriting and hedge automata in [9]. Regular tree model checking has been applied to the verification of programs with recursive calls and dynamic thread creation.

A central problem in XML document processing is *static typechecking* [10]. This problem amounts to verifying at compile time that every output XML document which is the result of a specified query or transformation applied to an input document with a valid input type has a valid output type. In general, input and output type are defined as regular tree languages. Being able to compute the closure or backward closure (under transformations) of such languages hence permits to solve the problem see *e.g.* [11].

A large amount of work has been devoted to secure XML querying and transformations. But most of the work focuses on read-only rights, and very few have considered update rights for a model based on standard XML update operations such as those of the XQuery Update Facility (XQUF) [12]; one may cite [13, 14, 15]. The sensitive problem of access control policy inconsistency is, *whether a forbidden operation can be simulated through a sequence of allowed operations.* For instance [15] presents a hospital database example where it is forbidden to rename a patient name in a medical file but the same effect can be obtained by deleting this file and inserting a new one. This example illustrates a so-called *local inconsistency* problem and its detection can be solved by rewrite closure computations.

Contributions. In this paper, we introduce an automata model for unranked trees generalizing HA and whose transitions are uniformly presented as rewrite rules. This model, called CF^{1}HA, is shown decidable and more expressive than previous ones. It permits us to capture significant classes of rewrite systems with computable closures, such as inverse-monadic rules, that are more general than the rules considered in [16]. Moreover, the representation of transitions as rewrite rules enables simpler constructions than in the case of HA [16, 15].

This implies that we can compute exact reachability sets when the configuration sets are represented by CF^{1}HA hence beyond the regular (HA) ones. These results are therefore interesting for automated verification where reachability sets are not always regular.

As an application we consider a model for XML update primitives of XQUF [12] as parameterized rewriting rules of the form: "insert an unranked tree from a regular tree language L as the first child of a node labeled by a". This model extends an initial proposal of [15] with (in particular) the possibility to insert a new parent node above a given node. We show how to compute the rewrite closure of HA languages with these extended rewriting systems. These rewrite closure results can be applied directly to the static analysis of access control policies (ACP) for XML updates. ACPs can be specified with two sets of atomic update primitives listing the operations that are allowed, resp. forbidden, to a user [13, 14]. We show how to verify the local consistency of ACPs, *i.e.* whether no sequence of allowed updates starting from a given document can achieve an explicitly forbidden update. Such situations may lead to serious security breaches which are challenging to detect according to [13].

We also consider (in Section 5.2) an extension of rewrite systems to suit the case of XML documents developed under DTDs (following the DDML/XSchema methodology), and we obtain in that case a negative result for closure computation, namely that reachability is undecidable even for non-recursive DTD, and therefore there is no hope to extend our construction in this direction.

Roadmap. Preliminary notions such as hedge rewriting systems are introduced in Section 1. We present our extension of hedge automata called *1-variable bidimensional context-free hedge automaton* ($\mathsf{CF}^1\mathsf{HA}$ in short) in Section 2. In Subsection 2.1 we present closure and decision properties for the class of languages they recognize. In Subsection 2.2 we show the correctness of the membership algorithm we have given for $\mathsf{CF}^1\mathsf{HA}$. The $\mathsf{CF}^1\mathsf{HA}$ are compared to known classes of automata on unranked trees in Subsection 2.3. In Section 3 we introduce 1-childvar inverse monadic hedge rewriting systems. We prove in Subsection 3.1 that these hedge rewriting systems preserve $\mathsf{CF}^1\mathsf{HA}$ languages. In Subsection 3.2 we show that hedge automata languages are preserved by backward closure application of inverse monadic systems. In Section 4 we show how XQuery Update Facility primitives can be directly modeled using a subclass of parameterized hedge rewriting rules called uPHRS. In Subsection 4.1 we introduce a subclass of uPHRS that admit no looping sequence of renamings. This sublass is used to simplify the computation, in Subsection 4.2, of the forward rewrite closure for uPHRS, using automata completion techniques. In Subsection 4.3 we give an application to typechecking XML updates. In Section 5 we study some models of Access Control Policies (ACP) for the update operations defined in Section 4, and the verification problems for these ACP. In Subsection 5.1 we consider ACP where update rules are divided into authorized and forbidden operations. In Subsection 5.2 we consider ACP defined by extending DTD with security annotations as in [17, 13]. We conclude by sketching future works in Section 6.

Related work. A detailed comparison of $\mathsf{CF}^1\mathsf{HA}$ with other unranked tree automata models can be found in Section 2.3. Many works have employed tree automata to compute sets of descendants for standard (ranked) term rewriting (see *e.g.* [7]). Regular model checking [6] is extended to hedge rewriting and hedge automata in [9], which gives a procedure to compute reachability sets *approximations.* Here we compute exact reachability sets for some classes of hedge rewrite systems.

[18] (see also [**?**]) presents a static analysis of XML document adaptations, expressed as sequences of XQUF primitives. The authors also use an automatic inference method for deriving the type, expressed as a HA, of a sequence of document updates. The type is computed starting from the original schema and from the XQuery Updates formulated as rewriting rules as in [15]. However differently from our case the updates are applied in parallel in one shot.

The first access control model for XML was proposed by [19] and was extended to secure updates in [20]. Static analysis has been applied to XML Access Control in [21] to determine if a query expression is guaranteed not to access to elements that are forbidden by the policy. In [13] the authors propose the XACU language. They study policy consistency and show that it is undecidable in their setting. On the positive side [22] considers policies whose allow/deny permissions are tied to a DTD and gives a PTIME algorithm for checking consistency, as well as analysis/heuristics for the repair problem.

This paper differs significantly from our conference paper [23] in many aspects. Compared to [23] we simplify our model by allowing only one variable in automata transition rules but this allows one to prove that the membership problem becomes PTIME in this new model (with a CYK-like algorithm [24]). We show the new result that our closure construction cannot be extended to take into account DTD, due to undecidability of closure computations in this case. We apply our positive result to the consistency verification of ACP with update primitives, an application that is not considered in [23].

1. Preliminaries

We consider a finite alphabet Σ and an infinite set of variables $\mathcal{X}$. The symbols of Σ are generally denoted $a, b, c \ldots$ and the variables x, $y \ldots$ The sets of *hedges* and *trees* over Σ and $\mathcal{X}$, respectively denoted $\mathcal{H}(\Sigma, \mathcal{X})$ and $\mathcal{T}(\Sigma, \mathcal{X})$, are defined recursively as the smallest sets such that: every $x \in \mathcal{X}$ is a tree, if $t_1, \ldots, t_n$ is a finite sequence of trees (possibly empty), then $t_1 \cdots t_n$ is a hedge and if h is a hedge and $a \in \Sigma$, then $a(h)$ is a tree. The empty hedge (case $n \geq 0$ above) is denoted by ε and the tree $a(\varepsilon)$ will be simply denoted by a. We use the operator $\cdot$ to denote the hedge concatenation. It is associative and ε is its unit element. We will sometimes consider a tree as a hedge of length one, *i.e.* consider that $\mathcal{T}(\Sigma, \mathcal{X}) \subset \mathcal{H}(\Sigma, \mathcal{X})$. The sets of ground trees (trees without variables) and ground hedges are respectively denoted $\mathcal{T}(\Sigma)$ and $\mathcal{H}(\Sigma)$.

The root of the tree $a(h)$ is its top node labeled by a. A root (resp. leaf) of a hedge $h = (t_1 \cdots t_n)$ is a root node (resp. leaf node, *i.e.* node without child) of one of the trees $t_1, \ldots, t_n$. The root node of the tree $a(h)$ is called the *parent* of every root of h and every root of h is called a *child* of the root of $a(h)$. A *path* is a sequence of nodes $p_0, \ldots, p_k$, $k \geq 0$, such that for all $0 \leq i < k$, p_{i+1} is a child of p_i. In this case, p_k is called a *descendant* of p_0. As usual, we can see a hedge $h \in \mathcal{H}(\Sigma, \mathcal{X})$ as a function from its set of nodes $\mathit{dom}(h)$ into labels in $\Sigma \cup \mathcal{X}$. The label of the node $p \in \mathit{dom}(h)$ is denoted by $h(p)$. The subtree of a hedge h at node $p \in \mathit{dom}(h)$ is denoted $h|_p$.

The set of variables occurring in a hedge $h \in \mathcal{H}(\Sigma, \mathcal{X})$ is denoted $\mathit{var}(h)$. A hedge $h \in \mathcal{H}(\Sigma, \mathcal{X})$ is called *linear* if every variable of $\mathit{var}(h)$ occurs once in h. A *substitution* σ is a mapping of finite domain from $\mathcal{X}$ into $\mathcal{H}(\Sigma, \mathcal{X})$. The application of a substitution σ to terms and hedges (written with postfix notation) is defined recursively by $x\sigma := \sigma(x)$ when $x \in \mathit{dom}(\sigma)$, $y\sigma := y$ when $y \in \mathcal{X} \setminus \mathit{dom}(\sigma)$, $(t_1 \cdots t_n)\sigma := (t_1\sigma \cdots t_n\sigma)$ for $n \geq 0$, and $a(h)\sigma := a(h\sigma)$.

The set $\mathcal{C}(\Sigma)$ of *contexts* over Σ contains the linear hedges of $\mathcal{H}\big(\Sigma, \{x\}\big)$. The application of a context $C \in \mathcal{C}(\Sigma)$ to a hedge $h \in \mathcal{H}(\Sigma, \mathcal{X})$ is defined by $C[h] := C\{x \mapsto h\}$. It consists in inserting h in C in place of the node labelled by x. Sometimes, we write $h[s]$ in order to emphasize that s is a subhedge (or subtree) of h.

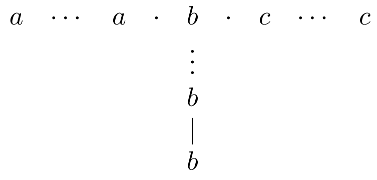
A *hedge rewriting system* (HRS) $\mathcal{R}$ over a finite unranked alphabet Σ is a set of *rewrite rules* of the form $\ell \to r$ where $\ell \in \mathcal{H}(\Sigma, \mathcal{X}) \setminus \mathcal{X}$ and $r \in \mathcal{H}(\Sigma, \mathcal{X})$; ℓ and r are respectively called left- and right-hand-side (*lhs* and *rhs*) of the rule. Note that we do not assume the cardinality of $\mathcal{R}$ to be finite. A HRS is called ground, resp. linear, if all its *lhs* and *rhs* of rules are ground, resp. linear.

The rewrite relation $\xrightarrow[\mathcal{R}]{}$ of a HRS $\mathcal{R}$ is the smallest binary relation on $\mathcal{H}(\Sigma, \mathcal{X})$ containing $\mathcal{R}$ and closed by application of substitutions and contexts. In other words, $h \xrightarrow[\mathcal{R}]{} h'$, iff there exists a context C, a rule $\ell \to r$ in $\mathcal{R}$ and a substitution σ such that $h = C[\ell\sigma]$ and $h' = C[r\sigma]$. The reflexive and transitive closure of $\xrightarrow[\mathcal{R}]{}$ is denoted $\xrightarrow[\mathcal{R}]{*}$. Given $L \subseteq \mathcal{H}(\Sigma, \mathcal{X})$ and a HRS $\mathcal{R}$, we define the *rewrite closure* of L under $\mathcal{R}$ as $\mathit{post}^*_{\mathcal{R}}(L) := \{h' \in \mathcal{H}(\Sigma, \mathcal{X}) \mid \exists h \in L, h \xrightarrow[\mathcal{R}]{*} h'\}$ and the *backward rewrite closure* of L under $\mathcal{R}$ as $\mathit{pre}^*_{\mathcal{R}}(L) := \{h \in \mathcal{H}(\Sigma, \mathcal{X}) \mid \exists h' \in L, h \xrightarrow[\mathcal{R}]{*} h'\}$.

Example 1. Let $\Sigma = \{a, b, c, d_0, d_1, d_2\}$ and let us consider the following rewrite rules over Σ

$$\mathcal{R} = \{d_0(x) \to a \cdot d_1(x), d_1(x) \to d_2(x) \cdot c, d_2(x) \to d_0(b(x)), d_2(x) \to b(x)\}.$$

Starting from $d_0 = d_0(\varepsilon)$, we have the following rewrite sequence $d_0 \to a \cdot d_1 \to a \cdot d_2 \cdot c \to a \cdot d_0(b) \cdot c \to a \cdot a \cdot d_1(b) \cdot c \to a \cdot a \cdot d_2(b) \cdot c \cdot c \to a \cdot a \cdot d_0(b(b)) \cdot c \cdot c \to \dots$ We can observe that the trees of the rewrite closure of d_0 under $\mathcal{R}$ which do not contain the symbols d_0, d_1, d_2 is the set of trees of the form $a \cdots a \cdot b(\dots b(b)) \cdot c \cdots c$ with the same number of a, b and c. We call them T-patterns since they have the shape of character T:



2. One-Variable Bidimensional Context-Free Hedge Automata

We consider a model of automata computing of unranked trees by means of reduction with rewrite rules similar to (inverse) productions of CF grammars, applied in horizontal or vertical directions in trees.

Definition 1. A *1-variable bidimensional context-free hedge automaton* ($\mathsf{CF}^1\mathsf{HA}$) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$ where Σ is a finite unranked alphabet, Q is a finite set of states disjoint from Σ, $Q^{\mathsf{f}} \subseteq Q$ is a set of final states, and Δ is a set of rewrite rules of one of the following form, where $p \in Q \cup \Sigma$, $q_1, q_2, q \in Q$:

$$\begin{array}{rclcrclcrcl} \varepsilon & \to & q(\varepsilon) & \quad & q_1(x) \cdot q_2 & \to & q(x) & \quad & q_1\big(q_2(x)\big) & \to & q(x) \\ p(x) & \to & q(x) & & q_1 \cdot q_2(x) & \to & q(x) & & & & \end{array}$$

The rules of the second column are called *horizontal* transitions, and those of the last column are called *vertical* transitions. By convention, $q(\varepsilon)$ in the first rule will be simply written q below.

The move relation $\xrightarrow[\mathcal{A}]{}$ between ground hedges of $\mathcal{H}(\Sigma \cup Q)$ is defined as the rewrite relation defined by Δ. The language of a $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A}$ in one of its states q, denoted by $L(\mathcal{A}, q)$, is the set of ground hedges $h \in \mathcal{H}(\Sigma)$ such that $h \xrightarrow[\mathcal{A}]{*} q$ (we recall that q stands for $q(\varepsilon)$). A hedge h is accepted by $\mathcal{A}$ if there exists $q \in Q^{\mathsf{f}}$ such that $h \in L(\mathcal{A}, q)$. The language of $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of hedges accepted by $\mathcal{A}$.
In the following, for the sake of conciseness, we should also consider horizontal and vertical transitions of the form

$$\begin{array}{rclcrcl} p_1(x) \cdot p_2 & \to & q(x) & \quad & p_1\big(p_2(x)\big) & \to & q(x) \\ p_1 \cdot p_2(x) & \to & q(x) & & & & \end{array}$$

where $p_1, p_2 \in Q \cup \Sigma$ and $q \in Q$. This does not increase the expressiveness of $\mathsf{CF}^1\mathsf{HA}$, as it is possible to simulate such rules with a linear number of additional states and rules of Definition 1 of the form $a \to q$ or $a(x) \to q(x)$.
Moreover, it is possible to force the variable x in the transitions in Definition 1 to be equal to ε, *i.e.* to consider transitions of the form ($p_1, p_2 \in Q \cup \Sigma$, $q \in Q$)

$$p_1 \ \to \ q \qquad p_1 \cdot p_2 \ \to \ q \qquad p_1\big(p_2\big) \ \to \ q$$

This does not change the expressiveness of the model. We can indeed assume, without loss of generality, a unique state q_ε such that there is a transition $\varepsilon \to q_\varepsilon$ and that q_ε does not occur in *lhs* of horizontal transitions. With every symbol $p \in \Sigma \cup Q$, we associate a copy p^ε and a new transition $p(q_\varepsilon(x)) \to p^\varepsilon(x)$. Then $p_1 \to q$ corresponds to $p_1^\varepsilon(x) \to q(x)$, $p_1 \cdot p_2 \to q$ to $p_1^\varepsilon(x) \cdot p_2 \to q(x)$, and $p_1(p_2) \to q$ to $p_1(p_2^\varepsilon(x)) \to q(x)$. More generally, we also accept horizontal rules of the form $p_1 \cdots p_n \to q$ ($n \geq 0$).

Example 2. The language of T-patterns over $\Sigma = \{a, b, c\}$, as defined in Example 1, is recognized by $\langle \Sigma, \{q_0, q_1, q_2\}, \{q_0\}, \Delta \rangle$ with

$$\Delta = \{b(x_1) \to q_2(x_1),\ a{\cdot}q_1(x_2) \to q_0(x_2),\ q_2(x_1){\cdot}c \to q_1(x_1),\ q_0(b(x)) \to q_2(x)\}.$$

As an example of derivation, it holds that: $a \cdot a \cdot b(b) \cdot c \cdot c \to a \cdot a \cdot q_2(b) \cdot c \cdot c \to a \cdot a \cdot q_1(b) \cdot c \to a \cdot q_0(b) \cdot c \to a \cdot q_2 \cdot c \to a \cdot q_1 \to q_0$.

2.1. Properties

In this Subsection we present closure and decision properties of $\mathsf{CF}^1\mathsf{HA}$.

Property 1. *The class of CF^1HA languages is closed under union and not closed under intersection or complementation.*

Proof. Closure under union works with a direct construction by disjoint union of automata. Non-closure under intersection is a consequence of the same result for context-free word languages, which are defined as $\mathsf{CF}^1\mathsf{HA}$ without vertical transitions. □

The emptiness problem is the problem to decide whether the language of a given $\mathsf{CF}^1\mathsf{HA}$ is empty or not.

Property 2. *The emptiness problem is decidable in PTIME for $\mathsf{CF}^1\mathsf{HA}$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$. We use a marking algorithm with two marks: h and v. Let us iterate the following operations until no marking is possible (note that the marking is not exclusive: some states may have 2 marks h and v).

For all transitions $\varepsilon \to q$, mark q with h.

For all transitions $a(x) \to q(x)$, mark q with h and v.

For all transitions $q_1(x) \to q(x)$, mark q with the marks of q_1.

For all transitions $q_1(x) \cdot q_2 \to q(x)$ such that both q_1 and q_2 are marked, if q_1 is marked with v, then mark q with v, otherwise mark q with h.

For all transitions $q_1 \cdot q_2(x) \to q(x)$ such that both q_1 and q_2 are marked, if q_2 is marked with v, then mark q with v, otherwise mark q with h.

For all transitions $q_1\big(q_2(x)\big) \to q(x)$ such that q_1 is marked v, if q_2 is marked with v, then mark q with v, otherwise, if q_2 is marked with h, then mark q with h.

The number of iterations is at most $2|Q|$ and the cost of each iteration is linear in the size of $\mathcal{A}$. It holds that (i) $q \in Q$ is marked with h iff there exists $h \in \mathcal{H}(\Sigma)$ such that $h \xrightarrow[\Delta]{*} q$, and (ii) $q \in Q$ is marked with v iff there exists $C[\,] \in \mathcal{C}(\Sigma)$, non-trivial and such that for all $h \in \mathcal{H}(\Sigma)$, $C[h] \xrightarrow[\Delta]{*} q(h)$. Not that the latter implies in particular that $C[\varepsilon] \in L(\mathcal{A}, q)$. These two properties can be proved simultaneously by induction over the hedge structure. It follows from (i) that $L(\mathcal{A}) = \emptyset$ iff no state of Q^{f} is marked with h or with v or with both. □

The membership problem is the problem to decide whether a given hedge is accepted by a given $\mathsf{CF}^1\mathsf{HA}$.

Property 3. *The membership problem is PTIME-complete for $\mathsf{CF}^1\mathsf{HA}$.*

PTIME-hardness follows from the PTIME-hardness of membership for context-free grammars.

For proving the upper-bound we present a dynamic programming algorithm à la Cocke-Younger-Kasami (see *e.g.* [25]), which, given an hedge $h = a(t_1 \cdots t_n)$ and a $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$ runs in deterministic polynomial time. The algorithm associates set of states with some convex regions in h, which are defined by patterns of the two following kinds. For convenience, the ith child of a node p in h is denoted as $p \cdot i$, and we use a special symbol $\top$ for representing a virtual node, parent of all the roots of h, and, for $1 \leq i \leq n$, we denote $\top \cdot i$ the ith root of h, *i.e.* the root node of t_i.

A *h-pattern* is a tuple of the form $\pi = \langle p, i, j\rangle$ where $p \in \mathit{dom}(h) \cup \{\top\}$, and $i \leq j$ are natural numbers such that either $p \cdot i, p \cdot j \in \mathit{dom}(h)$ or p is a leaf of h and $i = j = 1$. This pattern represents an hedge denoted $h|_\pi$, which is ε in the latter case, and $h|_{p\cdot i} \cdots h|_{p\cdot j}$ in the former case.

A *v-pattern* is a tuple of the form $\pi = \langle p, i, p', j\rangle$ where $p \in \mathit{dom}(h) \cup \{\top\}$, $i \leq j$ are natural numbers such that $p \cdot i, p \cdot j \in \mathit{dom}(h)$, and $p' \in \mathit{dom}(h)$ is a descendant of $p \cdot i'$ for some $i \leq i' \leq j$. This pattern represents the context $h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[x] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot j}$, denoted as $h|_\pi$, where x is the only child of the node p'.
The algorithm associates a set of states of $\mathcal{A}$ with each pattern of type h or v, by iteration of the following rules.

1. if $p \cdot i$ is the node of a leaf of h, and $\varepsilon \to q \in \Delta$,
   then we add q to (the set of states associated with) the h-pattern $\langle p, 1, 1\rangle$;
2. if $p \cdot i$ is labeled by a in h, and $a(x) \to q(x) \in \Delta$,
   then we add q to the v-pattern $\langle p, i, p \cdot i, i\rangle$;
3. if q_1 is in the h-pattern $\pi = \langle p, i, j\rangle$, or the v-pattern $\pi = \langle p, i, p', j\rangle$,
   and $q_1(x) \to q(x) \in \Delta$ then we add q to π;
4. if q_1 is in the v-pattern $\langle p, i, p', j\rangle$ and q_2 is in the h-pattern $\langle p, j + 1, k\rangle$,
   and $q_1(x) \cdot q_2 \to q(x) \in \Delta$, then we add q to the v-pattern $\langle p, i, p', k\rangle$;
5. if q_1 is in the h-pattern $\langle p, i, j\rangle$ and q_2 is in the v-pattern $\langle p, j + 1, p', k\rangle$,
   and $q_1 \cdot q_2(x) \to q(x) \in \Delta$, then we add q to the v-pattern $\langle p, i, p', k\rangle$;
6. if q_1 is in the h-pattern $\langle p, i, j\rangle$ and q_2 is in the h-pattern $\langle p, j + 1, k\rangle$,
   and $q_1(x) \cdot q_2 \to q(x) \in \Delta$, or $q_1 \cdot q_2(x) \to q(x) \in \Delta$,
   then we add q to the h-pattern $\langle p, i, k\rangle$;
7. if q_1 is in the v-pattern $\langle p, i, p', j\rangle$ and q_2 is in the v-pattern $\langle p', 1, p'', k\rangle$,
   where k is the number of children of p', and $q_1(q_2(x)) \to q(x) \in \Delta$,
   then we add q to the v-pattern $\langle p, i, p'', j\rangle$;
8. if q_1 is in the v-pattern $\langle p, i, p', j\rangle$ and q_2 is in the h-pattern $\langle p', 1, k\rangle$,
   where k is the number of children of p', and $q_1(q_2(x)) \to q(x) \in \Delta$,
   then we add q to the h-pattern $\langle p, i, j\rangle$.

The number P of patterns is at most cubic in the size of h, and the algorithm will reach a fix point in at most $P|Q|$ iterations. Every iteration needs a time polynomial in the size of $\mathcal{A}$, hence the algorithm terminates in polynomial time. Then it answers that the hedge h is accepted by $\mathcal{A}$ iff the set of states associated with the h-pattern $\langle \top, 1, n\rangle$ contains an accepting state of Q^{f} (n is the number of children of h).

2.2. Correctness of the membership algorithm

In order to establish the correctness of the above algorithm, we show the two following claims simultaneously.

(h) for every h-pattern π, q belongs to π iff $h|_\pi \xrightarrow[\mathcal{A}]{*} q$

(v) for every v-pattern π, q belongs to π iff $h|_\pi \xrightarrow[\mathcal{A}]{*} q(x)$

For the *only if* direction, let us assume that q belongs to a pattern π (h- or v-pattern). We do an induction on the number of iterations of the algorithm before adding q to π, and consider the case of the rule that has added q to π.

rule 1. In this case π is a h-pattern $\langle p, 1, 1\rangle$, p is a leaf node, $\varepsilon \to q \in \Delta$, and hence $h|_\pi = \varepsilon \xrightarrow[\mathcal{A}]{*} q$.

rule 2. Then π is a v-pattern $\langle p, i, p \cdot i, i\rangle$, $p \cdot i$ is labeled by a in h, and $a(x) \to q(x) \in \Delta$, and hence $h|_\pi = a(x) \xrightarrow[\mathcal{A}]{*} q(x)$.

rule 3. By hypothesis, $q_1(x) \to q(x) \in \Delta$ and q_1 is already in π, hence, by induction hypothesis, $h|_\pi \xrightarrow[\mathcal{A}]{*} q_1 \xrightarrow[\mathcal{A}]{} q$ if π is a h-pattern and $h|_\pi \xrightarrow[\mathcal{A}]{*} q_1(x) \xrightarrow[\mathcal{A}]{} q(x)$ if π is a v-pattern.

rule 4. In this case, π is a v-pattern $\langle p, i, p', k\rangle$ and by hypothesis, $q_1(x) \cdot q_2 \to q(x) \in \Delta$, q_1 is already in the v-pattern $\pi_1 = \langle p, i, p', j\rangle$ and q_2 is already in the h-pattern $\pi_2 = \langle p, j+1, k\rangle$. Hence, by induction hypothesis, $h|_\pi = h|_{\pi_1} \cdot h|_{\pi_2} \xrightarrow[\mathcal{A}]{*} q_1(x) \cdot q_2 \xrightarrow[\mathcal{A}]{} q(x)$.

rule 5. This case is similar to the previous one.

rule 6. In this case π is a h-pattern $\langle p, i, k\rangle$, and by hypothesis, $q_1(x) \cdot q_2 \to q(x) \in \Delta$, or $q_1 \cdot q_2(x) \to q(x) \in \Delta$, q_1 is already in the h-pattern $\pi_1 = \langle p, i, j\rangle$ and q_2 is already in the h-pattern $\pi_2 = \langle p, j+1, k\rangle$. Hence by induction hypothesis, $h|_\pi = h|_{\pi_1} \cdot h|_{\pi_2} \xrightarrow[\mathcal{A}]{*} q_1 \cdot q_2 \xrightarrow[\mathcal{A}]{} q(x)$.

rule 7. In this case, π is a v-pattern $\langle p, i, p'', j\rangle$ and by hypothesis, $q_1(q_2(x)) \to q(x) \in \Delta$, q_1 is already in the v-pattern $\pi_1 = \langle p, i, p', j\rangle$ and q_2 is already in the v-pattern $\pi_2 = \langle p', 1, p'', k\rangle$. Using the induction hypothesis, $h|_\pi = h|_{\pi_1}[h|_{\pi_2}] \xrightarrow[\mathcal{A}]{*} q_1(q_2(x)) \xrightarrow[\mathcal{A}]{} q(x)$.

rule 8. Then π is a h-pattern $\langle p, i, j\rangle$, and $q_1(q_2(x)) \to q(x) \in \Delta$, q_1 is already in the v-pattern $\pi_1 = \langle p, i, p', j\rangle$, and q_2 is already in the v-pattern $\pi_2 = \langle p', 1, k\rangle$, where k is the number of children of p' in h. Using the induction hypothesis, $h|_\pi = h|_{\pi_1}[h|_{\pi_2}] \xrightarrow[\mathcal{A}]{*} q_1(q_2) \xrightarrow[\mathcal{A}]{} q$.

Note that the base case of the induction enters in one of the cases 1 or 2 above.

For the *if* direction, let us assume that $h|_\pi \xrightarrow[\mathcal{A}]{*} q$ or $h|_\pi \xrightarrow[\mathcal{A}]{*} q(x)$, according to the type of π, and let us reason by induction on the length of the reduction. If the reduction has length one, then we are in one of the following cases

π is an h-pattern and $h|_\pi = \varepsilon \xrightarrow[\mathcal{A}]{} q$. It means that $\pi = \langle p, 1, 1\rangle$ where p is a leaf node of h. The transition of $\mathcal{A}$ involved in the reduction is necessarily $\varepsilon \to q$, and the rule 1 has been applied by the algorithm to add q to π.

π is an v-pattern and $h|_\pi = a(x) \xrightarrow[\mathcal{A}]{} q(x)$. It means that $\pi = \langle p, i, p \cdot i, i\rangle$ where $p \cdot i$ is a node of h labeled by a. The transition of $\mathcal{A}$ involved in the reduction must be $a(x) \to q(x)$, and the rule 2 has been applied by the algorithm to add q to π.

Assume now that the reduction has length more than one. We make a case analysis on the transition rule of $\mathcal{A}$ involved in the last step of the reduction.

if the last transition is $\varepsilon \to q$ then, according to the form of the $\mathsf{CF}^1\mathsf{HA}$ transitions, the reduction has length one, and this case has been treated above.

assume that the last transition is $q_1(x) \to q(x)$. If π is a h-pattern, then the reduction has the form $h|_\pi \xrightarrow[\mathcal{A}]{*} q_1 \xrightarrow[\mathcal{A}]{} q$. By induction hypothesis on the first part of the reduction, q_1 is in π, and the rule 3 has been applied by the algorithm and hence q has been added to π. The case where π is a v-pattern, and the reduction has the form $h|_\pi \xrightarrow[\mathcal{A}]{*} q_1(x) \xrightarrow[\mathcal{A}]{} q(x)$ is similar.

assume that the last transition is $q_1(x) \cdot q_2 \to q(x)$. If π is a h-pattern $\langle p, i, k\rangle$, then the reduction has the form

$$h|_\pi = h|_{p\cdot i} \cdots h|_{p\cdot k} \xrightarrow[\mathcal{A}]{*} q_1 \cdot q_2 \xrightarrow[\mathcal{A}]{} q.$$

It means that there exists j such that $i \leq j \leq k$ and $h|_{p\cdot i} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1$ and $h|_{p\cdot j+1} \cdots h|_{p\cdot k} \xrightarrow[\mathcal{A}]{*} q_2$. Hence, by induction hypothesis, q_1 is in the h-pattern $\langle p, i, j\rangle$ and q_2 is in the h-pattern $\langle p, j+1, k\rangle$. It follows that the rule 6 has been applied and that q is in π.
If π is a v-pattern $\langle p, i, p', k\rangle$, then the reduction has the form

$$h|_\pi = h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[x] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot k} \xrightarrow[\mathcal{A}]{*} q_1(x) \cdot q_2 \xrightarrow[\mathcal{A}]{} q(x).$$

It means that there exists j such that $i' \leq j \leq k$ and $h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[x] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1(x)$ and $h|_{p\cdot j+1} \cdots h|_{p\cdot k} \xrightarrow[\mathcal{A}]{*} q_2$. Hence, by induction hypothesis, q_1 is in the v-pattern $\langle p, i, p', j\rangle$, and q_2 is in the h-pattern $\langle p, j+1, k\rangle$. It follows that the rule 4 has been applied and that q is in π.

the case of a last transition of the form $q_1 \cdot q_2(x) \to q(x)$ is similar to the previous one.

assume that the last transition is $q_1(q_2(x)) \to q(x)$. If π is a h-pattern $\langle p, i, j\rangle$, then the reduction has the form

$$h|_\pi = h|_{p\cdot i} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1(q_2) \xrightarrow[\mathcal{A}]{} q.$$

It means that there exists i' such that $i \leq i' \leq j$ such that $h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[x] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1(x)$ and $h|_{p'\cdot 1} \cdots h|_{p'\cdot k} \xrightarrow[\mathcal{A}]{*} q_2$, where $p' \in \mathit{dom}(h)$ is the parent node of x (it is a descendant of $p \cdot i'$), and k is the number of children of p' in h. By induction hypothesis, q_1 is in the v-pattern $\langle p, i, p', j\rangle$ and q_2 is in the h-pattern $\langle p', 1, k\rangle$. It follows that the rule 8 has been applied and that q is in π.
If π is a v-pattern $\langle p, i, p'', j\rangle$, then the reduction has the form

$$h|_\pi = h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[x] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1(q_2(x)) \xrightarrow[\mathcal{A}]{} q(x)$$

such that $p \cdot i'$ is an ancestor of p'' and p'' is the parent node of x. It means that there exists a node p' which is a descendant of $p \cdot i'$ such that $h|_{p\cdot i} \cdots h|_{p\cdot i'-1} \cdot (h|_{p\cdot i'})[y] \cdot h|_{p\cdot i'+1} \cdots h|_{p\cdot j} \xrightarrow[\mathcal{A}]{*} q_1(y)$, such that p' is the parent node of y, and $h|_{p'\cdot 1} \cdots h|_{p'\cdot i''-1} \cdot (h|_{p'\cdot i''})[x] \cdot h|_{p'\cdot i'+1} \cdots h|_{p'\cdot k} \xrightarrow[\mathcal{A}]{*} q_2(x)$ such that p'' is the parent node of x (it is a descendant of $p' \cdot i''$), and k is the number of children of p' in h. By induction hypothesis, q_1 is in the v-pattern $\langle p, i, p', j\rangle$ and q_2 is in the h-pattern $\langle p', 1, p'', k\rangle$. It follows that the rule 7 has been applied and that q is in π.

2.3. Related Models

The $\mathsf{CF}^1\mathsf{HA}$ capture the expressiveness of two models of automata on unranked trees: the hedge automata [2] and the lesser known extension of [26] that we call CFHA.

A *hedge automaton* (HA) resp. *context-free hedge automaton* (CFHA) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$ where Σ, Q and Q^{f} are as above, and the transitions of Δ have the form $a(L) \to q$ where $a \in \Sigma$, $q \in Q$ and $L \subseteq Q^*$ is a regular word language (resp. a context-free word language). The language of hedges accepted is defined as for $\mathsf{CF}^1\mathsf{HA}$, using the (possibly infinite) set of rewrite rules $\{a(q_1 \cdots q_n) \to q \mid a(L) \to q \in \Delta, q_1 \cdots q_n \in L\}$.

The CFHA languages form a strict subclass of $\mathsf{CF}^1\mathsf{HA}$ languages. Indeed every CFHA $\langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$ can be presented as a $\mathsf{CF}^1\mathsf{HA}$ with variable-free transitions of the form

$$\varepsilon \to q \quad p_1 \to q \quad p_1 \cdot p_2 \to q \quad a(q_1) \to q$$

where $a \in \Sigma$, $p_1, p_2 \in Q \cup \Sigma$, $q_1, q \in Q$.
It can be shown that the set of T-patterns of Example 2 is not a CFHA language, using a pumping argument on the paths labeled by b.

The HA languages, also called *regular* unranked tree languages, also form a strict subclass of $\mathsf{CF}^1\mathsf{HA}$ languages. Every HA can indeed be presented as a $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ with variable-free transitions constrained with a type discipline: $Q = Q_{\mathsf{h}} \uplus Q_{\mathsf{v}}$ and every transition of Δ has one of the forms

$$a \to q_{\mathsf{h}} \quad a \to q_{\mathsf{v}} \quad q_{\mathsf{h}} \cdot q_{\mathsf{v}} \to q'_{\mathsf{h}} \quad a(q_{\mathsf{h}}) \to q_{\mathsf{h}} \quad a(q_{\mathsf{h}}) \to q_{\mathsf{v}}$$

where $a \in \Sigma$, $q_{\mathsf{h}}, q'_{\mathsf{h}} \in Q_{\mathsf{h}}$, $q_{\mathsf{v}} \in Q_{\mathsf{v}}$. The distinction between horizontal and vertical states permits to ensure that horizontal transitions are applied left to right in siblings. This latter model, can also compared to the *forest automata* of [27], where the horizontal transitions is presented as a finite monoid of states $(Q, +, 0)$, and the vertical transition is defined by a function $\delta : \Sigma \times Q \to Q$ ($\delta(a, 0) = q$ being the analogous of the transition $a \to q$). Forest Automata and HA have the same expressive power. As outlined in introduction, using rewrite rules for the definition of $\mathsf{CF}^1\mathsf{HA}$ transitions permitted to simplify dramatically the construction of rewrite closure in Sections 3 and 4, compared to the former models of HA and CFHA which are more ad-hoc. In the paper, the HA and CFHA that we shall consider will be presented as $\mathsf{CF}^1\mathsf{HA}$.

The class of HA languages is closed under Boolean operations and the class of CFHA languages is closed under union but not closed under intersection and complementation, see [2, 26, 1]. The intersection of a CFHA language and a HA language is a CFHA language. All these results are effective, with PTIME (resp. EXPTIME) constructions of automata of polynomial (resp. exponential) sizes for the closures under union and intersection (resp. complement).

In [23] we proposed a model called $\mathsf{CF}^2\mathsf{HA}$, more general than $\mathsf{CF}^1\mathsf{HA}$, with some 2-variable horizontal transitions of the form $q_1(x_1) \cdot q_2(x_2) \to q(x_1 \cdot x_2)$. The following examples shows that $\mathsf{CF}^2\mathsf{HA}$ are strictly more general than $\mathsf{CF}^1\mathsf{HA}$.

Example 3. The hedge language $\{g(a^n \cdot a^n) \mid n \geq 1\}$ is recognized by the $\mathsf{CF}^2\mathsf{HA}$ with the following transition rules: $a(x_1) \cdot a(x_2) \to q(x_1 \cdot x_2)$, $q(q(x)) \to q(x)$, $g(q(x)) \to q_{\mathsf{f}}(x)$ (q_{f} is the unique final state).

One can observe that the language of Example 3 is not recognizable by a $\mathsf{CF}^1\mathsf{HA}$.

Some models of automata strictly extending HA have been introduced to compute on unranked trees while simultaneously testing for equalities and disequalities between subtrees [28, 29]. These models can recognize the language of Example 3 (which is not $\mathsf{CF}^1\mathsf{HA}$). On the other hand, they cannot recognize the language of T-patterns of Example 2. Hence their expressive power cannot be compared to $\mathsf{CF}^1\mathsf{HA}$.

3. Inverse Monadic Hedge Rewriting Systems

A rewrite rule $\ell \to r$ over Σ is called *monadic* (following [5, 30]) if $r = a(x)$ with $a \in \Sigma$, $x \in \mathcal{X}$, *inverse-monadic* if $r \to \ell$ is monadic and $r \notin \mathcal{X} \cup \{\varepsilon\}$, and *1-childvar* if it contains at most one variable x and every parent node of this variable x in ℓ and r has no other child than x. Intuitively, every finite, linear, inverse-monadic, 1-childvar HRS can be transformed into a HRS equivalent *w.r.t.* reachability whose rules are inverse of transitions of $\mathsf{CF}^1\mathsf{HA}$. It follows that such HRS preserve $\mathsf{CF}^1\mathsf{HA}$ languages.

Example 4. The HRS of Example 1 is linear, inverse-monadic, and 1-childvar. The closure of the language $\{p_0\}$ is the $\mathsf{CF}^1\mathsf{HA}$ language of T-patterns.

3.1. Rewrite Closure

Theorem 1. *Let L be the language of $\mathcal{A}_L \in \mathsf{CF}^1\mathsf{HA}$ and $\mathcal{R}$ be a finite, linear, inverse-monadic, 1-childvar* HRS. *There exists an effectively computable $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A}'$ recognizing $post^*_{\mathcal{R}}(L)$, of size polynomial in the size of $\mathcal{R}$ and $\mathcal{A}_L$.*

Proof. Let $\mathcal{A}_L = \langle \Sigma, Q_L, Q^{\mathsf{f}}_L, \Delta_L \rangle$, we construct a $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$. The state set Q contains all the states of Q_L, one state $\underline{h}$ for every sub-hedge of a *rhs* of rule of $\mathcal{R}$, one state $\underline{a}$ for each $a \in \Sigma$ and one new state $q \notin Q_L$. For each $p \in Q_L \cup \Sigma$, we denote $\underline{p} = \underline{a}$ if $p = a \in \Sigma$ and $\underline{p} = p$ otherwise. Let $Q^{\mathsf{f}} = Q^{\mathsf{f}}_L$ and let Δ_0 contain the rules of Δ_L and the following transition rules, where $a \in \Sigma$, $t \in \mathcal{T}(\Sigma, \{x\})$ and $h \in \mathcal{H}(\Sigma, \{x\}) \setminus \{\varepsilon\}$.

$$\begin{array}{rcll}
\underline{t}(x) \cdot \underline{h} & \to & \underline{t \cdot h}(x) & \text{if } x \in var(t), \underline{t \cdot h} \in Q \\
\underline{t}(x) \cdot \underline{h} & \to & q(x) & \text{if } x \in var(t), \underline{t \cdot h} \notin Q \\
\underline{t} \cdot \underline{h}(x) & \to & \underline{t \cdot h}(x) & \text{if } x \notin var(t), \underline{t \cdot h} \in Q \\
\underline{t} \cdot \underline{h}(x) & \to & q(x) & \text{if } x \notin var(t), \underline{t \cdot h} \notin Q \\
a(x) & \to & \underline{a}(x) & \\
a(\underline{h}(x)) & \to & \underline{a(h)}(x) & \text{if } \underline{a(h)} \in Q \\
a(\underline{h}(x)) & \to & q(x) & \text{if } \underline{a(h)} \notin Q \\
a(q(x)) & \to & q(x) &
\end{array}$$

Finally, let

$$\Delta = \Delta_0 \cup \{\underline{h}(x) \to \underline{a}(x) \mid a(x) \to h \in \mathcal{R}\}.$$

Intuitively, $\mathcal{A}$ will accept:

- in a state $p \in Q_L$: the rewrite closure of the language $L(\mathcal{A}_L, p)$ under $\mathcal{R}$,
- in a state $\underline{h}$ (resp. $\underline{a}$): the rewrite closure under $\mathcal{R}$ of the set of ground instances of the hedge h (resp. $a(x)$),

- in the state q: the rewrite closure under $\mathcal{R}$ of the set of ground hedges not matched by a subhedge of *rhs* of rule of $\mathcal{R}$.

Let $\ell \in \mathcal{H}(\Sigma)$ be such that

$$\ell \xrightarrow[\Delta]{*} s(u) \tag{$\star$}$$

with $s \in Q$ and $u \in \mathcal{H}(Q \cup \Sigma)$. We show by induction on the number N of applications of rules of $\Delta \setminus \Delta_0$ in $(\star)$ that there exists $\ell' \in \mathcal{H}(\Sigma)$ such that $\ell' \xrightarrow[\mathcal{R}]{*} \ell$ and moreover, if $s = \underline{h}$, then h matches ℓ', if $s = q$ then ℓ' is not matched by a subhedge of a *rhs* of a rule of $\mathcal{R}$, and if $s \in Q_L$, then $\ell' \in L(\mathcal{A}_L, s)$.

If $N = 0$, then the property holds with $\ell' = \ell$. This can be shown by induction on the length of $(\star)$, analyzing the different cases for the last transition used in $(\star)$. We consider the case of a transition $\underline{t}(x) \cdot \underline{h} \to \underline{t \cdot h}(x)$ (the other cases are similar). In that case, $(\star)$ has the form

$$\ell = \ell_1 \cdot \ell_2 \xrightarrow[\Delta_0]{*} \underline{t}(u) \cdot \underline{h} \xrightarrow[\Delta_0]{} \underline{t \cdot h}(u) = s(u),$$

with $\ell_1 \xrightarrow[\Delta_0]{*} \underline{t}(u)$ and $\ell_2 \xrightarrow[\Delta_0]{*} \underline{h}$. Since these two derivations are shorter than $(\star)$, by induction hypothesis, it holds that there exists $\ell'_1 \xrightarrow[\mathcal{R}]{*} \ell_1$ such that t matches ℓ_1, and there exists $\ell'_2 \xrightarrow[\mathcal{R}]{*} \ell_2$ such that h matches ℓ_2 (*i.e.* $h = \ell_2$). With $\ell' = \ell'_1 \cdot \ell'_2$, we can conclude since $\ell' \xrightarrow[\mathcal{R}]{*} \ell$ and $t \cdot h$ matches ℓ'.

If $N > 0$, we can assume that $(\star)$ has the following form.

$$\ell = C[k] \xrightarrow[\Delta_0]{*} C[\underline{h}(v)] \xrightarrow[\Delta \setminus \Delta_0]{} C[\underline{a}(v)] \xrightarrow[\Delta]{*} s(u)$$

It follows that h matches k, *i.e.* there exists w such that $k = h[w]$, and $w \xrightarrow[\Delta_0]{*} v$. Hence $\ell' = C[a(w)] \xrightarrow[\mathcal{R}]{} \ell$, and $\ell' \xrightarrow[\Delta_0]{*} C[a(v)] \xrightarrow[\Delta_0]{} C[\underline{a}(v)] \xrightarrow[\Delta]{} s(u)$. We can then apply the induction hypothesis to ℓ' and immediately conclude for ℓ. □

The following Example 5 illustrates the importance of the 1-childvar and condition in Theorem 1.

Example 5. With the following rewrite rule $a(x) \to c \cdot a(e \cdot x \cdot g) \cdot d$ we generate from $\{a\}$ the language $\{c^n a(e^n g^n) \cdot d^n \mid n \geq 1\}$, seemingly not $\mathsf{CF}^1\mathsf{HA}$.

3.2. Backward Rewrite Closure and Typechecking

The following result directly follows from a proof in [16] that the closure of a HA language under rewriting with a monadic HRS is a HA language.

Theorem 2. *Let L be the language of $\mathcal{A}_L \in \mathsf{HA}$ and $\mathcal{R}$ be a finite, inverse-monadic* HRS. *There exists an effectively computable HA $\mathcal{A}'$ recognizing $pre^*_{\mathcal{R}}(L)$.*

The *Reachability* problem is the problem to decide, given two hedges $h, h' \in \mathcal{H}(\Sigma)$ and a rewrite system $\mathcal{R}/\mathcal{A}$ whether $h \xrightarrow[\mathcal{R}/\mathcal{A}]{*} h'$.
The *Typechecking* problem (see *e.g.* [10]) is the problem to decide, given two sets of trees L_{in} and L_{out} called input and output types (generally presented as HA) and a rewrite system $\mathcal{R}/\mathcal{A}$ whether $post^*_{\mathcal{R}/\mathcal{A}}(L_{\mathsf{in}}) \subseteq L_{\mathsf{out}}$. Note that reachability is a special case of typechecking, when both L_{in} and L_{out} are singleton sets. Hence typechecking is undecidable whenever reachability is.

Corollary 1. *Typechecking is decidable for finite inverse-monadic* HRS *and* HA *languages as input and output types.*

Proof. Given languages L_{in} and L_{out} recognized respectively by HAs $\mathcal{A}_{\mathsf{in}}$ and $\mathcal{A}_{\mathsf{out}}$, one can built a HA recognizing the complement $\overline{L_{\mathsf{out}}}$ of L_{out}, by closure of HA languages under complementation, a HA recognizing $\mathit{pre}^*_{\mathcal{R}}(\overline{L_{\mathsf{out}}})$, by Theorem 2, and finally a HA recognizing $L_{\mathsf{in}} \cap \mathit{pre}^*_{\mathcal{R}/\mathcal{A}}(\overline{L_{\mathsf{out}}})$, by closure of HA languages under intersection. This reduces the problem of typechecking for inverse-monadic HRS to the emptiness for HA, which is a decidable problem. □

We conjecture that the intersection of a $\mathsf{CF}^1\mathsf{HA}$ language and a HA language is a $\mathsf{CF}^1\mathsf{HA}$ language (and the result is effective). Using Theorem 1, that would permit to decide typechecking for input types in $\mathsf{CF}^1\mathsf{HA}$ and output types in HA, by reduction to emptiness of $\mathsf{CF}^1\mathsf{HA}$ (Property 2).

4. Update Hedge Rewriting Systems

In this section, we turn to our motivation of studying XQuery Update Facility primitives modeled as parameterized rewriting rules.

Let $\mathcal{A} = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta \rangle$ be a HA. A hedge rewriting system over Σ parametrized by $\mathcal{A}$ (PHRS) is given by a finite set, denoted $\mathcal{R}/\mathcal{A}$, of rewrite rules $\ell \to r$ where $\ell \in \mathcal{H}(\Sigma, \mathcal{X})$ and $r \in \mathcal{H}(\Sigma \uplus Q, \mathcal{X})$ and symbols of Q can only label leaves of r ($\uplus$ stands for disjoint union, hence we implicitly assume that Σ and Q are disjoint sets). In this notation, $\mathcal{A}$ may be omitted when it is clear from context or not necessary. The rewrite relation $\xrightarrow[\mathcal{R}/\mathcal{A}]{}$ associated with a PHRS $\mathcal{R}/\mathcal{A}$ is defined as the rewrite relation $\xrightarrow[\mathcal{R}[\mathcal{A}]]{}$ where the HRS $\mathcal{R}[\mathcal{A}]$ is the (possibly infinite) set of all rewrite rules obtained from rules $\ell \to r$ in $\mathcal{R}/\mathcal{A}$ by replacing in r every state $p \in Q$ by a ground hedge of $L(\mathcal{A}, p)$. Note that when there are multiple occurrences of a state p in a rule, each occurrence of p is independently replaced with a hedge in $L(\mathcal{A}, p)$, which can generally be different from one another. Given a set $L \subseteq \mathcal{H}(\Sigma, \mathcal{X})$, we define $\mathit{post}^*_{\mathcal{R}/\mathcal{A}}(L)$ to be $\mathit{post}^*_{\mathcal{R}[\mathcal{A}]}(L)$
We call *updates* parametrized rewrite rules of the following form

$a(x) \to b(x)$		node renaming	(ren)
$a(x) \to a(u_1 \cdot x \cdot u_2)$	$u_1, u_2 \in Q^*$	addition of child nodes	(ac)
$a(x) \to v_1 \cdot a(x) \cdot v_2$	$v_1, v_2 \in Q^*$	addition of sibling nodes	(as)
$a(x) \to b\big(a(x)\big)$		addition of parent node	(ap)
$a(x) \to u$	$u \in Q^*$	replacement/recursive deletion	(rpl)
$a(x) \to x$		single node deletion	(del)

Note that the particular case of (rpl) of rpl with $u = \varepsilon$ corresponds to the deletion of the whole subtree $a(x)$. In the rest of the paper, a PHRS containing only updates will be called update PHRS (uPHRS).

Example 6. The data of patients in a hospital is stored in an XML document whose schema can be characterized by an HA $\mathcal{A}$ with the following transition rules, where the vertical ones are given before the horizontal ones:

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{hospital} & \to & q_{\mathsf{hospital}} & \mathsf{hospital}(q^{\mathsf{hospital}}) & \to & q_{\mathsf{hospital}} \\
\mathsf{patient}(q^{\mathsf{epatient}}) & \to & q_{\mathsf{epatient}} & \mathsf{patient}(q^{\mathsf{patient}}) & \to & q_{\mathsf{patient}} \\
\mathsf{name}(q^{\mathsf{name}}) & \to & q_{\mathsf{name}} & \mathsf{treatment}(q^{\mathsf{treatment}}) & \to & q_{\mathsf{treatment}} \\
\mathsf{drug}(q^{\mathsf{drug}}) & \to & q_{\mathsf{drug}} & \mathsf{diagnosis}(q^{\mathsf{diagnosis}}) & \to & q_{\mathsf{diagnosis}} \\
\mathsf{date}(q^{\mathsf{date}}) & \to & q_{\mathsf{date}} & & &
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{a} \to q_{\mathsf{str}}\ \mathsf{b} & \to & q_{\mathsf{str}} \ldots & \mathsf{a} \cdot q_{\mathsf{str}} & \to & q_{\mathsf{str}} \ldots \\
q_{\mathsf{str}} & \to & q^{\mathsf{name}} & q_{\mathsf{str}} & \to & q^{\mathsf{drug}} \\
q_{\mathsf{str}} & \to & q^{\mathsf{diagnosis}} & q_{\mathsf{str}} & \to & q^{\mathsf{date}} \\
q_{\mathsf{drug}} \cdot q_{\mathsf{diagnosis}} \cdot q_{\mathsf{date}} & \to & q^{\mathsf{treatment}} & & & \\
q_{\mathsf{name}} & \to & q^{\mathsf{epatient}} & q_{\mathsf{name}} \cdot q_{\mathsf{treatment}} & \to & q^{\mathsf{patient}} \\
q_{\mathsf{epatient}} & \to & q^{\mathsf{hospital}} & q_{\mathsf{patient}} & \to & q^{\mathsf{hospital}} \\
q_{\mathsf{epatient}} \cdot q^{\mathsf{hospital}} & \to & q^{\mathsf{hospital}} & q_{\mathsf{patient}} \cdot q^{\mathsf{hospital}} & \to & q^{\mathsf{hospital}}
\end{array}$$

In this example, the states with superscripts correspond to hedges, and states with subscripts (except q_{str}) correspond to trees. The state q_{epatient} represents the type of a patient with an empty treatment and q_{patient} represents the other patients. The state q_{hospital} is the only final state *i.e.* it represents the type of the valid documents (it is sometimes also called *entry point* of the schema).

A (rpl) rule such as $\mathsf{patient}(x) \to \varepsilon$ will delete one patient in the base and a (ac) rule $\mathsf{hospital}(x) \to \mathsf{hospital}(x \cdot q_{\mathsf{patient}})$ will insert a new patient at the last node below the root node hospital. We can ensure that the newly added patient has an empty treatment list using $\mathsf{hospital}(x) \to \mathsf{hospital}(x \cdot q_{\mathsf{epatient}})$. A (as) rule $\mathsf{name}(x) \to \mathsf{name}(x) \cdot q_{\mathsf{treatment}}$ can be used to insert later a treatment next to the patient's name. ◇

As illustrated by the above Example 6, the class PHRS can be used to model XML update operations for XML documents. More precisely, rules of type (ren), (ac), (as), (rpl) correspond to primitives defined in the XQUF standard [12] for the node renaming, insertion and replacement in XML documents. The deletion of a whole subtree, which is also a primitive of [12] is also a particular case of (rpl) of the form $a(x) \to \varepsilon$. The two other kinds of rules (ap) and (del) are not in [12]. We believe that they can be useful as well for modeling some inplace modifications of XML documents. For instance, (del) deletes a single node n whose arguments inherit the position. In other words, it replaces a tree with the hedge containing its children. This operation is employed to build user views of XML documents *e.g.* in [17], and can also be useful for updates as well.

Example 7. Assume that some patients of the hospital of Example 6 are grouped in one department like in $\mathsf{hospital}(\ldots \mathsf{surgery}(q^*_{\mathsf{patient}}) \ldots)$, and that we want to suppress the department surgery while keeping its patients. This can be done with the (del) rule $\mathsf{surgery}(x) \to x$. If on the contrary we want to include surgery in a larger department emergency, we apply an (ap) rule $\mathsf{surgery}(x) \to \mathsf{emergency}(\mathsf{surgery}(x))$. ◇

The rules of type (ren), (as), (ap) and (rpl) can be easily simulated by the HRS of Theorem 1, Section 3. In particular, the parameters' semantics can be simulated using ground rewrite rules (with such rules, a symbol can generate a HA language). The rules (ac) are not 1-childvar and the rule (del) is not inverse-monadic.

The main result of this section, is a proof of forward rewrite closure for uPHRS, using automata completion techniques: starting from an automaton recognizing the initial set, we add transition for, roughly, simulating the rewrite rules. Since we do not add new states, the construction terminates with a fixpoint.

In Section 4.2, we consider first (Theorem 3) the case of *loop-free* uPHRS, introduced in Section 4.1, which do not permit looping sequences of renaming. Then we lift to the general case (Corollary 3) using results of Section 4.1. In Section 4.3 we recall a result of backward rewrite closure (Theorem 4) and application of the above results to typechecking for uPHRS (Corollary 4).

4.1. Loop-free uPHRS

In the proof of the next Theorem 3, we will reduce the class of rewrite systems in consideration to the case where there exists no looping sequence of renaming, for simplification purposes. Therefore we introduce the following definition:

Definition 2. An uPHRS $\mathcal{R}/\mathcal{A}$ is *loopfree* if there exists no sequence $a_1, \ldots, a_n$ ($n > 1$) such that for all $1 \leq i < n$, $a_i(x) \to a_{i+1}(x)$ is a renaming rule of $\mathcal{R}$ and $a_1 = a_n$.

Given a uPHRS $\mathcal{R}/\mathcal{A}$, we consider the directed graph G whose set of nodes is Σ and containing an edge $\langle a, b\rangle$ iff $a(x) \to b(x)$ is in $\mathcal{R}$. For every strongly connected component in G we select a representative. We denote by $\hat{a}$ the representative of a in its component and more generally by $\hat{h}$ the hedge obtained from $h \in \mathcal{H}(\Sigma)$ by replacing every function symbol a by its representative $\hat{a}$. We define $\hat{\mathcal{R}}$ to be $\mathcal{R}$ where every rule $\ell \to r$ is replaced by $\hat{\ell} \to \hat{r}$ (if the two members get equal we can remove the rule). We define $\hat{\mathcal{A}}$ analogously.

Lemma 1. *Given an* uPHRS $\mathcal{R}/\mathcal{A}$ *the* uPHRS $\hat{\mathcal{R}}/\hat{A}$ *is loopfree and for all* $h, h' \in \mathcal{H}(\Sigma)$ *we have* $h \xrightarrow[\mathcal{R}/\mathcal{A}]{*} h'$ *iff* $\hat{h} \xrightarrow[\hat{R}/\hat{A}]{*} \hat{h}'$.

Proof. By induction on the length of derivations. □

4.2. Rewrite Closure

The rest of the section is devoted to the construction of $\mathsf{CF}^1\mathsf{HA}$ for the forward closure by updates.

Theorem 3. *Let $\mathcal{A}$ be a HA over Σ, and L be the language of $\mathcal{A}_L \in \mathsf{CFHA}$, and $\mathcal{R}/\mathcal{A}$ be a loop-free* uPHRS. *There exists an effectively computable CFHA recognizing $post^*_{\mathcal{R}/\mathcal{A}}(L)$, of size polynomial in the size of $\mathcal{R}/\mathcal{A}$ and $\mathcal{A}_L$ and exponential in the size of the alphabet Σ.*

The construction of the CFHA works in 2 steps: construction of an initial automaton and completion step. We need first a notion of normalization of CFHA in order to simplify the proofs: a CFHA $\langle \Sigma, Q, Q^{\mathsf{f}}, \Delta\rangle$ is called *normalized* if for all $a \in \Sigma$ and $q \in Q$, there exists one unique state of Q denoted q^a such that $a(q^a) \to q \in \Delta$, and moreover, q^a does neither occur in a left hand side of an horizontal transition of Δ nor in a right hand side of a vertical transition of Δ.

Lemma 2 (Normalization). *For all CFHA $\mathcal{A}$, there exists a normalized CFHA $\mathcal{A}'$ such that $L(\mathcal{A}') = L(\mathcal{A})$, of size linear in the size of $\mathcal{A}$ and which can be constructed in PTIME.*

The proof of Lemma 2 follows the lines of the similar construction for normalized HA, see Remark 8.2.6 (and above) in [1].

Initial automaton. Let $\mathcal{A} = \langle \Sigma, Q_{\mathcal{A}}, Q^{\mathsf{f}}_{\mathcal{A}}, \Delta_{\mathcal{A}} \rangle$ and $\mathcal{A}_L = \langle \Sigma, Q_L, Q^{\mathsf{f}}_L, \Delta_L \rangle$. We assume that the state sets $Q_{\mathcal{A}}$ and Q_L are disjoint. We will construct a $\mathsf{CF}^1\mathsf{HA}$ $\mathcal{A}'$ for the recognition of $post^*_{\mathcal{R}/\mathcal{A}}(L)$.

First, in order to simplify the construction, let us merge $\mathcal{A}$ and $\mathcal{A}_L$ into a CFHA $\mathcal{B} = \langle \Sigma, P, P^{\mathsf{f}}, \Gamma \rangle$ obtained by the normalization of $\langle \Sigma, Q_{\mathcal{A}} \uplus Q_L, Q^{\mathsf{f}}_L, \Delta_{\mathcal{A}} \uplus \Delta_L \rangle$. Below, the states of P will be denoted by the letters p or q. Let P_{in} be the subset of states of P of the form q^a (remember that q^a is a state of P uniquely characterized by $a \in \Sigma$, $q \in P$, since $\mathcal{B}$ is normalized). We assume *wlog* that P_{in} and P^{f} are disjoint and that $\mathcal{B}$ is *clean*, *i.e.* for all $p \in P$, $L(\mathcal{B}, p) \neq \emptyset$.

Preliminary transformation. Next, in a preliminary construction step, we transform the initial automaton $\mathcal{B}$ into a CFHA $\mathcal{A}_0 = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta_0 \rangle$ (presented as a $\mathsf{CF}^1\mathsf{HA}$, see Section 2.3). Let us call *renaming chain* a sequence $a_1, \ldots, a_n$ of symbols of Σ such that $n \geq 1$ for all $1 \leq i < n$, $a_i(x) \to a_{i+1}(x) \in \mathcal{R}$. Since $\mathcal{R}$ is loop-free, the length of every renaming chain is bounded by $|\Sigma|$. The fresh state symbols of Q are defined as extensions of the symbols of $P \setminus P_{\mathsf{in}}$ with renaming chains. We consider two modes for such states: the *hedge* and *tree* modes, characterized by a chain respectively in superscript or subscript.

$$Q = P \cup \{q_a \mid q^a \in P_{\mathsf{in}}\} \cup \left\{ \begin{array}{c|l} q^{a_1 \cdots a_n} & q \in P \setminus P_{\mathsf{in}}, n \geq 2, \\ q_{a_1 \ldots a_n} & a_1, \ldots, a_n \text{ is a renaming chain} \end{array} \right\}$$

Let $Q^{\mathsf{f}} = P^{\mathsf{f}}$ be the subset of final states. Intuitively, in the state $q^{a_1 \cdots a_n}$, the chain of Σ^+ represents a sequence of renamings, with $\mathcal{R}/\mathcal{A}$, of the parent of the current symbol, starting with a_1 and ending with a_n. Note that the states of P_{in} are particular cases of such states, with a chain of length one. A state $q_{a_1 \ldots a_n}$ will be used below to represent the tree $a_n(q^{a_1 \cdots a_n})$.
The initial set of transitions Δ_0 is defined as follows

$$\begin{array}{rcl} \Delta_0 = \Gamma_h & \cup & \{q_{a_1} \to q \mid q_{a_1} \in Q\} \\ & \cup & \{a_n\big(q^{a_1 \cdots a_n}\big) \to q_{a_1 \ldots a_n} \mid q^{a_1 \cdots a_n}, q_{a_1 \ldots a_n} \in Q, n \geq 1\} \end{array}$$

where Γ_h is the subset of horizontal transitions of Γ. Note that $\mathcal{A}_0$ is not normalized. The following lemma is immediate by construction of Γ and $\mathcal{A}_0$.

Lemma 3. *For all $q \in Q_{\mathcal{A}}$ (resp. $q \in Q_L$) $L(\mathcal{A}_0, q) = L(\mathcal{A}, q)$ (resp. $L(\mathcal{A}_L, q)$).*

Proof. Every vertical transition in Γ has the form $a(q^a) \to q$ and can be simulated by the 2 steps $a(q^a) \to q_a \to q$. Moreover, all the states $q^{a_1 \cdots a_n}$ and $q_{a_1 \ldots a_n}$ with $n \geq 2$ are empty for $\mathcal{A}_0$. □

Completion. For the construction of $\mathcal{A}'$, we shall complete Δ_0 into Δ' by adding some transition rules, according to a case analysis of the rules of $\mathcal{R}/\mathcal{A}$ presented in Table 1. In the rules of Table 1, $a_1, \ldots, a_n, b$ are symbols of Σ, and u, v are sequences of $Q^*_{\mathcal{A}}$. Only a bounded number of rules can be added to Δ_0. Let $\mathcal{A}' = \langle \Sigma, Q, Q^{\mathsf{f}}, \Delta' \rangle$ be the completed automata that we obtain.

Example 8. Let $\mathcal{A}$ be the HA of Example 6, let $\mathcal{R}/\mathcal{A} = \{\mathsf{hospital}(x) \to \mathsf{hospital}(x \cdot q_{\mathsf{epatient}})\}$, and let L be recognized by a HA $\mathcal{A}_L$ obtained from $\mathcal{A}$ by deleting the four transitions containing q^{epatient} or q_{epatient}.

The CFHA $\mathcal{A}'$ obtained by completion of $\mathcal{A}_L$ wrt $\mathcal{R}/\mathcal{A}$ recognizes (in its final state q_{hospital}) the set of trees $\mathsf{hospital}(h_{\mathsf{pa}} \cdot h_{\mathsf{epa}})$, where h_{pa} and h_{epa} are finite

	$\mathcal{R}/\mathcal{A}$ contains	$\Delta' = \Delta_0 \cup$
(ren)	$a_n(x) \to b(x)$	$\{q^{a_1 \ldots a_n} \to q^{a_1 \ldots a_n b} \mid q^{a_1 \ldots a_n b} \in Q\}$ $\cup \quad \{q_{a_1 \ldots a_n b} \to q_{a_1 \ldots a_n} \mid q_{a_1 \ldots a_n b} \in Q\}$
(ac)	$a_n(x) \to a_n(u \cdot x \cdot v)$	$\{u \cdot q^{a_1 \ldots a_n} \cdot v \to q^{a_1 \ldots a_n} \mid q^{a_1 \ldots a_n} \in Q\}$
(as)	$a_n(x) \to u \cdot a_n(x) \cdot v$	$\{u \cdot q_{a_1 \ldots a_n} \cdot v \to q_{a_1 \ldots a_n} \mid q_{a_1 \ldots a_n} \in Q\}$
(ap)	$a_n(x) \to b\big(a_n(x)\big)$	$\{b\big(q_{a_1 \ldots a_n}\big) \to q_{a_1 \ldots a_n} \mid q_{a_1 \ldots a_n} \in Q\}$
(rpl)	$a_n(x) \to u$	$\{u \to q_{a_1 \ldots a_n} \mid q_{a_1 \ldots a_n} \in Q\}$
(del)	$a_n(x) \to x$	$\{q^{a_1 \ldots a_n} \to q_{a_1 \ldots a_n} \mid q^{a_1 \ldots a_n} \in Q\}$

Table 1: CFHA Completion

sequences of trees respectively of the form: $\mathsf{patient}(\mathsf{name}(\ldots) \cdot \mathsf{treatment}(\ldots))$ and $\mathsf{patient}(\mathsf{name}(\ldots))$. The completion indeed adds to $\mathcal{A}'$ a new horizontal transition $q^{\mathsf{hospital}} \cdot q_{\mathsf{epatient}} \to q^{\mathsf{hospital}}$, using the line (ac) of Table 1. ◇

Example 9. Let $\mathcal{A}$ be again the HA of Example 6, and let $\mathcal{R}/\mathcal{A} = \{\mathsf{name}(x) \to \mathsf{name}(x) \cdot q_{\mathsf{treatment}}, \mathsf{patient}(x) \to \varepsilon\}$. The first rule is the insertion of a treatment for a patient and the second ones deletes a patient.

Let $L = \{\mathsf{hospital}(\mathsf{patient}(\mathsf{name}(\mathsf{a})))\}$. This language is recognized by the following HA $\mathcal{A}_L = \big(\{q_0, q_1, q_2, q_3\}, \{q_3\}, \{\mathsf{a} \to q_0, \mathsf{name}(q_0) \to q_1, \mathsf{patient}(q_1) \to q_2, \mathsf{hospital}(q_2) \to q_3\}\big)$.

Then, in addition to the transitions of $\mathcal{A}$ and $\mathcal{A}_L$, the completed CFHA $\mathcal{A}'$ contains the transitions $q_{\mathsf{name}} \cdot q_{\mathsf{treatment}} \to q_{\mathsf{name}}$ (according to the line (as) of Table 1) and $\varepsilon \to q_{\mathsf{patient}}$ (according to the line (rpl) of Table 1). It recognizes the language containing the tree $\mathsf{hospital}$ and all trees of the form $\mathsf{hospital}(\mathsf{patient}(\mathsf{name}(\mathsf{a}) \cdot h))$ where h is a sequence of trees of the form $\mathsf{treatment}(\ldots)$. This language is indeed the forward closure of L by $\mathcal{R}/\mathcal{A}$. ◇

Example 10. Let us come back again to Example 6, with a slight variation $\mathcal{A}^v$ of $\mathcal{A}$, obtained by the replacement of the rule $\mathsf{patient}(q^{\mathsf{epatient}}) \to q_{\mathsf{epatient}}$ by $\mathsf{patient}'(q^{\mathsf{epatient}}) \to q_{\mathsf{epatient}}$ where $\mathsf{patient}'$ is a new symbol (for patients without treatment). We consider the following PHRS

$$\mathcal{R}/\mathcal{A}^v = \{\mathsf{patient}'(x) \to \mathsf{patient}(x), \mathsf{patient}(x) \to q_{\mathsf{patient}} \cdot \mathsf{patient}(x)\}$$

and the language $L = \{\mathsf{hospital}(\mathsf{patient}')\}$ recognized by the following HA

$$\mathcal{A}_L = \big(\{q, q_1\}, \{q_1\}, \{\mathsf{patient}' \to q, \mathsf{hospital}(q) \to q_1\}\big).$$

The initial normalization step introduces a transition $\mathsf{patient}'(q^{\mathsf{patient}'}) \to q$ where the state $q^{\mathsf{patient}'}$ recognizes only ε, and the preliminary transformation adds the transitions $q_{\mathsf{patient}'} \to q$, $\mathsf{patient}'(q^{\mathsf{patient}'}) \to q_{\mathsf{patient}'}$, and $\mathsf{patient}(q^{\mathsf{patient}'\mathsf{patient}}) \to q_{\mathsf{patient}'\mathsf{patient}}$ (note that $\mathsf{patient}'\mathsf{patient}$ is a renaming chain).

The completion procedure introduces the following transitions: $q^{\mathsf{patient}'} \to q^{\mathsf{patient}'\mathsf{patient}}$ and $q_{\mathsf{patient}'\mathsf{patient}} \to q_{\mathsf{patient}'}$ by the line (ren) of Table 1, and $q_{\mathsf{patient}} \cdot q_{\mathsf{patient}} \to q_{\mathsf{patient}}$ by (as). Moreover, it also generates, by (as) again: $q_{\mathsf{patient}} \cdot q_{\mathsf{patient}'\mathsf{patient}} \to q_{\mathsf{patient}'\mathsf{patient}}$.

The automaton obtained after the completion recognizes $\mathsf{hospital}(\mathsf{patient}')$ and all the trees of the form $\mathsf{hospital}(t_1 \cdots t_n{\cdot}\mathsf{patient})$ where $n \geq 0$ and $t_1, \ldots, t_n \in L(\mathcal{A}', q_{\mathsf{patient}})$. $\diamond$

4.2.1. Correctness.

The following Lemma 4 shows that the automata computations simulate the rewrite steps, *i.e.* that $L(\mathcal{A}') \subseteq \mathit{post}^*_{\mathcal{R}/\mathcal{A}}(L)$. Let us abbreviate $\mathcal{R}/\mathcal{A}$ by $\mathcal{R}$. We use the notation $h \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h'$, for a renaming chain $a_1, \ldots, a_n$ ($n \geq 1$), if there exists $h_1, \ldots h_n \in \mathcal{H}(\Sigma)$ such that

$$h = a_1(h_1) \xrightarrow[\mathcal{R}]{*} a_1(h_2) \xrightarrow[\mathsf{ren}]{} a_2(h_2) \xrightarrow[\mathcal{R}]{*} \ldots \xrightarrow[\mathcal{R}]{*} a_{n-1}(h_n) \xrightarrow[\mathsf{ren}]{} a_n(h_n) \xrightarrow[\mathcal{R}]{*} h'$$

where the reductions denoted $\xrightarrow[\mathsf{ren}]{}$ are rewrite steps with rules of $\mathcal{R}/\mathcal{A}$ of type (ren), applied at the nodes labeled a_1,..., a_n, and all the other rewrite steps (denoted $\xrightarrow[\mathcal{R}]{*}$) involve no rule of type (ren).

Lemma 4 (Correctness). *For all $h \in \mathcal{H}(\Sigma)$,*

i. if $h \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n}$, with $n \geq 1$, then there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$,

ii. if $h \xrightarrow[\mathcal{A}']{*} q^{a_1 \cdots a_n}$, with $n \geq 1$, then there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $h_1 \xrightarrow[\mathcal{B}]{*} q^{a_1}$, and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} a_n(h)$,

iii. if $h \xrightarrow[\mathcal{A}']{*} q \in P \setminus P_{\mathsf{in}}$, then there exists $h' \in \mathcal{H}(\Sigma)$ such that $h' \xrightarrow[\mathcal{B}]{*} q$ and $h' \xrightarrow[\mathcal{R}]{*} h$.

Proof. Let $s \in Q$ be such that $h \xrightarrow[\mathcal{A}']{*} s$ and let us call ρ this reduction. With a commutation of transitions, we can assume that ρ has the following form,

$$\rho : h = t_1 \cdots t_m \xrightarrow[\mathcal{A}']{*} \underbrace{s_1 \cdots s_m \xrightarrow[\mathcal{A}']{*} s}_{\rho_0}$$

where $t_1, \ldots, t_m \in \mathcal{T}(\Sigma)$, $s_1, \ldots, s_m \in Q$, and for all $1 \leq i \leq m$, $t_i \xrightarrow[\mathcal{A}']{*} s_i$, and the last step of this reduction involves a vertical transition $a(q^{a_1 \cdots a_n}) \to s_i$ or $b(q_{a_1 \ldots a_n}) \to s_i$. The proof is by induction on the length of ρ.

The shortest possible ρ has 2 steps: $h = t_1 = a(\varepsilon) \xrightarrow[\mathcal{A}_0]{} a(q^a) \xrightarrow[\mathcal{A}_0]{} q = s$ and (iii) holds immediately with $h' = h$, by Lemma 3.

For the induction step, we consider the length of ρ_0.
If $|\rho_0| = 0$, we have necessarily $m = 1$, and the reduction ρ has one of the two following forms ($\vec{v} \in Q^*$).

$$\begin{aligned} h &= t_1 = b(h') \xrightarrow[\mathcal{A}']{*} b(\vec{v}) \xrightarrow[\mathcal{A}']{*} b(q_{a_1 \ldots a_n}) \xrightarrow[\mathcal{A}']{} q_{a_1 \ldots a_n} = s_1 = s & (1) \\ h &= t_1 = a_n(h') \xrightarrow[\mathcal{A}']{*} a_n(\vec{v}) \xrightarrow[\mathcal{A}']{*} a_n(q^{a_1 \cdots a_n}) \xrightarrow[\mathcal{A}_0]{} q_{a_1 \ldots a_n} = s_1 = s & (2) \end{aligned}$$

In the case (1), assume that the vertical transition $b(q_{a_1 \ldots a_n}) \to q_{a_1 \ldots a_n}$ has been added to $\mathcal{A}'$ because $\mathcal{R}/\mathcal{A}$ contains a rule $a_n(x) \to b\big(a_n(x)\big)$. By induction hypothesis (i) applied to the sub-reduction $h' \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n}$, there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$, and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h'$. It follows in particular that there exists h_n such that $a_n(h_n) \xrightarrow[\mathcal{R}]{*} h'$, and using the above (ap) rewrite rule, $a_n(h_n) \xrightarrow[\mathcal{R}]{} b\big(a_n(h_n)\big) \xrightarrow[\mathcal{R}]{*} b(h') = h$. Therefore, $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$ and (i) holds for h and s.

In the case (2), by induction hypothesis (ii) applied to the sub-reduction $h' \xrightarrow[\mathcal{A}']{*} q^{a_1 \ldots a_n}$, there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $h_1 \xrightarrow[\mathcal{B}]{*} q^{a_1}$, hence $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$, and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} a_n(h') = h$. Therefore (i) holds for h and s.

Assume now that $|\rho_0| > 0$, and let us analyze the horizontal transition rule used in the last step of ρ_0.

Case Δ_0.1. The last step of ρ_0 involves $q_1 \cdots q_n \to q \in \Gamma_h$ (horizontal transition of $\mathcal{B}$), with $n \geq 0$. In this case, the reduction ρ has the form

$$h = h_1 \cdots h_n \xrightarrow[\mathcal{A}']{*} s_1 \cdots s_m \xrightarrow[\mathcal{A}']{*} q_1 \cdots q_n \xrightarrow[\mathcal{A}_0]{} q = s$$

with $n \leq m$, $h_i \in \mathcal{H}(\Sigma)$ and $h_i \xrightarrow[\mathcal{A}']{*} q_i$ for all $i \leq n$. By induction hypothesis (iii) applied to the latter reductions, for all $i \leq n$, there exists h'_i such that $h'_i \xrightarrow[\mathcal{B}]{*} q_i$ and $h'_i \xrightarrow[\mathcal{R}]{*} h_i$. Hence (iii) holds for h and s with $h' = h'_1 \cdots h'_n$, since $h' \xrightarrow[\mathcal{B}]{*} q_1 \cdots q_n \xrightarrow[\mathcal{B}]{} q$, and $h' \xrightarrow[\mathcal{R}]{*} h$.

Case Δ_0.2. The last step of ρ_0 uses $q_{a_1} \to q \in \Delta_0$. In this case, the reduction ρ has the form

$$h \xrightarrow[\mathcal{A}']{*} q_{a_1} \xrightarrow[\mathcal{A}_0]{} q = s$$

By induction hypothesis (i) applied to $h \xrightarrow[\mathcal{A}']{*} q_{a_1}$, there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1} h$. Hence, (iii) holds with $h' = a_1(h_1)$.

Case (ren).1. The last step of ρ_0 uses $q^{a_1 \cdots a_{n-1}} \to q^{a_1 \cdots a_n}$ and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}$ contains a rule $a_{n-1}(x) \to a_n(x)$. In this case, the reduction ρ has the form

$$h \xrightarrow[\mathcal{A}']{*} q^{a_1 \ldots a_{n-1}} \xrightarrow[\mathcal{A}']{} q^{a_1 \ldots a_n} = s \tag{3}$$

By induction hypothesis (ii) applied to $h \xrightarrow[\mathcal{A}']{*} q^{a_1 \ldots a_{n-1}}$, there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $h_1 \xrightarrow[\mathcal{B}]{*} q^{a_1}$, and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_{n-1}} a_{n-1}(h)$. Since by hypothesis $a_{n-1}(h) \xrightarrow[\mathcal{R}]{} a_n(h)$, we have $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} a_n(h)$ and (ii) holds for h and s.

Case (ren).2. The last step of ρ_0 uses $q_{a_1 \ldots a_n} \to q_{a_1 \ldots a_{n-1}}$ and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}$ contains a rule $a_{n-1}(x) \to a_n(x)$. In this case, the reduction ρ has the form

$$h \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n} \xrightarrow[\mathcal{A}']{} q_{a_1 \ldots a_{n-1}} = s \tag{4}$$

By induction hypothesis (i) applied to $h \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n}$, there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$, and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$. It follows, by definition of $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$ at the begining of Section 4.2.1, that $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_{n-1}} ht$ (because $\xrightarrow[\mathsf{ren}]{}$ is included, as a relation, in $\xrightarrow[\mathcal{R}]{*}$).

Case (ac). The last step of ρ_0 uses $u \cdot q^{a_1 \cdots a_n} \cdot v \to q^{a_1 \cdots a_n}$ and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}$ contains a rule $a_n(x) \to a_n(u \cdot x \cdot v)$, with $u, v \in Q^*_{\mathcal{A}}$. In this case, the reduction ρ has the following form,

$$h = \ell \cdot h' \cdot r \xrightarrow[\mathcal{A}']{*} u \cdot q^{a_1 \ldots a_n} \cdot v \xrightarrow[\mathcal{A}']{} q^{a_1 \ldots a_n} = s \tag{5}$$

where $\ell \xrightarrow[\mathcal{A}']{*} u$, $h' \xrightarrow[\mathcal{A}']{*} q^{a_1 \cdots a_n}$, and $r \xrightarrow[\mathcal{A}']{*} v$. By induction hypothesis (ii) applied to $h' \xrightarrow[\mathcal{A}']{*} q^{a_1 \cdots a_n}$, there exists h_1 such that $h_1 \xrightarrow[\mathcal{B}]{*} q^{a_1}$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} a_n(h')$, and by induction hypothesis (iii) applied to $\ell \xrightarrow[\mathcal{A}']{*} u$ (resp. $r \xrightarrow[\mathcal{A}']{*} v$), and by Lemma 3, there exists $\ell' \in \mathcal{H}(\Sigma)$ (resp. $r' \in \mathcal{H}(\Sigma)$) such that $\ell' \xrightarrow[\mathcal{A}]{*} u$ (resp. $r' \xrightarrow[\mathcal{A}]{*} v$) and $\ell' \xrightarrow[\mathcal{R}]{*} \ell$ (resp. $r' \xrightarrow[\mathcal{R}]{*} r$). It follows that $a_n(h') \xrightarrow[\mathcal{R}]{} a_n(\ell' \cdot h' \cdot r') \xrightarrow[\mathcal{R}]{*} a_n(\ell \cdot h' \cdot r) = a_n(h)$. Hence $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} a_n(h)$ and (ii) holds for h and s.

Case (as). The last step of ρ_0 uses $u \cdot q_{a_1 \ldots a_n} \cdot v \to q_{a_1 \ldots a_n}$ and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}$ contains a rule $a_n(x) \to u \cdot a_n(x) \cdot v$, with $u, v \in Q^*_{\mathcal{A}}$. In this case, the reduction ρ has the following form,

$$h = \ell \cdot h' \cdot r \xrightarrow[\mathcal{A}']{*} u \cdot q_{a_1 \ldots a_n} \cdot v \xrightarrow[\mathcal{A}']{} q_{a_1 \ldots a_n} = s$$

where $\ell \xrightarrow[\mathcal{A}']{*} u$, $h' \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n}$, and $r \xrightarrow[\mathcal{A}']{*} v$. By induction hypothesis (i) applied to $h' \xrightarrow[\mathcal{A}']{*} q_{a_1 \ldots a_n}$, there exists h_1 such that $a_1(h_1) \xrightarrow[\mathcal{B}]{*} q$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h'$. To be more precise, the latter reduction has the form

$$a_1(h_1) \xrightarrow[\mathcal{R}]{*} a_1(h_2) \xrightarrow[\mathsf{ren}]{} a_2(h_2) \xrightarrow[\mathcal{R}]{*} \ldots \xrightarrow[\mathcal{R}]{*} a_{n-1}(h_n) \xrightarrow[\mathsf{ren}]{} a_n(h_n) \xrightarrow[\mathcal{R}]{*} h'$$

for some $h_2, \ldots, h_n \in \mathcal{H}(\Sigma)$.

Moreover, by induction hypothesis (iii) applied to $\ell \xrightarrow[\mathcal{A}']{*} u$ (resp. $r \xrightarrow[\mathcal{A}']{*} v$), and by Lemma 3, there exists $\ell' \in \mathcal{H}(\Sigma)$ (resp. $r' \in \mathcal{H}(\Sigma)$) such that $\ell' \xrightarrow[\mathcal{A}]{*} u$ (resp. $r' \xrightarrow[\mathcal{A}]{*} v$) and $\ell' \xrightarrow[\mathcal{R}]{*} \ell$ (resp. $r' \xrightarrow[\mathcal{R}]{*} r$). Therefore, $a_n(h_n) \xrightarrow[\mathcal{R}]{} \ell' \cdot a_n(h_n) \cdot r' \xrightarrow[\mathcal{R}]{*} \ell \cdot a_n(h_n) \cdot r \xrightarrow[\mathcal{R}]{*} \ell \cdot h' \cdot r = h$. Hence $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$ and (i) holds for h and s.

Case (rpl). The last step of ρ_0 uses $u \to q_{a_1 \ldots a_n}$, and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}$ contains a rule $a_n(x) \to u$, with $u \in Q^*_{\mathcal{A}}$. In this case, the reduction ρ has the following form,

$$h \xrightarrow[\mathcal{A}']{*} u \xrightarrow[\mathcal{A}']{} q_{a_1 \ldots a_n} = s$$

By induction hypothesis (iii) applied to $h \xrightarrow[\mathcal{A}']{*} u$ and by Lemma 3, there exists $h' \in \mathcal{H}(\Sigma)$ such that $h' \xrightarrow[\mathcal{A}]{*} u$ and $h' \xrightarrow[\mathcal{R}]{*} h$. Since $\mathcal{B}$ is assumed clean, there exists $h_1 \in L(\mathcal{B}, q^{a_1})$, and, using the above (rpl) rewrite rule, $a_n(h_1) \xrightarrow[\mathcal{R}]{} h' \xrightarrow[\mathcal{R}]{*} h$. Hence $a_1(h_1) \xrightarrow[\mathcal{B}]{} q$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1 \ldots a_n} h$ and (i) holds for h and s.

Case (del). The last step of ρ_0 uses $q^{a_1 \cdots a_n} \to q_{a_1 \ldots a_n}$ and this transition has been added to Δ' because $\mathcal{R}/\mathcal{A}_0$ contains a rule $a_n(x) \to x$. In this case, the reduction ρ has the following form,

$$h \xrightarrow[\mathcal{A}']{*} q^{a_1 \ldots a_n} \xrightarrow[\mathcal{A}']{} q_{a_1 \ldots a_n} = s$$

By induction hypothesis (ii) applied to $h \xrightarrow[\mathcal{A}']{*} q^{a_1\cdots a_n}$ there exists $h_1 \in \mathcal{H}(\Sigma)$ such that $h_1 \xrightarrow[\mathcal{B}]{*} q^{a_1}$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1\ldots a_n} a_n(h)$. Therefore, $a_1(h_1) \xrightarrow[\mathcal{B}]{} q$ and $a_1(h_1) \xrightarrow[\mathcal{R}]{a_1\ldots a_n} h$, and (i) holds for h and s. □

Corollary 2. $L(\mathcal{A}') \subseteq \mathit{post}^*_{\mathcal{R}/\mathcal{A}}(L)$

Proof. By definition of Q^{f}, $h \in L(\mathcal{A}')$ iff $h \xrightarrow[\mathcal{A}']{*} q \in P^{\mathsf{f}} = Q^{\mathsf{f}}_L$, and $P^{\mathsf{f}} \subseteq P \setminus P_{\mathsf{in}}$. By Lemma 4, case (iii), it follows that $h \in \mathit{post}^*_{\mathcal{R}/\mathcal{A}}(L(\mathcal{B}, q)) \subseteq \mathit{post}^*_{\mathcal{R}/\mathcal{A}}(L)$. □

4.2.2. Completeness.

Lemma 5 (Completeness). *For all $h \in \mathcal{H}(\Sigma)$ and $s \in Q$, if $h \xrightarrow[\mathcal{A}_0]{*} s$ and $h \xrightarrow[\mathcal{R}]{*} h'$, then $h' \xrightarrow[\mathcal{A}']{*} s$.*

Proof. The proof is by induction on the length of the rewrite sequence $h \xrightarrow[\mathcal{R}]{*} h'$ If the length is 0, the result is immediate.
Otherwise, we analyze the last rewrite step. More precisely, assume that the rewrite step has the following form

$$h \xrightarrow[\mathcal{R}]{*} C[a_n(h_n)] \xrightarrow[\mathcal{R}]{} C[r\sigma] = h'$$

for some context $C[\,]$, where the last step applies one rewrite rule $\rho = a_n(x) \to r$ and the substitution σ associates x with $h_n \in \mathcal{H}(\Sigma)$. By induction hypothesis, $C[a_n(h_n)] \xrightarrow[\mathcal{A}']{*} s$. This latter reduction can be decomposed as follows, modulo permutation of transitions,

$$C[a_n(h_n)]_p \xrightarrow[\mathcal{A}']{*} C[a_n(s_1 \cdots s_m)] \xrightarrow[\mathcal{A}']{*} C[a_n(q^{a_1\cdots a_n})] \xrightarrow[\mathcal{A}_0]{} C[q_{a_1\ldots a_n}] \xrightarrow[\mathcal{A}']{*} s$$

where $s_1, \ldots, s_m \in Q$ and $a_1, \ldots, a_{n-1} \in \Sigma$. We show, with a case analysis over ρ, that $r\sigma \xrightarrow[\mathcal{A}']{*} q_{a_1\ldots a_n}$, which implies that $h' = C[r\sigma] \xrightarrow[\mathcal{A}']{*} C[q_{a_1\ldots a_n}] \xrightarrow[\mathcal{A}']{*} s$.

Case (ren). Assume that $\rho = a_n(x) \to b(x)$. In this case, two transitions have been added to $\mathcal{A}'$: $q^{a_1\cdots a_n} \to q^{a_1\cdots a_n b}$ and $q_{a_1\ldots a_n b} \to q_{a_1\ldots a_n}$. Hence we have,

$$r\sigma = b(h_n) \xrightarrow[\mathcal{A}']{*} b(q^{a_1\cdots a_n}) \xrightarrow[\mathcal{A}']{} b(q^{a_1\cdots a_n b}) \xrightarrow[\mathcal{A}_0]{} q_{a_1\ldots a_n b} \xrightarrow[\mathcal{A}']{} q_{a_1\ldots a_n}$$

Case (ac). Assume that $\rho = a_n(x) \to a_n(u \cdot x \cdot v)$, with $u, v \in Q^*_{\mathcal{A}}$. In this case, the following transition has been added to $\mathcal{A}'$: $u \cdot q^{a_1\cdots a_n} \cdot v \to q^{a_1\cdots a_n}$, and $r\sigma = a_n(h_1 \cdot h_n \cdot h_2)$ where $h_1 \xrightarrow[\mathcal{A}]{*} u$ and $h_2 \xrightarrow[\mathcal{A}]{*} v$. Hence, using Lemma 3 for the first steps,

$$r\sigma = a_n(h_1{\cdot}h_n{\cdot}h_2) \xrightarrow[\mathcal{A}_0]{*} a_n(u{\cdot}h_n{\cdot}v) \xrightarrow[\mathcal{A}']{*} a_n(u{\cdot}q^{a_1\cdots a_n}{\cdot}v) \xrightarrow[\mathcal{A}']{} a_n(q^{a_1\cdots a_n}) \xrightarrow[\mathcal{A}_0]{} q_{a_1\ldots a_n}$$

Case (as). Assume that $\rho = a_n(x) \to u \cdot a_n(x) \cdot v$, with $u, v \in Q^*_{\mathcal{A}}$. In this case, the following transition has been added to $\mathcal{A}'$: $u \cdot q_{a_1\ldots a_n} \cdot v \to q_{a_1\ldots a_n}$, and $r\sigma = h_1 \cdot a_n(h_n) \cdot h_2$ where $h_1 \xrightarrow[\mathcal{A}]{*} u$ and $h_2 \xrightarrow[\mathcal{A}]{*} v$. Hence it holds that (using Lemma 3 for the first steps)

$$r\sigma = h_1{\cdot}a_n(h_n){\cdot}h_2 \xrightarrow[\mathcal{A}_0]{*} u{\cdot}a_n(h_n){\cdot}v \xrightarrow[\mathcal{A}']{*} u{\cdot}a_n(q^{a_1\cdots a_n}){\cdot}v \xrightarrow[\mathcal{A}_0]{} u{\cdot}q_{a_1\ldots a_n}{\cdot}v \xrightarrow[\mathcal{A}']{} q_{a_1\ldots a_n}$$

Case (ap). Assume that $\rho = a_n(x) \to b\big(a_n(x)\big)$. In this case, the following vertical transitions have been added to $\mathcal{A}'$: $b(q_{a_1\dots a_n}) \to q_{a_1\dots a_n}$, and we have:

$$r\sigma = b\big(a_n(h_n)\big) \xrightarrow[\mathcal{A}']{*} b\big(a_n(q^{a_1\cdots a_n})\big) \xrightarrow[\mathcal{A}_0]{} b(q_{a_1\dots a_n}) \xrightarrow[\mathcal{A}']{} q_{a_1\dots a_n}$$

Case (rpl). Assume that $\rho = a_n(x) \to u$, with $u \in Q^*_{\mathcal{A}}$. In this case, the following transition has been added to $\mathcal{A}'$: $u \to q_{a_1\dots a_n}$. It holds that $r\sigma \xrightarrow[\mathcal{A}]{*} u$, hence $r\sigma \xrightarrow[\mathcal{A}_0]{*} u \xrightarrow[\mathcal{A}']{} q_{a_1\dots a_n}$, using Lemma 3 for the first steps.

Case (del). Assume that $\rho = a_n(x) \to x$. In this case, the following transition has been added to $\mathcal{A}'$: $q^{a_1\cdots a_n} \to q_{a_1\dots a_n}$, and $r\sigma = h_n \xrightarrow[\mathcal{A}']{*} q^{a_1\cdots a_n} \xrightarrow[\mathcal{A}']{} q_{a_1\dots a_n}$. □

Example 5 shows the problems that can arise when combining in one single rewrite rule two rules of the form (as) and (ac), forcing synchronization of two updates. Note that the rule $a(x) \to c \cdot a(e \cdot x \cdot g) \cdot d$ of this example can be simulated by the 2 rules $a(x) \to c \cdot a'(x) \cdot d$ and $a'(x) \to a(e \cdot x \cdot g)$. The former rule is of the type of Theorem 3 (it combines types (as) and (ren)). The latter (which is not 1-childvar) combines types (ac) and (ren). This shows that such combinations can also lead to the behavior exposed in Example 5.

Corollary 3. *Let $\mathcal{A}$ be a HA over Σ, and L be the language of $\mathcal{A}_L \in$ CFHA, and $\mathcal{R}/\mathcal{A}$ be a* uPHRS. *There exists an effectively computable CFHA recognizing $post^*_{\mathcal{R}/\mathcal{A}}(L)$, of size polynomial in the size of $\mathcal{R}/\mathcal{A}$ and $\mathcal{A}_L$ and exponential in the size of the alphabet Σ.*

Proof. We transform $\mathcal{R}/\mathcal{A}$ into $\hat{\mathcal{R}}/\hat{A}$ as in Lemma 1, and transform L into $\hat{L} = \{\hat{h} \mid h \in L\}$. Then we apply Theorem 3 in order to construct a CFHA $\mathcal{A}'$ recognizing $post^*_{\hat{\mathcal{R}}/\hat{A}}(\hat{L})$.

Then, according to Lemma 1, the CFHA $\mathcal{A}'$ obtained can be transformed into a CFHA recognizing $post^*_{\mathcal{R}/\mathcal{A}}(L)$ by adding new transitions: for every CFHA transition of $\mathcal{A}'$ of the form $\hat{a}(q_1) \to q$, we add n copies $a_1(q_1) \to q$, $\dots$, $a_n(q_1) \to q$ where $\hat{a}$ is the representative of $\{a_1, \dots, a_n\}$. □

4.3. Backward Rewrite Closure and Typechecking

The following theorem is a direct consequence of Theorem 1 in [16] on the rewrite closure of HA languages under monadic HRS.

Theorem 4. *Let $\mathcal{A}$ be a HA over Σ, and L be the language of $\mathcal{A}_L \in$ HA, and $\mathcal{R}/\mathcal{A}$ be a* uPHRS. *There exists an effectively computable HA recognizing $pre^*_{\mathcal{R}/\mathcal{A}}(L)$.*

Corollary 4. *Typechecking is decidable for* uPHRS*, with CFHA languages as input types and HA languages as output types.*

Proof. Let $\mathcal{R}/\mathcal{A}$ be a uPHRS. Let L_{in} be the input type, recognized by a CFHA $\mathcal{A}_{\mathsf{in}}$, and let L_{out} be the output type, recognized by an HA $\mathcal{A}_{\mathsf{out}}$. Using Corollary 3, on can build a CFHA recognizing $post^*_{\mathcal{R}/\mathcal{A}}(L_{\mathsf{in}})$. By closure of HA languages under complementation, on can also build a HA recognizing the complement $\overline{L_{\mathsf{out}}}$ of L_{out}. Then, there exists a CFHA recognizing $post^*_{\mathcal{R}/\mathcal{A}}(L_{\mathsf{in}}) \cap \overline{L_{\mathsf{out}}}$, and the typechecking of $\mathcal{R}/\mathcal{A}$ *wrt* L_{in} and L_{out} is equivalent to the emptiness of this latter CFHA (a decidable problem). □

5. Verification of Consistency of ACP for XML Updates

In this last section we study some models of Access Control Policies (ACP) for the update operations defined in Section 4, and the verification problems for these ACP. We consider two kind of formalisms from the literature for the specification of XML ACPs based on PHRS. The first formalism is the most widespread. It consists in defining an ACP as a set of update rules, partitioned into authorized and forbidden operations. The second one is a more recent proposal of [13] based on [17], where the ACP is defined by adding security annotations to a DTD.

5.1. Local Consistency of Rule-based ACPs

An ACP for XML updates can be defined by a pair $(\mathcal{R}_{\mathsf{a}}/\mathcal{A}, \mathcal{R}_{\mathsf{f}}/\mathcal{A})$ of uPHRS, where $\mathcal{R}_{\mathsf{a}}$ contains allowed operations and $\mathcal{R}_{\mathsf{f}}$ contains forbidden operations (see *e.g.* [22]). Such an ACP is called *inconsistent* [13, 22] if some forbidden operation can be simulated through a sequence of allowed operations, *i.e.* if there exists $t, u \in \mathcal{T}(\Sigma)$ such that $t \xrightarrow[\mathcal{R}_{\mathsf{f}}/\mathcal{A}]{} u$ and $t \xrightarrow[\mathcal{R}_{\mathsf{a}}/\mathcal{A}]{*} u$.

Example 11. Assume that in the $\mathsf{hospital}$ document of Example 6, it is forbidden to rename a $\mathsf{patient}$, that is the following update of (rpl) is forbidden: $\mathsf{name}(x) \to q_{\mathsf{name}}$. If the following updates are allowed: $\mathsf{patient}(x) \to \varepsilon$ for deleting a patient, and $\mathsf{hospital}(x) \to \mathsf{hospital}(x \cdot q_{\mathsf{patient}})$ to insert a new $\mathsf{patient}$, then we have an inconsistency in the sense of [22], since the effect of the forbidden update can be obtained by a combination of allowed updates. ◇

Using the results of Section 4, we can decide the above problem individually for trees of $\mathcal{T}(\Sigma)$. More precisely, we solve the following problem called *local inconsistency*:

Definition 3. Given a HA $\mathcal{A}$ over Σ and a tree $t \in \mathcal{T}(\Sigma)$, an ACP $(\mathcal{R}_{\mathsf{a}}/\mathcal{A}, \mathcal{R}_{\mathsf{f}}/\mathcal{A})$ is locally inconsistent if there exists $u \in \mathcal{T}(\Sigma)$ such that $t \xrightarrow[\mathcal{R}_{\mathsf{f}}/\mathcal{A}]{} u$ and $t \xrightarrow[\mathcal{R}_{\mathsf{a}}/\mathcal{A}]{*} u$.

Hence the term *local* in Definition 3 means consistency with respect to a given tree t, *i.e.* consistency in general is local consistency for all $t \in \mathcal{T}(\Sigma)$.

Theorem 5. *Local inconsistency is decidable in PTIME for ACPs specified by pairs of* uPHRS.

Proof. It can be easily shown that the set $\{u \in \mathcal{T}(\Sigma) \mid t \xrightarrow[\mathcal{R}_{\mathsf{f}}/\mathcal{A}]{} u\}$ is the language of a HA of size polynomial and constructed in PTIME on the sizes of $\mathcal{A}$, $\mathcal{R}_{\mathsf{f}}$ and t. By Theorem 3, $post^*_{\mathcal{R}_{\mathsf{a}}/\mathcal{A}}(\{t\})$ is the language of a CFHA of polynomial size and constructed in polynomial time on the sizes of $\mathcal{A}$, $\mathcal{R}_{\mathsf{a}}$ and t. The ACP is locally inconsistent w.r.t. t iff the intersection of the two above languages is not empty, and this property can be tested in PTIME. □

It is shown in [13] that inconsistency is undecidable for an ACP defined by a pair of rewrite systems $(\mathcal{R}_{\mathsf{a}}, \mathcal{R}_{\mathsf{f}})$ of a kind strictly more general than the above uPHRS. Roughly, they extend the uPHRS with the possibility to select the rewrite positions by XPath expressions. Moreover, for such rewrite systems, the reachability problem (whether a given tree t can be obtained from a given tree

s using instances of rules of $\mathcal{R}_{\mathsf{a}}$ which are not in $\mathcal{R}_{\mathsf{f}}$) is also undecidable [31], therefore local consistency is undecidable as well in this case. A decidable fragment is also presented in [31].

A related class of TRS are the (prefix-) controlled TRS of [32, 33] where rewrite positions are selected for each rewrite rules using a Query Automaton [34] or a regular expression. For example, with such TRS, one can specify that a particular rule of the form $a(x) \to u$ can be applied only at a position labeled by a and without a b occurring on the path to the root. The application of the results of decidability and rewrite closure in [32, 33] to the decision of inconsistency in the context of hedge rewriting is an interesting question to investigate.

5.2. Local Consistency of DTD-based ACPs

A DTD over Σ is function D that maps Σ to regular expressions over Σ. This standard formalism for defining XML types is strictly less expressive than HA [3]. The dependency graph of a DTD D is a directed graph on the set of vertices Σ such that the set of edges contains all (a, b) such that b occurs in the regular expression $D(a)$. A DTD is *non recursive* if this graph is acyclic.

Following the principle of DTD-based ACPs [17], [13] have proposed the language $\mathsf{XACU}_{\mathsf{annot}}$ for the definition of ACP for XML updates in presence of a DTD D. The idea is to add to D some security annotations specifying the authorizations for the update operations for XML documents valid for D. This formalism of [17, 13] imposes the condition that every document t to which we want to apply an update operation (under the given ACP) must be valid for the DTD D.

In our rewrite-based formalism, the latter condition may be expressed by adding global constraints to the parametrized rewrite rules of Section 4. These global constraints restrict the rewrite relation to trees in a given HA language. Theorem 6 below shows that, unfortunately, adding such constraints to parametrized rewrite rules of type (ren) or (rpl) makes the reachability undecidable.

A hedge rewriting system over Σ, parametrized by a HA $\mathcal{A}$ and with global constraints (PGHRS) is given by a PHRS, denoted $\mathcal{R}/\mathcal{A}$, (as defined in Section 4) and a HA language $L \subseteq \mathcal{H}(\Sigma)$. We say that L is the constraint of $\mathcal{R}$. The rewrite relation generated by the PGHRS is defined as the restriction of the relation defined in Section 4 to ground hedges such that for the application of a rule $\ell \to r \in \mathcal{R}/\mathcal{A}$ to a hedge h, we require that $h \in L$.

Theorem 6. *Reachability is undecidable for* PGHRS*'s with rules of type* (ren) *or* (as) *and constraint given by a non recursive DTD.* □

Proof. The proof is a variant of the one given by A. Spelten [35] for subtree and flat prefix rewriting. We reduce the halting problem of a Deterministic Turing Machine (TM) $\mathcal{M}$ that works on half a tape (unbounded on the right).

TM configurations are encoded as flat trees. We consider the same tape alphabet $\Gamma = \{0, 1, \flat\}$, ($\flat$ is the blank symbol) of $\mathcal{M}$, let $S = \{s_1, s_2, \ldots, s_n\}$ be the state set of $\mathcal{M}$ and Θ be the set of instructions of $\mathcal{M}$. We consider the following alphabet Σ for the representation of the configurations of $\mathcal{M}$.

$$\Sigma := \{g\} \cup \{0, 1, \flat\} \cup (S \times \Gamma) \cup (\Theta \times \Gamma).$$

For instance, the TM configuration with tape $abcde \cdot \flat \cdot \flat \ldots$, symbol d under head, state s, will be represented by the following flat tree of $\mathcal{T}(\Sigma)$:

$$g(abc\langle s, d\rangle e \cdot \flat \cdot \flat).$$

We shall also use a trivial HA $\mathcal{A} = (\Sigma, Q, Q, \Delta)$ which recognizes only constant symbols by taking $Q = \{p_\sigma \mid \sigma \in \Sigma\}$ and $\Delta = \{\sigma \to p_\sigma \mid \sigma \in \Sigma\}$.

We define a PGHRS $\mathcal{R}/\mathcal{A}$ such that every transition of $\mathcal{M}$ can be simulated by a sequence of (at most three) rewrite steps with $\mathcal{R}/\mathcal{A}$. Let us first introduce some standard auxiliary PHRS rules and some word regular languages for controlling rule applications. We have three cases to consider:

For each instruction θ_1 of $\mathcal{M}$ of type: *"In state s_1 reading a_1 go to state r_1 and write b_1"*, we define the following HRS rule: $\langle s_1, a_1\rangle(x) \to \langle r_1, b_1\rangle(x)$.
We also define the regular word language $L_{\langle s_1, a_1\rangle} = \Gamma^*\langle s_1, a_1\rangle \cdot \Gamma^*$.

For each instruction θ_2 of $\mathcal{M}$ of type: *"In state s_2 reading a_2 go to state r_2 and move right"*, we define the following PHRS rules of types (ren) and (as) (we recall that $p_\flat$ is a state of $\mathcal{A}$):

$$\begin{array}{rclcrcl} b_2(x) & \to & \langle\theta_2, b_2\rangle(x) & \text{for all } b_2 \in \Gamma & \flat(x) & \to & \flat(x)\ p_\flat \\ \langle\theta_2, b_2\rangle(x) & \to & \langle r_2, b_2\rangle(x) & \text{for all } b_2 \in \Gamma & \langle s_2, a_2\rangle(x) & \to & a_2(x) \end{array}$$

We also define the regular word languages:

$$\begin{array}{rcl} L_{\langle s_2, a_2\rangle} & = & \Gamma^* \cdot \langle s_2, a_2\rangle \cdot \Gamma^* \\ L_{\langle \theta_2, a_2\rangle} & = & \Gamma^* \cdot \langle \theta_2, a_2\rangle \cdot \Gamma^* \\ L_{\langle s_2, a_2\rangle\langle\theta_2, b_2\rangle} & = & \Gamma^* \cdot \langle s_2, a_2\rangle\langle\theta_2, b_2\rangle \cdot \Gamma^* \quad \text{for all } \ b_2 \in \Gamma. \end{array}$$

For each instruction θ_3 of $\mathcal{M}$ of type: *"In state s_3 reading a_3 go to state r_3 and move left"*, we define the following HRS rules:

$$\begin{array}{rclcrcl} b_3(x) & \to & \langle\theta_3, b_3\rangle(x) & \text{for all } b_3 \in \{0,1\} & \langle s_3, a_3\rangle(x) & \to & a_3(x) \\ \langle\theta_3, b_3\rangle(x) & \to & \langle r_3, b_3\rangle(x) & \text{for all } b_3 \in \{0,1\} & & & \end{array}$$

We also define the regular word languages:

$$\begin{array}{rcl} L_{\langle s_3, a_3\rangle} & = & \Gamma^* \cdot \langle s_3, a_3\rangle \cdot \Gamma^* \\ L_{\langle \theta_3, a_3\rangle} & = & \Gamma^* \cdot \langle \theta_3, a_3\rangle \cdot \Gamma^* \\ L_{\langle \theta_3, b_3\rangle\langle s_3, a_3\rangle} & = & \Gamma^* \cdot \langle \theta_3, b_3\rangle\langle s_3, a_3\rangle \cdot \Gamma^* \text{ for all } b_3 \in \{0,1\}. \end{array}$$

The constraint of the PGHRS will be defined by the non recursive DTD $D : g \to L$ where L is the finite union of the regular languages associated with the instructions of $\mathcal{M}$ as above. Since the machine to be simulated is deterministic, the union is disjoint.

Our final PGHRS is given by $\mathcal{R}/\mathcal{A}$ and L so that the rewrite rules in $\mathcal{R}/\mathcal{A}$ can only be applied to trees satisfying the DTD D. With the above constraint, the PGHRS rules of $\mathcal{R}/\mathcal{A}$ can only be applied to trees valid for the DTD D, ensuring a correct chaining for the application of these rules.

By case inspection we can show that for any couple of TM configurations T_1, T_2 and their respective tree encodings t_1, t_2, there is a sequence of transitions from T_1 to T_2 iff $t_1 \xrightarrow[\mathcal{R}/\mathcal{A}]{*} t_2$. The theorem follows. □

Note that the above result also holds for PGHRS's whose rules are ground (without variables nor parameters): in the above rewrite rules, every variable x could be replaced by the empty hedge ε, and every parameter such as $p_\flat$ could be replaced by the corresponding ground tree $\flat$. Hence the above result can be contrasted with the decidability of reachability for ground tree rewriting [36].

Corollary 5. *Local inconsistency is undecidable for* PGHRS*'s with rules of type* (ren) *or* (as) *and with constraint given by a non recursive DTD.*

This is in contrast with a result [22] of decidability of consistency for policies tied to a different class of DTD, and which do not include updates of type (ren).

6. Future Works

As for future works on $\mathsf{CF}^1\mathsf{HA}$ languages several directions deserve to be followed. A first direction might be to derive pumping properties for these classes of languages. A second direction would be to look for an analogue of Parikh characterization for the number of different symbols occurring in the hedges of given $\mathsf{CF}^1\mathsf{HA}$ languages. One may define and study HRS with counting constraints on horizontal and vertical paths.

One may wonder whether our result on rewrite closure computation (Theorem 3) could be adapted to compute regular over-approximations of output types for XML transformations, leading to an approximating forward type inference algorithm, in an approach similar to *e.g.* [9].

Finally, it would be is worth investigating the parallel rewriting of [18], on all a-nodes, since it is closer to the semantics of XQUF, and get an analogous of Theorem 3 for the parallel rewrite closure.

Acknowledgements

The authors wish to thank the reviewers of [23] and Giorgio Delzanno for valuable discussions on rewrite based models and verification techniques for XQUF.

References

[1] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, Available on: `http://tata.gforge.inria.fr/`, 2007. Release October, 12th 2007.

[2] M. Murata, Hedge automata: a formal model for XML schemata, `http://www.xml.gr.jp/relax/hedge_nice.html`, 2000.

[3] T. Schwentick, Automata for XML - A Survey, J. Comput. Syst. Sci. 73 (2007) 289–315.

[4] T. Touili, Computing transitive closures of hedge transformations, International Journal of Critical Computer-Based Systems (IJCCBS) 3 (2012) 132–150.

[5] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, J. Comput. Syst. Sci. 37 (1988) 367–394.

[6] A. Bouajjani, B. Jonsson, M. Nilsson, T. Touili, Regular model checking, in: Proc. of the 12th Int. Conf. on Computer Aided Verification, volume 1855 of *Lecture Notes in Computer Science*, 2000, pp. 403–418.

[7] G. Feuillade, T. Genet, V. Viet Triem Tong, Reachability Analysis over Term Rewriting Systems, Journal of Automated Reasoning 33 (3-4) (2004) 341–383.

[8] A. Bouajjani, T. Touili, On computing reachability sets of process rewrite systems, in: Proceedings 16th Int. Conf. Term Rewriting and Applications (RTA), volume 3467 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 484–499.

[9] T. Touili, Computing transitive closures of hedge transformations, in: Proc. 1st International Workshop on Verification and Evaluation of Computer and Communication Systems (VECOS), eWIC Series, British Computer Society, 2007.

[10] T. Milo, D. Suciu, V. Vianu, Typechecking for XML Transformers, Journal of Comp. Syst. Sci. 66 (2003) 66–97.

[11] A. Tozawa, Towards static type checking for xslt, in: DocEng '01: Proceedings of the 2001 ACM Symposium on Document engineering, ACM, New York, NY, USA, 2001, pp. 18–27.

[12] D. Chamberlin, J. Robie, M. Dyck, D. Florescu, J. Melton, J. Siméon, Xquery update facility 1.0, W3C Recommendation. `http://www.w3.org/TR/xquery-update-10/`, 2011.

[13] I. Fundulaki, S. Maneth, Formalizing XML access control for update operations, in: Proc. of the 12th ACM symposium on Access control models and technologies (SACMAT), ACM, 2007, pp. 169–174.

[14] L. Bravo, J. Cheney, I. Fundulaki, ACCOn: checking consistency of XML write-access control policies, in: Proc. 11th Int. Conf. on Extending Database Technology (EDBT), volume 261 of *ACM Int. Conf. Proc. Series*, ACM, 2008, pp. 715–719.

[15] F. Jacquemard, M. Rusinowitch, Rewrite-based verification of XML updates, in: Proc. of the 12th ACM SIGPLAN Int. Symp. on Principles and Practice of Declarative Programming (PPDP), ACM, 2010, pp. 119–130.

[16] F. Jacquemard, M. Rusinowitch, Closure of Hedge-automata languages by Hedge rewriting, in: Proc. of the 19th RTA, volume 5117 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 157–171.

[17] W. Fan, C.-Y. Chan, M. Garofalakis, Secure XML querying with security views, in: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM, 2004, pp. 587–598.

[18] A. Solimando, G. Delzanno, G. Guerrini, Automata-based Static analysis of XML Document Adaptation, in: Proceedings 3d International Symposium on Games, Automata, Logics and Formal Verification, volume 96, 2012, pp. 85–98.

[19] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, Securing XML documents, in: Proceedings of the 7th International Conference on Extending Database Technology (EDBT), volume 1777 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 121–135.

[20] S. C. Lim, S. H. Son, Access control of XML documents considering update operations, in: Proc. of ACM Workshop on XML Security, 2003.

[21] M. Murata, A. Tozawa, M. Kudo, S. Hada, XML access control using static analysis, ACM Trans. Inf. Syst. Secur. 9 (2006) 292–324.

[22] L. Bravo, J. Cheney, I. Fundulaki, R. Segovia, Consistency and repair for xml write-access control policies, VLDB Journal 21 (2012) 843–867.

[23] F. Jacquemard, M. Rusinowitch, Rewrite closure and cf hedge automata, in: Proceedings of the 7th International Conference on Language and Automata Theory and Applications (LATA), volume 7810 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 371–382.

[24] J. Cocke, J. T. Schwartz, Programming languages and their compilers: Preliminary notes, Technical Report, Courant Institute of Mathematical Sciences, New York University, 1970.

[25] D. H. Younger, Recognition and parsing of context-free languages in time n3, Information and Control 10 (1967) 189–208.

[26] H. Ohsaki, H. Seki, T. Takai, Recognizing boolean closed a-tree languages with membership conditional rewriting mechanism, in: Proc. of the 14th RTA, volume 2706 of *Lecture Notes in Computer Science*, Springer Verlag, 2003, pp. 483–498.

[27] M. Bojanczyk, I. Walukiewicz, Forest algebras, in: Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas], volume 2, Amsterdam University Press, 2008, pp. 107–132.

[28] E. Filiot, J.-M. Talbot, S. Tison, Tree automata with global constraints, in: 12th International Conference in Developments in Language Theory (DLT 2008), volume 5257 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 314–326.

[29] W. Karianto, C. Löding, Unranked tree automata with sibling equalities and disequalities, in: Proceedings of 34th International Colloquium on Automata, Languages and Programming (ICALP), volume 4596 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 875–887.

[30] J.-L. Coquide, M. Dauchet, R. Gilleron, S. Vagvolgyi, Bottom-up tree pushdown automata : Classification and connection with rewrite systems, Theoretical Computer Science 127 (1994) 69–98.

[31] N. Moore, Computational complexity of the problem of tree generation under fine-grained access control policies, Inf. Comput. 209 (2011) 548–567.

[32] F. Jacquemard, Y. Kojima, M. Sakai, Controlled term rewriting, in: Proceedings of the 8th International Symposium Frontiers of Combining Systems (FroCoS), volume 6989 of *Lecture Notes in Artificial Intelligence*, Springer, 2011, pp. 179–194.

[33] F. Jacquemard, Y. Kojima, M. Sakai, Term rewriting with prefix context constraints and bottom-up strategies, in: 25th International Conference on Automated Deduction, CADE, Lecture Notes in Computer Science, Springer, 2015.

[34] F. Neven, T. Schwentick, Query automata over finite trees, Theoretical Computer Science 275 (2002) 633–674.

[35] A. Spelten, Rewriting Systems over Unranked Trees, Master's thesis, Diplomarbeit, RWTH Aachen, 2006.

[36] R. Gilleron, Decision problems for term rewrite systems and recognizable tree languages, in: 8^{th} Annual Symposium on Theoretical Aspects of Computer Science, volume 480, 1991, pp. 148–159.