

On Hedge Automata Languages and Hedge Rewriting

COPS Meeting

Florent Jacquemard & Michael Rusinowitch

February 5, 2008

Regular Tree Model Checking

Regularity preservation:

- a **regular** tree language L (set of **configurations**)
- a rewrite system \mathcal{R} (set of **transitions**)
- compute $\mathcal{R}^*(L)$ (\mathcal{R}^* is the reflexive-transitive closure of \mathcal{R}).
Is it a **regular** tree language?
- or an over-approximation.

Applications to system/programs verification:

- symbolic **reachability**: $s \xrightarrow{\mathcal{R}^*} t$?
- regular tree M.C. = **safety**: $\mathcal{R}^*(L_{\text{init}}) \cap L_{\text{bad}} \neq \emptyset$

over-approximation: OK for validation but false bugs/attacks.

Program Verification

Applications:

- analysis of tree shaped networks

[Abdulla et al, Bouajjani & Touili, Lakhnech]

token protocol: $\odot(\odot(x_1, x_2), \odot(x_3, x_4)) \rightarrow \odot(\odot(x_1, x_2), \odot(x_3, x_4))$

- analysis of multithreaded programs

[Lugiez & Schoebelen, Bouajjani & Touili]

procedure call: $X \rightarrow Y \cdot Z$

thread creation: $X \rightarrow Y \parallel Z$

Case of Ranked Terms

- ranked signature: function symbols come with fixed **arity**
- ex: process composition, \cdot and \parallel are binary
- **tree automata** completion:
 - ▶ start with a TA for L ,
 - ▶ add TA transitions (based on rules of \mathcal{R}) until fixpoint,
 - ▶ you obtain a TA for $R^*(L)$ or over-approximation.
- exact for monadic & linear TRS [Salomaa 88].
- exact for monadic & right-linear (**non left-linear**) TRS [Nagaya Toyama 99].

Case of Unranked Ordered Terms

- unranked alphabet: function symbols without arity,
= labels for arbitrary trees (**unbounded width**)
- arbitrary number of processes, XML documents...
- **hedge automata** completion:
 - ▶ for relabeling hedge-transducers
[d'Orso & Touili 06]
 - ▶ for **linear** rewrite rules, non-terminating → over-approximation
[Touili 07]

Our goal

in the context of unranked ordered terms and hedge automata:

Goal

identify the cases where $\mathcal{R}^*(L)$ is **exactly** a regular language

- 1 positive result for a class of non-linear rewrite systems, generalizes both constructions of [Touili 07] and [Nagaya Toyama 99],
- 2 positive result for a generalization of hedge-automata other construction
- 3 several negative results on extensions of 1. and 2.

Hedges and Rewriting

- Σ unranked alphabet, \mathcal{X} set of variables
- $\mathcal{T}(\Sigma, \mathcal{X})$ set of **trees** = $\{a(h) \mid a \in \Sigma, h \in \mathcal{H}(\Sigma, \mathcal{X})\}$
- $\mathcal{H}(\Sigma, \mathcal{X})$ set of **hedges** or finite sequences of terms
- substitution: mapping from \mathcal{X} to $\mathcal{H}(\Sigma, \mathcal{X})$

hedge rewriting generalizes term rewriting and word rewriting.

Definition (HRS)

An hedge rewriting system (HRS) is a set of rewrite rules of the form $\ell \rightarrow r$ where $\ell, r \in \mathcal{T}(\Sigma, \mathcal{X})$

Examples:

$$\begin{aligned}\mathcal{R} &= \{g(x) \rightarrow x\} : & g(abc) &\xrightarrow{\mathcal{R}} abc \\ \mathcal{R} &= \{g(axa) \rightarrow h(x)\} : & g(abbca) &\xrightarrow{\mathcal{R}} h(bbcb) \\ \mathcal{R} &= \{g(x) \rightarrow g(axb)\} : & g(c) &\xrightarrow[\mathcal{R}]{*} g(a^n cb^n)\end{aligned}$$

A rewrite rule $\ell \rightarrow r$ is called

- **left-linear** (resp. **right-linear**, **linear**) if ℓ (resp. r , both) is linear,
- **left-ground** (resp. **right-ground**) if $\ell \in \mathcal{T}(\Sigma)$ (resp. $r \in \mathcal{T}(\Sigma)$),
- **collapsing** if $r \in \text{var}(\ell)$,
- **variable-disjoint** if $\text{var}(\ell) \cap \text{var}(r) = \emptyset$,
- **context-free** if $\ell = f(x)$ with $x \in \text{var}(r)$
- **inverse context-free** if $r \rightarrow \ell$ is context-free.

Examples

$$\text{doc}(x) \rightarrow \text{doc}(\langle \text{a} \rangle x \langle /\text{a} \rangle)$$

$$\text{doc}(x \langle \text{comment} \rangle y \langle /\text{comment} \rangle) \rightarrow \text{doc}(x)$$

$$\text{doc}(\langle \text{todo} \rangle x \langle /\text{todo} \rangle y \langle \text{done} \rangle x \langle /\text{done} \rangle) \rightarrow \text{doc}(y)$$

$$\text{users}(\text{user}(\text{id}(x) x_1) \text{user}(\text{id}(x) x_2) y) \rightarrow \text{users}(y)$$

Hedge Automaton: example

Set of:

- trees labeled with $\{a, b\}$.
- of depth 1,
- with a a at the root,
- with an even number of leaves, all labelled by b .
- i.e. $a, a(bb), a(bbbb), \dots$

Finite description: $a((bb)^*)$

Hedge Automata (HA)

Definition

A *hedge automaton* is a tuple $\mathcal{A} = (Q, \Sigma, Q^f, \Delta)$ where Q is a set of states, Σ is an unranked alphabet, $Q^f \subseteq Q$ is a set of final states, and Δ is a set of transitions of the form $f(L) \rightarrow q$ where $f \in \Sigma, q \in Q$ and $L \subseteq Q^*$ is a **regular** word language.

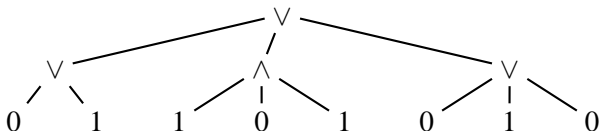
We have $t = C[f(q_1 \dots q_n)] \xrightarrow{\mathcal{A}} t' = C[q]$ for some context $C[x]$, if $f(L) \rightarrow q$ is a transition and $q_1 \dots q_n \in L$.

Hedge Automaton: example 2

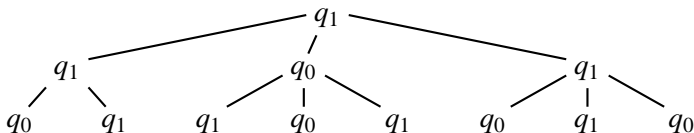
$$\Sigma = \{\vee, \wedge, 0, 1\}, Q = \{q_0, q_1\}, Q^f = \{q_1\},$$

$$\Delta = \begin{array}{ll} \wedge((q_1|q_0)^*q_0(q_1|q_0)^*) & \rightarrow q_0 & \wedge(q_1^*q_1) & \rightarrow q_1 \\ \vee((q_1|q_0)^*q_1(q_1|q_0)^*) & \rightarrow q_1 & \vee(q_0^*q_0) & \rightarrow q_0 \\ 0(\epsilon) & \rightarrow q_0 & 1(\epsilon) & \rightarrow q_1 \end{array}$$

Tree



Run



Hedge Automaton: Properties

- hedge automata are closed under Boolean operations,
- membership and emptiness are decidable for HA
[Murata 00]

Unranked vs Ranked Terms

unranked trees labelled by Σ

$$t = a(t_1, \dots, t_n)$$

Hedge Automata

Hedge Rewriting Systems

terms over $\mathcal{T}(\Sigma : 0 \cup \{\# : 2\})$

$$\tilde{t} := \#(\dots \#(\#(a, \tilde{t}_1), \tilde{t}_2), \dots, \tilde{t}_n))$$

Tree Automata

NOT TRS!

XML applications

Validation	DTD, XML Schema Check whether an XML document is of given type
Querying	XPath , XQuery Extract information from an XML document
Transformation	XSLT, XDuce, CDuce Construct a new XML document from a given one

Example: DTD Validation

DOCTYPE Author

ELEMENT Author (Name,Bio,Book*)

ELEMENT Bio (Born, Died?)

ELEMENT Born (When,Where)

ELEMENT Died (When,Where)

ELEMENT Book (Title, Year)

Corresponding tree automaton

$$\Sigma = \{Author, Name, Bio, \dots\}$$
$$Q = \{Author, Name, Bio, \dots\}$$
$$Q^f = \{Author\}$$
$$Author(Name.Bio.Book^*) \rightarrow Author$$
$$Bio(Born + Born.Died) \rightarrow Bio\dots$$

Example: XML Typechecking

Tree transformation

$$\begin{array}{lcl} \text{Author}(\text{Name}(x), \text{Bio}(y), z) & \rightarrow & \text{Author}'(z) \\ \text{Book}(u, v) & \rightarrow & u \end{array}$$

DOCTYPE Author'

[ELEMENT Author' (Title*)]

Typechecking

Given two tree languages L_{in} , L_{out} and a transducer T ,
does $T(L_{in}) \subseteq L_{out}$?

general proof technique: show that $T^{-1}(L_{out}) \supseteq L_{in}$ and is regular.

Preservation of HA Languages

- **inverse context-free** rule: $\ell \rightarrow f(x)$, $x \in \text{var}(\ell)$
- ex: $\text{users}(\text{user}(\text{id}(x) x_1) \text{user}(\text{id}(x) x_2) y) \rightarrow \text{users}(y)$

Theorem

Given an HA language $L \subseteq \mathcal{T}(\Sigma)$ and an HRS \mathcal{R} whose rules are

- *inverse context-free*
- *or right-linear and variable-disjoint,*

$\mathcal{R}^(L)$ is an HA language.*

Corollary

Reachability and regular hedge model-checking are decidable for such HRS.

Proof idea

Generalization of the constructions of

[Touili 07]: HA completion

[Nagaya Toyama 99]: TA completion for non-left-linear TRS (using deterministic TA).

- Let $\mathcal{A}_L = (Q_L, Q_L^f, \Delta_L)$ be a complete HA recognizing L .
- Compute an HA for each rhs r of \mathcal{R} : $\mathcal{A}_r = (Q_r, Q_r^f, \Delta_r)$
- Let $\mathcal{A} := (Q, Q^f, \Delta)$ with

$$Q := Q_L \uplus \bigsqcup_{r \in \text{rhs}(\mathcal{R})} Q_r, \Delta := \Delta_L \uplus \bigsqcup_{r \in \text{rhs}(\mathcal{R})} \Delta_r$$

- Construct a deterministic HA (subset construction) $\mathcal{A}_d = (Q_d, Q_d^f, \Delta_d)$ recognizing $L(\mathcal{A})$.
- Complete $\mathcal{A}_0 = \mathcal{A}_d$ into \mathcal{A}_1, \dots , according to the rules of \mathcal{R} , preserving determinism.

Collapsing Rules

collapsing rule: $\ell \rightarrow x, x \in \text{var}(\ell)$

Collapsing TRS preserve TA languages [Nagaya Toyama 99] but

Theorem

Collapsing HRS do not preserve HA languages

$\Sigma = \{f, g, a, b, c\}, \mathcal{R} = \{g(x) \rightarrow x\},$

L language of the HA

$$\mathcal{A} = \left(\{q, q_1, q_2, q_f\}, \{q_f\}, \left\{ \begin{array}{l} c \rightarrow q, a \rightarrow q_1, b \rightarrow q_2, \\ g(q_1 q q_2) \rightarrow q, f(q) \rightarrow q_f \end{array} \right\} \right)$$

$$\mathcal{R}^*(L) \cap \{f(a^* c b^*)\} = \{f(a^n c b^n) \mid n \geq 0\}.$$

Flat Rules

flat rule: $\ell \rightarrow r$ with ℓ and r of depth 1.

Flat & linear TRS preserve TA languages but

Theorem

Flat and linear HRS do not preserve HA languages

$$\Sigma = \{g, a, b\}, \mathcal{R} = \{g(x) \rightarrow g(axb)\}, L = \{g(c)\}, \\ \mathcal{R}^*(L) = \{g(a^n c b^n) \mid n \geq 0\}.$$

Example

$\mathcal{R}^*(L)$ not recursive with \mathcal{R} flat linear HRS and L HA language.

$\mathcal{R}^*(L)$ = Turing machine configurations :

$$g(x a q y) \rightarrow g(x q' a' y)$$

Preservation of Context-Free Languages

For **word** languages word rewriting, there are several cases where:

- a class of \mathcal{R} preserves regular languages
- the symmetric class of \mathcal{R}^{-1} preserves context-free languages

Example:

- **inverse-CF** RS preserves **regular** languages

$$wx \rightarrow ax, wx \rightarrow x, a \in \Sigma, w \in \Sigma^*.$$

- **CF** RS preserves **CF** languages

$$ax \rightarrow wx, x \rightarrow wx, a \in \Sigma, w \in \Sigma^*.$$

Context-Free Hedge Automata (CF-HA)

Generalization of HA with CF horizontal languages.

Definition

A *context-free hedge automaton* is a tuple $\mathcal{A} = (Q, \Sigma, Q^f, \Delta)$ where Q is a set of states, Σ is an unranked alphabet, $Q^f \subseteq Q$ is a set of final states, and Δ is a set of transitions of the form $f(L) \rightarrow q$ where $f \in \Sigma$, $q \in Q$ and $L \subseteq Q^*$ is a **context-free** language.

- CF-HA are **not** closed under intersection (unlike HA)
- membership and emptiness are decidable for CF-HA
- a CF-HA language is a regular tree language modulo associative symbols [Ohsaki 03]

Preservation of CF-HA Languages

- **context-free** rule: $f(x) \rightarrow r, x \in \text{var}(r)$
- ex: $\text{doc}(x) \rightarrow \text{doc}(\langle a \rangle x \langle /a \rangle)$

Theorem

Given a CF-HA language L and an HRS \mathcal{R} whose rule are

- *context-free,*
- *right-linear,*
- *for all $f(x) \rightarrow r \in \mathcal{R}$, x occurs in r at depth at most 1,*

$\mathcal{R}^*(L)$ is an CF-HA language.

proof: completion of the CF-grammars (acc. to \mathcal{R}) in CF-HA transition rules.

Preservation of CF-HA Languages (csq)

Corollary

Reachability and regular hedge model-checking are decidable for such restricted CF HRS.

The intersection of a CF-HA language and an HA language is a CF-HA language, and emptiness of CF-HA is decidable.

Restriction on right variable is necessary

Theorem

HRS \mathcal{R} whose rule are

- *context-free,*
- *linear,*

do not preserve CF-HA languages.

$$\mathcal{R} = \{f(x) \rightarrow g(f(ax))\},$$

$$L = \{f(c)\},$$

$$\mathcal{R}^*(L) = \left\{ \underbrace{g(g(\dots g(f(a^n c))))}_n \mid n \in \mathbb{N} \right\} \text{ is not CF-HA}$$

(pumping argument).

Right-linearity is necessary

Theorem

HRS \mathcal{R} whose rule are

- *context-free,*
- *for all $f(x) \rightarrow r \in \mathcal{R}$, x occurs in r at depth at most 1,*

do not preserve CF-HA languages.

$$\mathcal{R} = \{f(x) \rightarrow f(xx)\},$$

$$L = \{f(a)\},$$

$$\mathcal{R}^*(L) = \{f(a^n) \mid n = 2^k, k \geq 0\} \text{ is not CF-HA.}$$

Combining CF and inverse-CF rules

Example

$\mathcal{R}^*(L)$ not recursive with \mathcal{R} HRS whose rules are linear and either inverse-CF or CF restricted.

reduction of PCP

$$f_0(x) \rightarrow f_0(\tilde{u}_i x v_i)$$

$$f_0(x) \rightarrow f_1(x)$$

$$f_1(axa) \rightarrow f_2(x)$$

$$f_2(axa) \rightarrow f_2(x)$$

$$f_0(c) \xrightarrow{\mathcal{R}}^* f_2(c) \text{ iff } \exists \text{ solution.}$$

Conclusion

Summary:

- inverse context-free HRS preserve HA languages
- context-free, right-linear HRS w. restriction on right-variables preserve CF-HA languages
- restriction are necessary

Further Questions:

- case of inverse context-free and prefix rules $\ell \rightarrow f(ux)$, resp. inverse context-free and postfix rules $\ell \rightarrow f(xu)$
- preservation of sheave-automata or Presburger-automata (Seidl, Musholl...) by rewriting
- applications to access control.