

# Restrictions for Restarting Tree Automata

**Friedrich Otto** and **Heiko Stamer**

Universität Kassel  
Fachbereich Elektrotechnik/Informatik  
Wilhelmshöher Allee 71-73, 34109 Kassel, Germany  
stamer@theory.informatik.uni-kassel.de  
76F7 3011 329D 27DB 8D7C 3F97 4F58 4EB8 FB2B E14F

17. Theorietag der GI-Fachgruppe 0.1.5  
Universität Leipzig, 28.–29. September 2007

## 1 Restarting Tree Automata

## 2 Restrictions for Restarting Tree Automata

- Single-Path Restarting Tree Automaton
- Ground-Rewrite Restarting Tree Automaton

## 3 Conclusion

Roughly speaking, a *Restarting Tree Automaton* is an

iterated top-down tree automaton

equipped with an

height-bounded read/write-window

and the capability to

rewrite the input in a restricted way.

A *computation* consists of finitely many consecutive cycles of the form:

- 1 Process the input beginning at the root and fork down to the leaves using a *regular control* similar to a top-down finite tree automaton.
  - 2 On the way to the leaves, perform *at most one size-reducing rewrite* step in each separate branch.
  - 3 Depending on an additional regular control,
    - halt and reject the input, or
    - execute a *restart*, i.e., move the read/write-window to the top and start with the initial state (i.e., forget all collected information).
- } cycle

Repeat this cycle until a “simple form” of the input is obtained, which can be recognized in step 1 only.

# I. Single-Path Restriction

## Definition (Single-Path Top-Down Tree Automaton)

An **spNF $\downarrow$ T-automaton** is a five-tuple  $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, q_0, k, \Delta)$ , where

- $\mathcal{F}$  is a ranked alphabet,  $\mathcal{Q}$  is a finite set of states,  $q_0 \in \mathcal{Q}$  is the *initial state*,
- $k \geq 1$  is the height of the *look-ahead window*, and
- $\Delta$  is a finite term rewriting system (TRS) on  $\mathcal{F} \cup \mathcal{Q}$  that contains only
  - 1  $k$ -height bounded linear *top-down transitions* of the form

$$q(t) \rightarrow t[x_1, \dots, x_{i-1}, q'(x_i), x_{i+1}, \dots, x_n], \text{ where}$$

$n \geq 1$ ,  $t \in \text{Ctx}(\mathcal{F}, \mathcal{X}_n)$ ,  $\text{Hgt}(t) \leq k$ ,  $q, q' \in \mathcal{Q}$ ,  $i \in \{1, \dots, n\}$ , and

- 2  $k$ -height bounded *final transitions* of the form  $q(t) \rightarrow t$ , where  $t \in \mathcal{T}(\mathcal{F})$ ,  $\text{Hgt}(t) \leq k$ , and  $q \in \mathcal{Q}$ .

The recognized tree language is  $L(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid q_0(t) \rightarrow_{\Delta}^* t\}$ .

Obviously,  $\mathcal{L}(\text{FINT}) \subsetneq \mathcal{L}(\text{spNF}\downarrow\text{T}) \subsetneq \mathcal{L}(\text{RTG})$  holds,  
but  $\mathcal{L}(\text{spNF}\downarrow\text{T})$  is incomparable to  $\mathcal{L}(\text{DF}\downarrow\text{T})$  under set inclusion.

## Definition (Single-Path Restarting Tree Automaton)

An **spRRWWT-automaton** is a six-tuple  $\mathcal{A} = (\mathcal{F}, \mathcal{G}, \mathcal{Q}, q_0, k, \Delta)$ ,

- $\mathcal{F}$  is a ranked input alphabet,  $\mathcal{G} \supseteq \mathcal{F}$  is a ranked working alphabet,
- $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$  is a finite set of states such that  $\mathcal{Q}_1 \cap \mathcal{Q}_2 = \emptyset$ ,
- $q_0 \in \mathcal{Q}_1$  is the *initial state* and simultaneously the *restart state*,
- $k \geq 1$  is the height of the read/write-window, and
- $\Delta = \Delta_1 \cup \Delta_2$  is a finite term rewriting system (TRS) on  $\mathcal{G} \cup \mathcal{Q}$ ,

where  $\mathcal{A}_{S_{\mathcal{G}}} = (\mathcal{G}, \mathcal{Q}_1, q_0, k, \Delta_1)$  is an spNF $\downarrow$ T-automaton on  $\mathcal{G}$ .

$\rightarrow_{\Delta}$   $\rightarrow_{\Delta_1}$  denotes the *move relation* induced by the TRS  $\Delta$  resp.  $\Delta_1$

$\hookrightarrow_{\mathcal{A}}$  *cycle relation*, i.e.,  $u \hookrightarrow_{\mathcal{A}} v$  iff  $q_0(u)(\rightarrow_{\Delta}^+ \setminus \rightarrow_{\Delta_1}^+)v$ , for  $u, v \in \mathcal{T}(\mathcal{G})$

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \mid \exists \ell \geq 0, \exists t_1, \dots, t_{\ell} \in \mathcal{T}(\mathcal{G}) \text{ such that } \right. \\ \left. t_0 \hookrightarrow_{\mathcal{A}} t_1, \dots, t_{\ell-1} \hookrightarrow_{\mathcal{A}} t_{\ell}, \text{ and } t_{\ell} \in L(\mathcal{A}_{S_{\mathcal{G}}}) \right\}$$

$\Delta_1$  contains only the following **linear** transition rules:

- 1 **Top-down transitions:**  $q(t) \rightarrow t[x_1, \dots, x_{i-1}, q'(x_i), x_{i+1}, \dots, x_n]$   
 $(n \geq 1, t \in \text{Ctx}(\mathcal{G}, \mathcal{X}_n), 1 \leq \text{Hgt}(t) \leq k, q, q' \in \mathcal{Q}_1, i \in \{1, \dots, n\})$
- 2 **Final transitions:**  $q(t) \rightarrow t$   $(t \in \mathcal{T}(\mathcal{G}), 0 \leq \text{Hgt}(t) \leq k, q \in \mathcal{Q}_1)$

$\Delta_2$  contains only the following **linear** transition rules:

- 1 **Top-down transitions:**  $q(t) \rightarrow t[x_1, \dots, x_{i-1}, q'(x_i), x_{i+1}, \dots, x_n]$   
 $(n \geq 1, t \in \text{Ctx}(\mathcal{G}, \mathcal{X}_n), 1 \leq \text{Hgt}(t) \leq k, q, q' \in \mathcal{Q}_2, i \in \{1, \dots, n\})$
- 2 **Final transitions:**  $q(t) \rightarrow t$   $(t \in \mathcal{T}(\mathcal{G}), 0 \leq \text{Hgt}(t) \leq k, q \in \mathcal{Q}_2)$
- 3 **Size-reducing rewrite transitions:**  
 $q(t) \rightarrow t'[x_1, \dots, x_{i-1}, q'(x_i), x_{i+1}, \dots, x_n]$   
 $(n \geq 1, t \in \mathcal{T}(\mathcal{G}, \mathcal{X}_n), t' \in \text{Ctx}(\mathcal{G}, \mathcal{X}_n), \|t\| > \|t'\|, \text{Hgt}(t) \leq k,$   
 $q \in \mathcal{Q}_1, q' \in \mathcal{Q}_2, i \in \{1, \dots, n\})$
- 4 **Size-reducing final rewrite transitions:**  $q(t) \rightarrow t'$   
 $(t, t' \in \mathcal{T}(\mathcal{G}), \|t\| > \|t'\|, \text{Hgt}(t) \leq k, q \in \mathcal{Q}_1)$

Basic variants of (sp)RRWWT-automata, derived from the corresponding restarting automata on words:

**RR-prefix:** General case, no restriction.

**R-prefix:** Every *rewrite transition* contains **no** successor state(s) in the corresponding RHS, i.e.  $q(t) \rightarrow t'[x_1, \dots, x_n]$ .  
That means, after a rewrite step no additional regular control can be performed in the affected branches.

**WW-suffix:** General case, no restriction.

**W-suffix:** The working alphabet  $\mathcal{G}$  coincides with the input alphabet  $\mathcal{F}$ , i.e., there are no auxiliary symbols available.

**$\lambda$ -suffix:** The term  $t'$  of the RHS of every *rewrite transition* is **homeomorphically embedded** in the LHS.

Example ( $L_4 = \{f(g^n(h^n(a)), g^n(h^n(a))) \mid n \geq 1\} \notin \mathcal{L}(\text{CFTG})$ )

RT-automaton:  $\mathcal{F} = \mathcal{G} = \{f(\cdot, \cdot), g(\cdot), h(\cdot), a\}$  (no auxiliary symbols),  
 $k = 3$ ,  $Q = Q_1 \cup Q_2$ , where  $Q_1 = \{q_0, q_1\}$  and  $Q_2 = \emptyset$ ,

$$\Delta_1 = \left\{ \begin{array}{c} \begin{array}{c} q_0 \\ f \quad f \\ g \ g \rightarrow g \ g \\ h \ h \rightarrow h \ h \\ a \ a \rightarrow a \ a \end{array} , \begin{array}{c} \begin{array}{c} q_0 \\ f \\ g \ g \rightarrow q_1 \ q_1 \\ x_1 \ x_2 \end{array} , \begin{array}{c} \begin{array}{c} q_1 \\ g \\ g \rightarrow q_1 \\ x_1 \end{array} \end{array} \right\},$$

$$\Delta_2 = \left\{ \begin{array}{c} \begin{array}{c} q_1 \\ g \\ h \rightarrow h \\ h \rightarrow x_1 \\ x_1 \end{array} \end{array} \right\}.$$

Note that  $L_4 \in \mathcal{L}(\text{spRWWT})$ , but on the other hand  $L_4 \notin \mathcal{L}(\text{spRRWT})$ .

Example ( $L_1 = \{f(g^n(a), g^n(a)) \mid n \geq 0\} \in \mathcal{L}(\text{CFTG}) \setminus \mathcal{L}(\text{RTG})$ )

spRT-automaton:  $\mathcal{F} = \mathcal{G} = \{f(\cdot, \cdot), g(\cdot), a\}$  (no auxiliary symbols),  
 $k = 2$ ,  $Q = Q_1 \cup Q_2$ , where  $Q_1 = \{q_0\}$  and  $Q_2 = \emptyset$ ,

$$\Delta_1 = \left\{ \begin{array}{c} q_0 \\ f \rightarrow f \\ a \ a \quad a \ a \end{array} \right\}, \quad \Delta_2 = \left\{ \begin{array}{c} q_0 \\ f \\ g \ g \rightarrow f \\ x_1 \ x_2 \quad x_1 \ x_2 \end{array} \right\}.$$

## Proposition

Let  $X \in \{R, RR, RW, RRW, RWW, RRWW\}$  be a type of restarting automaton. Then for each spXT-automaton  $\mathcal{A}$  there exists an XT-automaton  $\mathcal{A}'$  such that  $L(\mathcal{A}) = L(\mathcal{A}')$ .

## Theorem

$$\mathcal{L}(\text{RTG}) \subsetneq \mathcal{L}(\text{spRT}).$$

## Proof (idea):

- Simulate a complete  $\text{DF}\uparrow\text{T}$ -automaton  $\mathcal{B} = (\mathcal{F}, Q^{(\mathcal{B})}, Q_f^{(\mathcal{B})}, \Delta^{(\mathcal{B})})$  by an spRT-automaton  $\mathcal{A} := (\mathcal{F}, \mathcal{F}, Q, q_0, k, \Delta)$  where  $k := |Q^{(\mathcal{B})}| + 1$
- $\mathcal{A}$  accepts all  $t \in L(\mathcal{B})$  satisfying  $\text{Hgt}(t) \leq k$  by final transitions
- Construct final rewrite transitions of the form  $q(t) \rightarrow t_1[t_3]$ , for all  $t \in \mathcal{T}(\mathcal{F})$ , where  $\text{Hgt}(t) = k$  and  $t = t_1[t_2[t_3]]$ , such that

$$t_3 \xrightarrow{*}_{\Delta^{(\mathcal{B})}} p \quad \text{and} \quad t_2[t_3] \xrightarrow{*}_{\Delta^{(\mathcal{B})}} p,$$

for some  $p \in Q^{(\mathcal{B})}$  (pumping argument, note that  $\mathcal{B}$  is deterministic)

## Theorem

Given a linear context-free tree grammar  $G$ , an spRWWT-automaton  $\mathcal{A}$  can be constructed such that  $L(G) = L(\mathcal{A})$ .

Proof (idea):

(Stamer and Otto, CAI 2007)

- Transform  $G$  into a normal form with only *growing* productions
- Simulate all derivations of  $G$  nondeterministically and in reverse order
- This simulation can be done with an spRWWT-automaton as well

## Corollary

$\mathcal{L}(\text{lin-CFTG}) \subsetneq \mathcal{L}(\text{spRWWT})$ .

## II. Ground-Rewrite Restriction

## Definition (Ground-Rewrite Restarting Tree Automaton)

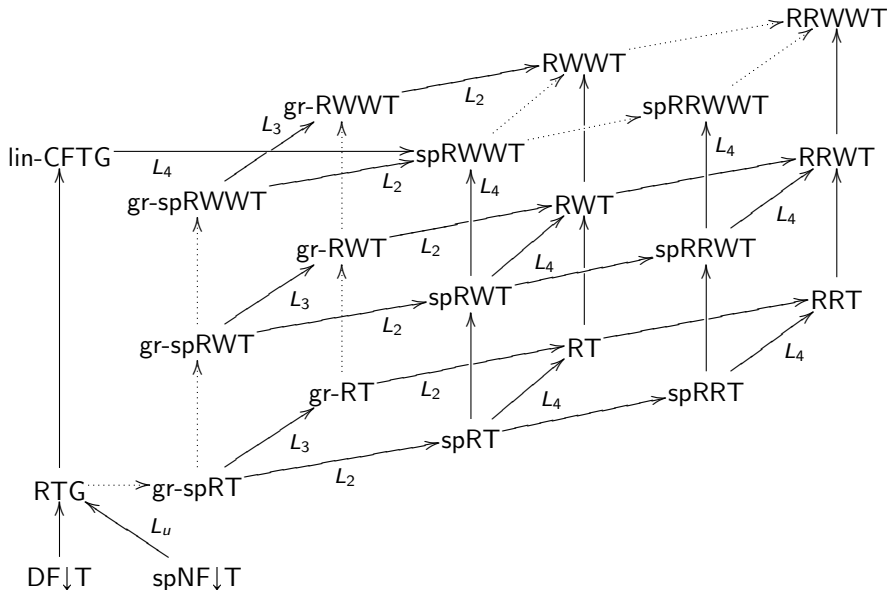
A gr-RWWT-automaton  $\mathcal{A} = (\mathcal{F}, \mathcal{G}, Q, q_0, k, \Delta)$  is essentially an RWWT-automaton for which  $\Delta_2$  only contains *final* rewrite transitions.

Example ( $L_1 = \{f(g^n(a), g^n(a)) \mid n \geq 0\} \in \mathcal{L}(\text{CFTG}) \setminus \mathcal{L}(\text{RTG})$ )

gr-RT-automaton:  $\mathcal{F} = \mathcal{G} = \{f(\cdot, \cdot), g(\cdot), a\}$ ,  $k = 1$ ,  $Q = Q_1 \cup Q_2$ ,  
 where  $Q_1 = \{q_0, q_1\}$  and  $Q_2 = \emptyset$ ,

$$\Delta_1 = \left\{ \begin{array}{c} \begin{array}{ccc} q_0 & & f \\ f & \rightarrow & q_1 \ q_1 \\ x_1 \ x_2 & & x_1 \ x_2 \end{array} , \quad \begin{array}{ccc} q_0 & & \\ f & \rightarrow & f \\ a \ a & & a \ a \end{array} , \quad \begin{array}{ccc} q_1 & & g \\ g & \rightarrow & q_1 \\ x_1 & & x_1 \end{array} \end{array} \right\},$$

$$\Delta_2 = \left\{ \begin{array}{c} q_1 \\ g \rightarrow a \\ a \end{array} \right\}.$$



## Open Questions:

- Does the equality  $\mathcal{L}(\text{RTG}) = \mathcal{L}(\text{gr-spRWT})$  hold?  
If so,  $\mathcal{L}(\text{RTG}) = \mathcal{L}(\text{gr-spRWWT})$  follows easily.
- Exists for each (gr-)spRWWT-automaton  $\mathcal{A}$  an equivalent (gr-)spRWWT-automaton  $\mathcal{B}$  such that  $L(\mathcal{B}_{S_G})$  is finite?

# Thank you! Questions?



Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard,  
Denis Lugiez, Sophie Tison, Marc Tommasi.  
Tree Automata Techniques and Applications.  
<http://www.grappa.univ-lille3.fr/tata/>



Heiko Stamer and Friedrich Otto.  
Restarting Tree Automata.  
Proceedings SOFSEM 2007, LNCS 4362, pp. 510–521, 2007.



Heiko Stamer and Friedrich Otto.  
Restarting Tree Automata and Linear Context-Free Tree Languages.  
Proceedings CAI 2007, Thessaloniki, May 2007.