Single-Path Restarting Tree Automata

Friedrich Otto and **Heiko Stamer**

Universität Kassel
Fachbereich Elektrotechnik/Informatik
Wilhelmshöher Allee 71-73, 34109 Kassel, Germany

otto@theory.informatik.uni-kassel.de

3rd International Conference on Algebraic Informatics,
Thessaloniki, May 19–22, 2009

Outline

Analyis by Reduction

Linguistic task: Analyze sentences of a natural language

- verify syntactic correctness
- determine dependencies
- disambiguate between morphological ambiguities

A linguistic technique: Analysis by reduction

Application: natural languages with a free word order
(Czech, Polish, Russian, Turkish, ...)

Restarting Automata on Strings

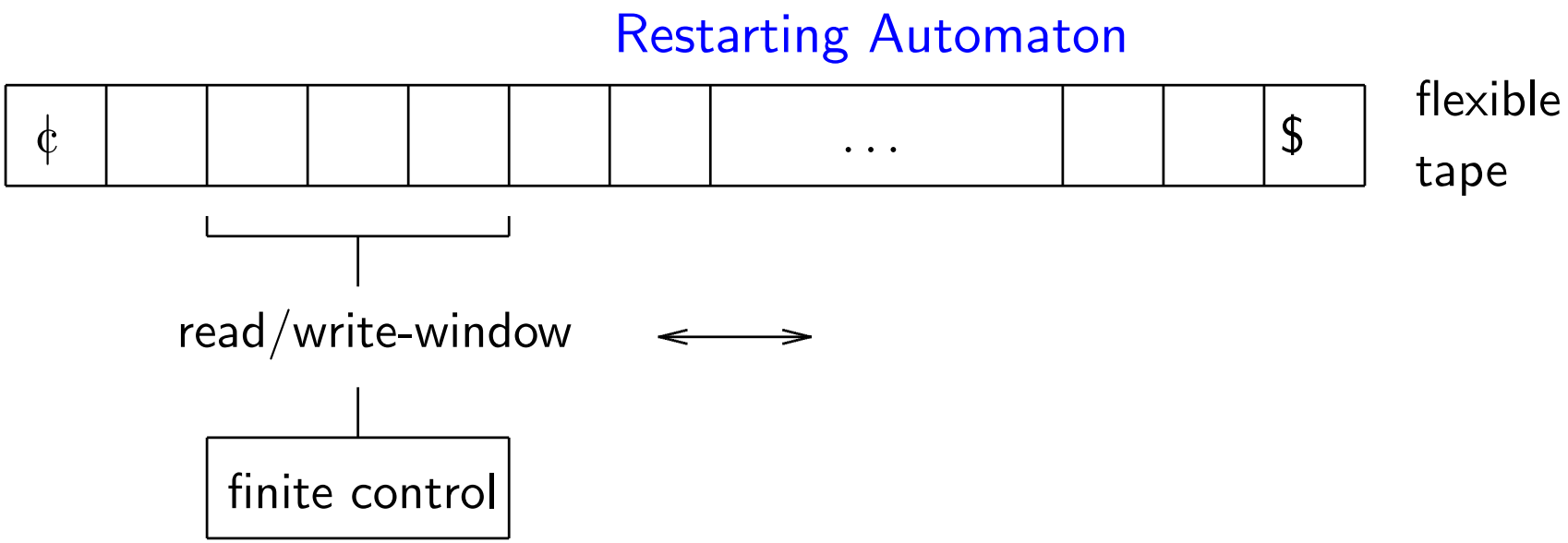
Operations used in the Analysis by Reduction:

Move-right: MVR

Delete/Rewrite: replace a small part of the sentence

Restart: start anew on a simplified sentence

Jančar, Mráz, Plátek, Vogel (1995 – 1999):



Restarting Tree Automata

Roughly speaking, a *Restarting Tree Automaton* is an

iterated top-down tree automaton

equipped with a

height-bounded read/write-window

and the capability to

rewrite the input in a restricted way.

Restarting Tree Automata

Roughly speaking, a *Restarting Tree Automaton* is an

iterated top-down tree automaton

equipped with a

height-bounded read/write-window

and the capability to

rewrite the input in a restricted way.

Restarting Tree Automata

Roughly speaking, a *Restarting Tree Automaton* is an

iterated top-down tree automaton

equipped with a

height-bounded read/write-window

and the capability to

rewrite the input in a restricted way.

Restarting Tree Automata

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 on 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 information collected).

} cycle

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

Definition (Restarting Tree Automaton)

An RRWWT-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 $\mathcal{G} \cup \mathcal{Q}$.

(Stateless) configuration: ground term from $\mathcal{T}(\mathcal{G} \cup \mathcal{Q})$ (resp. from $\mathcal{T}(\mathcal{G})$)

$\rightarrow_\Delta$ ($\rightarrow_{\Delta_1}$) denotes the *move relation* induced by the TRS Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } q_0(t_\ell) \rightarrow_{\Delta_1}^+ t_\ell \end{array} \right\}$$

Definition (Restarting Tree Automaton)

An RRWWT-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 $\mathcal{G} \cup \mathcal{Q}$.

(Stateless) configuration: ground term from $\mathcal{T}(\mathcal{G} \cup \mathcal{Q})$ (resp. from $\mathcal{T}(\mathcal{G})$)

$\rightarrow_\Delta$ $(\rightarrow_{\Delta_1})$ denotes the *move relation* induced by the TRS Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } q_0(t_\ell) \rightarrow_{\Delta_1}^+ t_\ell \end{array} \right\}$$

Definition (Restarting Tree Automaton)

An RRWWT-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 $\mathcal{G} \cup \mathcal{Q}$.

(Stateless) configuration: ground term from $\mathcal{T}(\mathcal{G} \cup \mathcal{Q})$ (resp. from $\mathcal{T}(\mathcal{G})$)

$\rightarrow_\Delta$ $(\rightarrow_{\Delta_1})$ denotes the *move relation* induced by the TRS Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } q_0(t_\ell) \rightarrow_{\Delta_1}^+ t_\ell \end{array} \right\}$$

Form of the Rules from Δ_1 and Δ_2

Δ_1 contains only the following *linear* transition rules:

1. Top-down transitions: $q(t) \to t[q_1(x_1), \ldots, q_n(x_n)]$
$(n \geq 1,\ t \in \mathrm{Ctx}(\mathcal{G}, \mathcal{X}_n),\ 1 \leq \mathrm{Hgt}(t) \leq k,\ q, q_1, \ldots, q_n \in \mathcal{Q}_1)$
2. Final transitions: $q(t) \to t$ $(t \in \mathcal{T}(\mathcal{G}),\ 0 \leq \mathrm{Hgt}(t) \leq k,\ q \in \mathcal{Q}_1)$

Δ_2 contains only the following *linear* transition rules:

1. *Top-down transitions:* $q(t) \to t[q_1(x_1), \ldots, q_n(x_n)]$
$(n \geq 1,\ t \in \mathrm{Ctx}(\mathcal{G}, \mathcal{X}_n),\ 1 \leq \mathrm{Hgt}(t) \leq k,\ q, q_1, \ldots, q_n \in \mathcal{Q}_2)$
2. *Final transitions:* $q(t) \to t$ $(t \in \mathcal{T}(\mathcal{G}),\ 0 \leq \mathrm{Hgt}(t) \leq k,\ q \in \mathcal{Q}_2)$
3. *Size-reducing rewrite transitions:* $q(t) \to t'[q_1(x_1), \ldots, q_n(x_n)]$
$(n \geq 1,\ t, t' \in \mathrm{Ctx}(\mathcal{G}, \mathcal{X}_n),\ ||t|| > ||t'||,\ \mathrm{Hgt}(t) \leq k,$
$q \in \mathcal{Q}_1,\ q_1, \ldots, q_n \in \mathcal{Q}_2)$
4. *Size-reducing final rewrite transitions:* $q(t) \to t'$
$(t, t' \in \mathcal{T}(\mathcal{G}),\ ||t|| > ||t'||,\ \mathrm{Hgt}(t) \leq k,\ q \in \mathcal{Q}_1)$

Results

Theorem (Stamer, Otto, SOFSEM 2007)

The non-context-free tree language

$$L_4 = \{ f(g^i(h^i(a)), g^i(h^i(a))) \mid i \geq 1 \}$$

is accepted by an RT*-automaton.*

Theorem (Stamer, Otto, CAI 2007)

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

Single-Path Restriction

I. Single-Path Restriction

Single-Path Top-Down Tree Automaton

Definition (Single-Path Top-Down Tree Automaton)

An spNF↓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
- Δ 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

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

  $n \geq 1$, $t \in \mathrm{Ctx}(\mathcal{F}, \mathcal{X}_n)$, $\mathrm{Hgt}(t) \leq k$, $q, q' \in \mathcal{Q}$, $i \in \{1, \ldots, n\}$, and
  2. k-height bounded *final transitions* of the form $q(t) \to t$, where $t \in \mathcal{T}(\mathcal{F})$, $\mathrm{Hgt}(t) \leq k$, and $q \in \mathcal{Q}$.

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

Single-Path Top-Down Tree Automaton

$\mathscr{L}(\mathrm{FINT}) \subsetneq \mathscr{L}(\mathrm{spNF}{\downarrow}\mathrm{T}) \subsetneq \mathscr{L}(\mathrm{RTG})$ holds,

but $\mathscr{L}(\mathrm{spNF}{\downarrow}\mathrm{T})$ is incomparable to $\mathscr{L}(\mathrm{DF}{\downarrow}\mathrm{T})$ under set inclusion,

as the language

$$L_d := \{\, f(g^i(a), g^j(b)) \mid i, j \geq 0 \,\} \in \mathscr{L}(\mathrm{DF}{\downarrow}\mathrm{T})$$

is not recognized by any $\mathrm{spNF}{\downarrow}\mathrm{T}$.

Single-Path Restarting Tree Automaton

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 Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } t_\ell \in L(\mathcal{A}_{S_\mathcal{G}}) \end{array} \right\}$$

Single-Path Restarting Tree Automaton

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 Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } t_\ell \in L(\mathcal{A}_{S_\mathcal{G}}) \end{array} \right\}$$

Single-Path Restarting Tree Automaton

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 Δ (resp. Δ_1)

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

$$L(\mathcal{A}) = \left\{ t_0 \in \mathcal{T}(\mathcal{F}) \;\middle|\; \begin{array}{l} \exists \ell \geq 0, \exists t_1, \ldots, t_\ell \in \mathcal{T}(\mathcal{G}) \text{ such that} \\ t_0 \hookrightarrow_\mathcal{A} t_1, \ldots, t_{\ell-1} \hookrightarrow_\mathcal{A} t_\ell, \text{ and } t_\ell \in L(\mathcal{A}_{S_\mathcal{G}}) \end{array} \right\}$$

Form of the Rules from Δ_1 and Δ_2

Δ_1 contains only the following linear transition rules:

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

Δ_2 contains only the following linear transition rules:

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

Form of the Rules from Δ_1 and Δ_2

Δ_1 contains only the following linear transition rules:

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

Δ_2 contains only the following linear transition rules:

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

Form of the Rules from Δ_1 and Δ_2

Δ_1 contains only the following linear transition rules:

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

Δ_2 contains only the following linear transition rules:

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

Basic Variants of (sp)RRWWT-Automata

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. $q(t) \to t'[x_1, \ldots, x_n]$.
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.

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

Expressive Power

Example ($L_4 = \{f(g^i(h^i(a)), g^i(h^i(a))) \mid i \geq 1\} \notin \mathscr{L}(\mathrm{CFTG})$)

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 $Q_2 = \emptyset$,

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

$$\Delta_2 = \left\{ \begin{array}{ccc} q_1 & & \\ g & & \\ h & \to & h \\ h & & x_1 \\ x_1 & & \end{array} \right\}.$$

Expressive Power

Example ($L_1 = \{f(g^i(a), g^i(a)) \mid i \geq 0\} \in \mathscr{L}(\mathrm{CFTG}) \setminus \mathscr{L}(\mathrm{RTG})$)

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 $Q_2 = \emptyset$,

$$\Delta_1 = \left\{ \begin{array}{ccc} q_0 & & \\ f & \to & f \\ a\ \ a & & a\ \ a \end{array} \right\},$$

$$\Delta_2 = \left\{ \begin{array}{ccc} q_0 & & \\ f & & f \\ g\ \ g & \to & x_1\ \ x_2 \\ x_1\ \ x_2 & & \end{array} \right\}.$$

Proposition

Let $X \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}, \mathrm{RWW}, \mathrm{RRWW}\}$ be a type of restarting automaton. Then for each sp$X$T-automaton $\mathcal{A}$ there exists an XT-automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.

Expressive Power

Example ($L_1 = \{f(g^i(a), g^i(a)) \mid i \geq 0\} \in \mathscr{L}(\mathrm{CFTG}) \setminus \mathscr{L}(\mathrm{RTG})$)

spRT-automaton: $\mathcal{F} = \mathcal{G} = \{f(\cdot,\cdot), g(\cdot), a\}$ (no auxiliary symbols), $k = 2$, $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, where $\mathcal{Q}_1 = \{ q_0 \}$ and $\mathcal{Q}_2 = \emptyset$,

$$\Delta_1 = \left\{ \begin{array}{ccc} q_0 & & \\ f & \to & f \\ a\ \ a & & a\ \ a \end{array} \right\},$$

$$\Delta_2 = \left\{ \begin{array}{ccc} q_0 & & \\ f & & f \\ g\ \ g & \to & x_1\ \ x_2 \\ x_1\ \ x_2 & & \end{array} \right\}.$$

Proposition

Let $X \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}, \mathrm{RWW}, \mathrm{RRWW}\}$ be a type of restarting automaton. Then for each sp$X$T*-automaton $\mathcal{A}$ there exists an XT-automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Regular Tree Languages

Theorem

$\mathscr{L}(\mathrm{RTG}) \subsetneq \mathscr{L}(\mathrm{spRT})$.

Proof (idea):

- Simulate a complete DF↑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 $\mathrm{Hgt}(t) \leq k$ by final transitions
- Construct final rewrite transitions of the form $q(t) \to t_1[t_3]$, for all $t \in \mathcal{T}(\mathcal{F})$, where $\mathrm{Hgt}(t) = k$ and $t = t_1[t_2[t_3]]$, such that

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

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

Linear Context-Free Tree Languages

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

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

Ground-Rewrite Restriction

II. Ground-Rewrite Restriction

Ground-Rewrite Restarting Tree Automaton

Definition (Ground-Rewrite Restarting Tree Automaton)

A gr-RWWT-automaton $\mathcal{A} = (\mathcal{F}, \mathcal{G}, \mathcal{Q}, q_0, k, \Delta)$ is essentially an RWWT-automaton for which Δ_2 only contains *final* rewrite transitions.

Example ($L_1 = \{f(g^i(a), g^i(a)) \mid i \geq 0\} \in \mathscr{L}(\mathrm{CFTG}) \setminus \mathscr{L}(\mathrm{RTG})$)

gr-RT-automaton: $\mathcal{F} = \mathcal{G} = \{f(\cdot,\cdot), g(\cdot), a\}$, $k = 1$, $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, where $\mathcal{Q}_1 = \{ q_0 , q_1 \}$ and $\mathcal{Q}_2 = \emptyset$,

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

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

Single-Path Versus Ground Rewrite Restriction

Let $L_2 = \{ g^i(h(g^i(a))) \mid i \geq 0 \}$.

Proposition

$L_2 \in \mathscr{L}(\text{spRT}) \setminus \mathscr{L}(\text{gr-RWWT})$.

Let $L_3 = \{ f(g^i(a), g^i(a)), f(g^i(a), g^{2i}(b)) \mid i \geq 0 \}$.

Proposition

$L_3 \in \mathscr{L}(\text{gr-RT}) \setminus \mathscr{L}(\text{spRWT})$.

Single-Path Versus Ground Rewrite Restriction

Let $L_2 = \{ g^i(h(g^i(a))) \mid i \geq 0 \}$.

Proposition

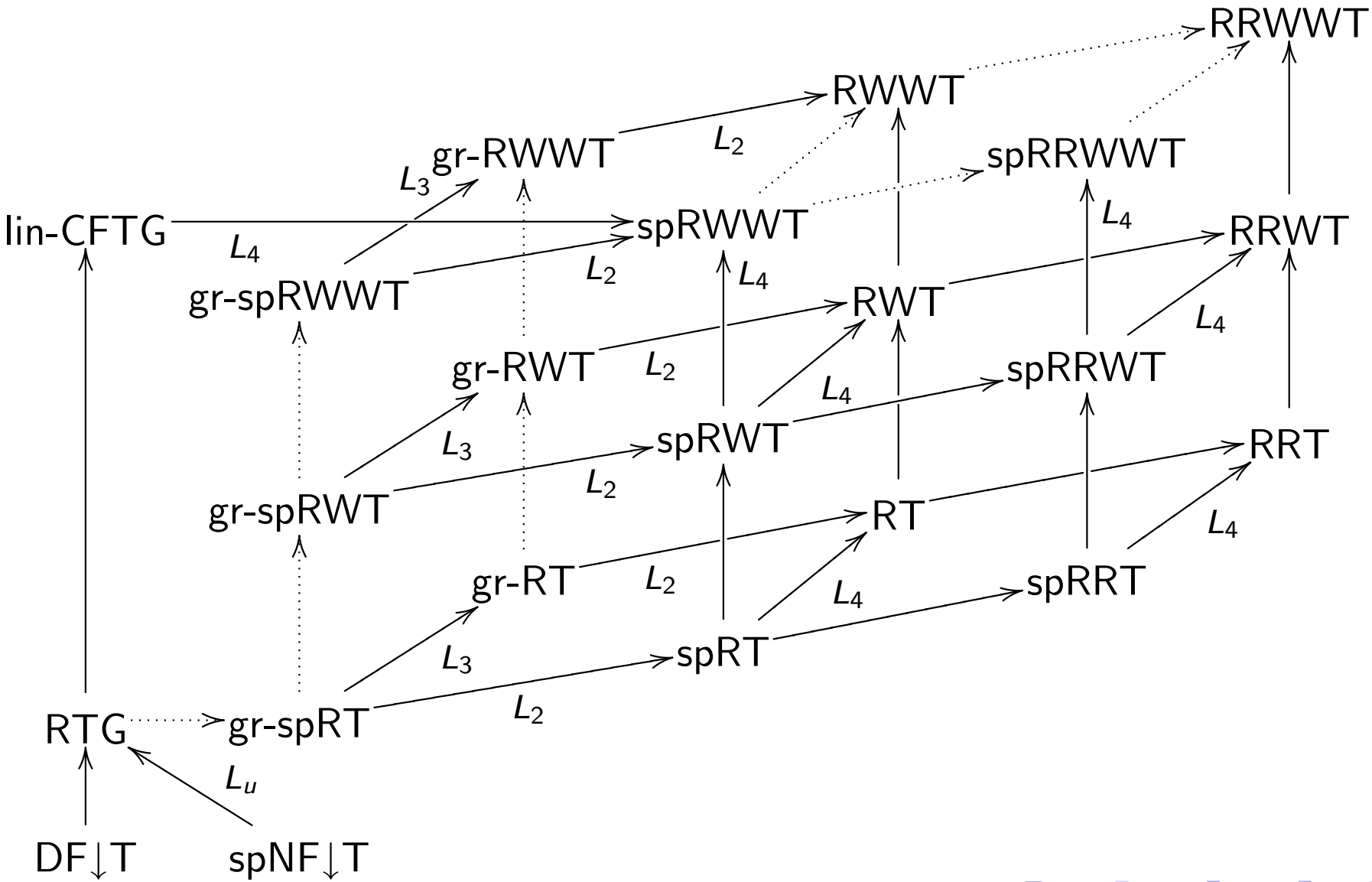
$L_2 \in \mathscr{L}(\text{spRT}) \setminus \mathscr{L}(\text{gr-RWWT})$.

Let $L_3 = \{ f(g^i(a), g^i(a)), f(g^i(a), g^{2i}(b)) \mid i \geq 0 \}$.

Proposition

$L_3 \in \mathscr{L}(\text{gr-RT}) \setminus \mathscr{L}(\text{spRWT})$.

Taxonomy of Restarting Tree Automata



Conclusion

Open Questions:

- Is any of the following inclusions proper:

$$\mathscr{L}(\text{gr-RT}) \subseteq \mathscr{L}(\text{gr-RWT}) \subseteq \mathscr{L}(\text{gr-RWWT})?$$

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

Thank you! Questions?

Conclusion

Open Questions:

- Is any of the following inclusions proper:

$$\mathscr{L}(\text{gr-RT}) \subseteq \mathscr{L}(\text{gr-RWT}) \subseteq \mathscr{L}(\text{gr-RWWT})?$$

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

Thank you! Questions?

References


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.
Restarting Tree Automata - Formal Properties and Possible Variations.
Doctoral Dissertation, Fachbereich Elektrotechnik/Informatik, Universität Kassel, September 2008.


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, LNCS 4728, pp. 275–289, 2007.