Hierarchical Relaxations of the Correctness Preserving Property for Restarting Automata[1]

František Mráz[1] Friedrich Otto[2] Martin Plátek[1]

[1] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

[2] Fachbereich Elektrotechnik/Informatik, Universität Kassel, Kassel, Germany

MCU 2007

[1] F. Mráz and M. Plátek were partially supported by the program 'Information Society' under project 1ET100300517. F. Mráz was also partially supported by the Grant Agency of Charles University in Prague under Grant-No. 358/2006/A-INF/MFF.

A linguistic motivation

(Czech, Russian, German) sentence analysis - - Prague linguistic group (Sgall, Hajičová, Panevová) , Melčuk, Kunze.

This method is different from the Chomskian type of sentence analysis in an essential way.
It is complex and has two basic phases.

- morphological disambiguation, lexico-semantic disambiguation, ... } additional information is inserted into the input sentence – auxiliary symbols
- analysis by reduction - correctness preserving simplifications of a fully disambigued sentence

Formal models

- model for the sentence analysis – restarting automaton
- model for the analysis by reduction – correctness preserving restarting automaton

Outline

1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
   - Basic properties of restarting automata
2. Relaxations of the Correctness Preserving Property
   - Cyclic relaxation and error relaxation of the Correctness Preserving Property
3. Results
   - The Hierarchy
   - Time-complexity results
   - Technicalities
4. Conclusions

Outline

1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
   - Basic properties of restarting automata
2. Relaxations of the Correctness Preserving Property
   - Cyclic relaxation and error relaxation of the Correctness Preserving Property
3. Results
   - The Hierarchy
   - Time-complexity results
   - Technicalities
4. Conclusions

Outline

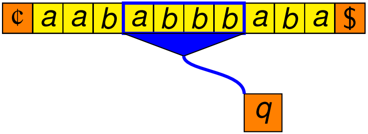
1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
   - Basic properties of restarting automata
2. Relaxations of the Correctness Preserving Property
   - Cyclic relaxation and error relaxation of the Correctness Preserving Property
3. Results
   - The Hierarchy
   - Time-complexity results
   - Technicalities
4. Conclusions

Outline

1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
   - Basic properties of restarting automata
2. Relaxations of the Correctness Preserving Property
   - Cyclic relaxation and error relaxation of the Correctness Preserving Property
3. Results
   - The Hierarchy
   - Time-complexity results
   - Technicalities
4. Conclusions

Restarting automaton (RLWW - automaton)

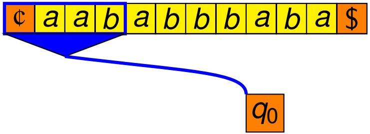
basic model



- restarting automaton consists of:
  - finite control
  - elastic working tape with sentinels
  - read/write window of fixed size
  - operations: move right, move left, rewrite, accept, restart

RLWW - automaton

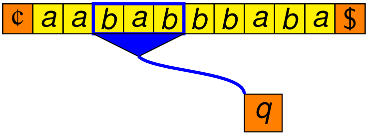
computation, start, restart, rewrite



- computation
  - starts on the left end in the starting state, the same situation after a restart
  - between two (re)starts exactly one rewriting of the content of its window must occur (local change), it must shorten the tape

RLWW-automaton

denotations



$M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- Γ is a finite tape alphabet, $\Sigma \subseteq \Gamma$,
- $¢, \$$ are sentinels, $\{¢, \$\} \cap \Gamma = \emptyset$
- $q_0 \in Q$ is the initial state
- δ is the transition relation = a finite set of instructions.

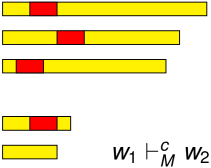
RLWW-automaton as a reduction system

Cycles, tails, reductions

a cycle: any part of a computation starting from a (re)starting configuration and ending by the next restart

a tail: the part of a computation after the last restart

$w_1 =$
$w_2 =$
$w_3 =$
...
$w_{n-1} =$
$w_n =$



$w_1 \vdash_M^c w_2 \vdash_M^c \cdots \vdash_M^c w_{n-1} \vdash_M^c w_n$

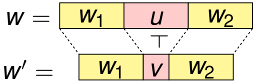
Note: if w_n is accepted by M then all $w_1, \ldots, w_n$ are accepted by M.

Meta-instructions

A more convenient representation

$(E_1, u \to v, E_2)$ a rewriting meta-instruction,

- $E_1, E_2 \subseteq \Gamma^*$ are regular languages called constraints
- $u, v \in \Gamma^*$ such that $|u| > |v|$,
- if $w = w_1 u w_2$, where $¢ \cdot w_1 \in E_1$, $w_2 \cdot \$ \in E_2$ then



(E, Accept) an accepting meta-instruction

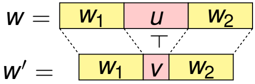
- $E \subseteq \Gamma^*$ is a regular language

Meta-instructions

A more convenient representation

$(E_1, u \to v, E_2)$ a rewriting meta-instruction,

- $E_1, E_2 \subseteq \Gamma^*$ are regular languages called constraints
- $u, v \in \Gamma^*$ such that $|u| > |v|$,
- if $w = w_1 u w_2$, where $¢ \cdot w_1 \in E_1$, $w_2 \cdot \$ \in E_2$ then

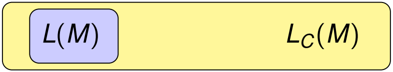


(E, Accept) an accepting meta-instruction

- $E \subseteq \Gamma^*$ is a regular language

Languages defined by a RLWW-automaton

- A word w is accepted by M if there exists a computation which starts by the (re)starting configuration $q_0¢w\$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the complete (characteristic) language accepted by the RLWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M



Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a,b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a,b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a+b)^*, aCa \to C, (a+b)^* \cdot \$)$
3. $(¢ \cdot (a+b)^*, bCb \to C, (a+b)^* \cdot \$)$
4. $(¢ \cdot (a+b)^*, aa \to C, (a+b)^* \cdot \$)$
5. $(¢ \cdot (a+b)^*, bb \to C, (a+b)^* \cdot \$)$

abbbbbba
abbCbba
abCba
aCa
C
Accept

Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a, b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a, b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a+b)^*, aCa \to C, (a+b)^* \cdot \$)$
3. $(¢ \cdot (a+b)^*, bCb \to C, (a+b)^* \cdot \$)$
4. $(¢ \cdot (a+b)^*, aa \to C, (a+b)^* \cdot \$)$
5. $(¢ \cdot (a+b)^*, bb \to C, (a+b)^* \cdot \$)$

abbbbbba
abbCbba
abCba
aCa
C
Accept

Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a, b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a, b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a + b)^*, aCa \to C, (a + b)^* \cdot \$)$
3. $(¢ \cdot (a + b)^*, bCb \to C, (a + b)^* \cdot \$)$
4. $(¢ \cdot (a + b)^*, aa \to C, (a + b)^* \cdot \$)$
5. $(¢ \cdot (a + b)^*, bb \to C, (a + b)^* \cdot \$)$

abbbbbba
abbCbba
abCba
aCa
C
Accept

Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a, b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a, b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a+b)^*, aCa \to C, (a+b)^* \cdot \$)$
3. $(¢ \cdot (a+b)^*, bCb \to C, (a+b)^* \cdot \$)$
4. $(¢ \cdot (a+b)^*, aa \to C, (a+b)^* \cdot \$)$
5. $(¢ \cdot (a+b)^*, bb \to C, (a+b)^* \cdot \$)$

abbbbbba
abbCbba
abCba
aCa
C
Accept

Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a,b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a,b\}^*\}$.

1. $(\text{¢} \cdot C \cdot \$, \text{Accept})$
2. $(\text{¢} \cdot (a+b)^*, aCa \to C, (a+b)^* \cdot \$)$
3. $(\text{¢} \cdot (a+b)^*, bCb \to C, (a+b)^* \cdot \$)$
4. $(\text{¢} \cdot (a+b)^*, aa \to C, (a+b)^* \cdot \$)$
5. $(\text{¢} \cdot (a+b)^*, bb \to C, (a+b)^* \cdot \$)$

abbbbbba
abbCbba
abCba
aCa
C
Accept

Sample restarting automaton M_s

M_s, with input alphabet $\Sigma = \{a, b\}$, one auxiliary symbol C ($\Gamma = \Sigma \cup \{C\}$) and the following meta-instructions:

- accepts the input language $L(M_s) = \{ww^R \mid w \in \{a, b\}^*\}$,
- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a, b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a + b)^*, aCa \to C, (a + b)^* \cdot \$)$
3. $(¢ \cdot (a + b)^*, bCb \to C, (a + b)^* \cdot \$)$
4. $(¢ \cdot (a + b)^*, aa \to C, (a + b)^* \cdot \$)$
5. $(¢ \cdot (a + b)^*, bb \to C, (a + b)^* \cdot \$)$

$abbbbbba$
$abbCbba$
$abCba$
aCa
C
Accept

A mistake by M_s

- the complete language of M_s is $L_C(M_s) = \{ww^R, wCw^R \mid w \in \{a,b\}^*\}$.

1. $(¢ \cdot C \cdot \$, \text{Accept})$
2. $(¢ \cdot (a+b)^*, aCa \to C, (a+b)^* \cdot \$)$
3. $(¢ \cdot (a+b)^*, bCb \to C, (a+b)^* \cdot \$)$
4. $(¢ \cdot (a+b)^*, aa \to C, (a+b)^* \cdot \$)$
5. $(¢ \cdot (a+b)^*, bb \to C, (a+b)^* \cdot \$)$

abbbbbba
abbbbCa

Important notion. A mistake by M - M reduces in a cycle a word from $L_C(M)$ to a word not belonging to $L_C(M)$.

M_s can make in the first cycle many different mistakes

Basic properties

Definition

(Correctness Preserving Property (no mistakes))
An RLWW-automaton M is *correctness preserving* if $u \in L_C(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L_C(M)$.

Definition

(Error Preserving Property)
An RLWW-automaton M is *error preserving* if $u \notin L_C(M)$ and $u \vdash_M^{c^*} v$ imply that $v \notin L_C(M)$.

Basic facts

- Each RLWW-automaton is error preserving.
- All deterministic RLWW-automata are correctness preserving.
- There are nondeterministic RLWW-automata that are not correctness preserving.

Cyclic relaxation and error relaxation

(Informal) definition. Let i be a non-negative integer. A RLWW-automaton M has *cyclic relaxation* of degree i, if M cannot make a mistake after the first i cycles (on a word w from $L_C(M)$).

(Informal) definition. Let M be an RLWW-automaton, and let j be a non-negative integer. M has *error relaxation* of degree j, if for the first cycle on a word from $L_C(M)$, there are at most j different mistakes that M can possibly make.

Notation

c(i)-RLWW – the class of RLWW-automata with cyclic relaxation of degree i,
e(j)-RLWW – the class of RLWW-automata with error relaxation of degree j,
ce(i, j)-RLWW – the class of RLWW-automata that simultaneously have cyclic relaxation of degree i and error relaxation of degree j.

Here we denote the corresponding classes of complete languages simply by –

c(i), e(j), ce(i, j).

Relaxations

Complete correctness preserving RLWW-languages

= c(0) = e(0) = ce(0, 0)

= complete deterministic RLWW-languages

[Messerchmidt,Otto 2007]

The Complete Hierarchy

$$
\begin{array}{ccccccccccc}
\mathrm{ce}(1,1) & \longrightarrow & \mathrm{ce}(2,1) & \longrightarrow & \mathrm{ce}(3,1) & \longrightarrow & \cdots & \longrightarrow & \bigcup_{i\geq 0}\mathrm{ce}(i,1) & \longrightarrow & \mathrm{e}(1) \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\mathrm{ce}(1,2) & \longrightarrow & \mathrm{ce}(2,2) & \longrightarrow & \mathrm{ce}(3,2) & \longrightarrow & \cdots & \longrightarrow & \bigcup_{i\geq 0}\mathrm{ce}(i,2) & \longrightarrow & \mathrm{e}(2) \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\mathrm{ce}(1,3) & \longrightarrow & \mathrm{ce}(2,3) & \longrightarrow & \mathrm{ce}(3,3) & \longrightarrow & \cdots & \longrightarrow & \bigcup_{i\geq 0}\mathrm{ce}(i,3) & \longrightarrow & \mathrm{e}(3) \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\bigcup_{j\geq 0}\mathrm{ce}(1,j) & \longrightarrow & \bigcup_{j\geq 0}\mathrm{ce}(2,j) & \longrightarrow & \bigcup_{j\geq 0}\mathrm{ce}(3,j) & \longrightarrow & \cdots & \longrightarrow & \bigcup_{i,j\geq 0}\mathrm{ce}(i,j) & \longrightarrow & \bigcup_{j\geq 0}\mathrm{e}(j) \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\mathrm{c}(1) & \longrightarrow & \mathrm{c}(2) & \longrightarrow & \mathrm{c}(3) & \longrightarrow & \cdots & \longrightarrow & \bigcup_{i\geq 0}\mathrm{c}(i) & \longrightarrow & \mathcal{L}_C(\mathrm{RLWW})
\end{array}
$$

Time-complexity results

Theorem

If M is a ce(i, j)-RLWW*-automaton, then the membership problems for the languages $L_C(M)$ and $L(M)$ are solvable in time $O((j+1)^i \cdot n^2)$.*

Theorem

If M is a c(i)-RLWW*-automaton for some $i \geq 0$, then the membership problems for the languages $L_C(M)$ and $L(M)$ are solvable in time $O(n^{i+2})$.*

Technicalities

- $L_{1,1} := \{ a^n b^n, a^n c b^n \mid n \geq 1 \} \cup \{ a^n b^{2n}, a^n d b^{2n} \mid n \geq 1 \}$.

Theorem

$L_{1,1} \notin ce(0,0)$ $\quad$ $L_{1,1} \in ce(1,1)$

- $L_{2,1} := L_{1,1} \cdot L_{1,1}$.

Theorem

$L_{2,1} \in ce(2,1)$ $\quad$ $L_{2,1} \notin ce(1,1)$.

- $L_{+,1} := L_{1,1}^{+}$.

Theorem

For all $j \geq 1$, $\bigcup_{i \geq 0} \mathrm{ce}(i,j) \subset \mathrm{e}(j)$, $\quad$ $\bigcup_{i \geq 0} \mathrm{c}(i) \subset \mathcal{L}_C(\mathrm{RLWW})$.

Important separations

Theorem

$L_{s1} \in \mathrm{c}(1) \smallsetminus \bigcup_{j\geq 0} \mathrm{e}(j)$.

- $L_{s2} := \{ ww^R dw_1 w_1^R, wcw^R dw_1 w_1^R, wcw^R dw_1 cw_1^R \mid w, w_1 \in \{a, b\}^* \}$ is accepted by an RLWW-automaton with cyclic relaxation of degree 2

Theorem

$L_{s2} \in \mathrm{c}(2) \smallsetminus (\mathrm{c}(1) \cup \bigcup_{j\geq 0} \mathrm{e}(j))$.

Conclusions

- I believe that we are able to show a similar hierarchy for the input languages.
- I believe that in the close future we will use the cyclic and error relaxations in order to characterize the degree of non-determinism of CFL. We will also consider the non-constant boundaries for the relaxations above.
- The real model for the sentence analysis is a bit more complex. It allows more than one rewriting in a cycle and it uses a more complex partition of the working alphabet (vocabulary) into so called 'levels'.