Lexicalized Syntactic Analysis by Restarting Automata

František Mráz[a] Friedrich Otto[a]
Dana Pardubská[b1] Martin Plátek[a2]

[a]Charles University, Prague, Czech Republic

[b]Comenius University, Bratislava, Slovakia

PSC 2019, Prague, August 27, 2019

[1]The research was partially supported by VEGA 2/0165/16.

[2]The research was partially supported by the grant of the Czech Science Foundation No. 19-05704S during the author's stay at Institute of Computer Science, Czech Academy of Sciences.

Contents

Lexicalized Syntactic Analysis

1. input sentence

PSC means an interesting conference

Lexicalized Syntactic Analysis

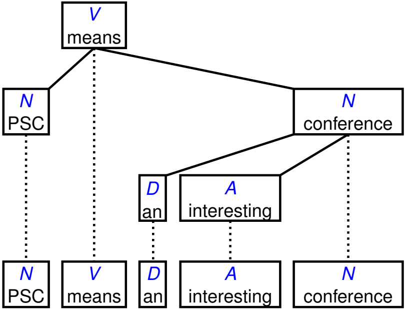
1. input sentence
2. lexicalization

N	*V*	*D*	*A*	*N*
PSC	means	an	interesting	conference

Lexicalized Syntactic Analysis

1. input sentence
2. lexicalization
3. lexicalized syntactic analysis – “Does the tagged word forms constitute a grammatically correct sentence which is correctly tagged?”

A Dependency tree
(dependency grammar)



Analysis by Reduction

The German team won the World Cup in Brazil.

Analysis by Reduction

The German team won the World Cup in Brazil.

The team won the World Cup in Brazil.

← each reduction

- preserves (non)correctness
- is local
- shortens

Analysis by Reduction

The German team won the World Cup in Brazil.

The team won the World Cup in Brazil.

The team won the Cup in Brazil.

← each reduction

- preserves (non)correctness
- is local
- shortens

Analysis by Reduction

The German team won the World Cup in Brazil.

The team won the World Cup in Brazil.

The team won the Cup in Brazil.

The team won in Brazil.

$\leftarrow$ each reduction

- preserves (non)correctness
- is local
- shortens

Analysis by Reduction

The German team won the World Cup in Brazil.

The team won the World Cup in Brazil.

The team won the Cup in Brazil.

The team won in Brazil.

The team won.

← each reduction

- preserves (non)correctness
- is local
- shortens

Analysis by Reduction

The German team won the World Cup in Brazil. ← each reduction

The team won the World Cup in Brazil.

The team won the Cup in Brazil.

The team won in Brazil.

- preserves (non)correctness
- is local
- shortens

The team won. ← simple correct sentence, hence Accept

Analysis by reduction

- checking correctness of sentences

Analysis by Reduction

The German team won the World Cup in Brazil.

The team won the World Cup in Brazil.

The team won the Cup in Brazil.

The team won in Brazil.

The team won.

← each reduction

- preserves (non)correctness
- is local
- shortens

← simple correct sentence, hence Accept

Analysis by reduction

- checking correctness of sentences
- localizing errors

Analysis by Reduction

The German team won the World Cup in Brazil. ← each reduction

The team won the World Cup in Brazil.

- preserves (non)correctness

The team won the Cup in Brazil.

- is local

The team won in Brazil.

- shortens

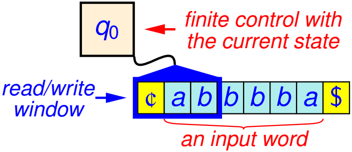
The team won. ← simple correct sentence, hence Accept

Analysis by reduction

- checking correctness of sentences
- localizing errors
- detecting (in)dependencies within a sentence

Restarting Automaton

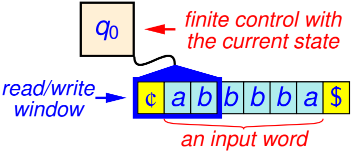
RLWW-automaton



- a finite set of *states* Q

Restarting Automaton

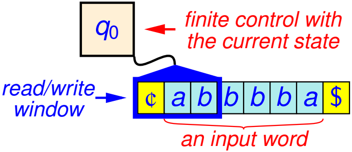
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ

Restarting Automaton

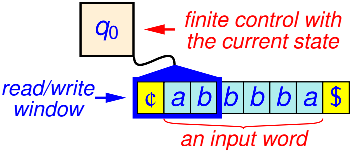
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ
- a *working alphabet* $\Gamma (\supseteq \Sigma)$

Restarting Automaton

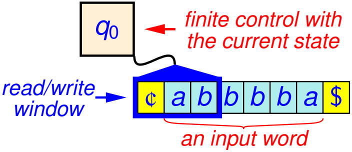
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ
- a *working alphabet* $\Gamma(\supseteq \Sigma)$
- the left and right *sentinels* ¢ and \$

Restarting Automaton

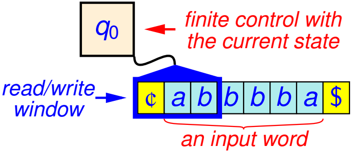
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ
- a *working alphabet* $\Gamma (\supseteq \Sigma)$
- the left and right *sentinels* ¢ and \$
- the *initial state* q_0

Restarting Automaton

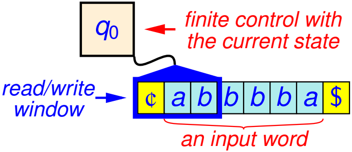
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ
- a *working alphabet* $\Gamma (\supseteq \Sigma)$
- the left and right *sentinels* ¢ and $
- the *initial state* q_0
- a read/write *window* od length k

Restarting Automaton

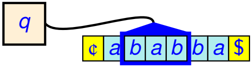
RLWW-automaton



- a finite set of *states* Q
- an *input alphabet* Σ
- a *working alphabet* $\Gamma (\supseteq \Sigma)$
- the left and right *sentinels* ¢ and $
- the *initial state* q_0
- a read/write *window* od length k
- a partial *transition function* δ

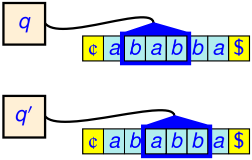
RLWW-Automaton

Possible Steps



RLWW-Automaton

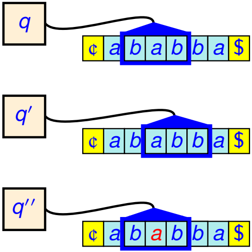
Possible Steps



- move right and change the state

RLWW-Automaton

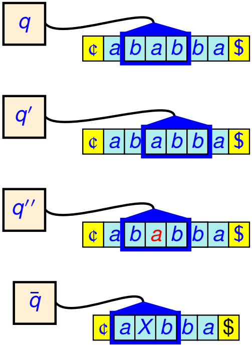
Possible Steps



- move right and change the state
- move left and change the state

RLWW-Automaton

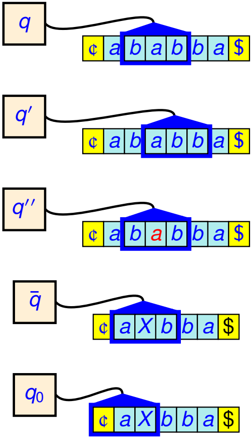
Possible Steps



- move right and change the state
- move left and change the state
- rewrite
  - must shorten the tape,
  - “complete” the window from the left
  - a new state is entered,

RLWW-Automaton

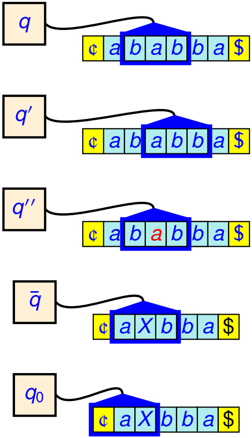
Possible Steps



- move right and change the state
- move left and change the state
- rewrite
  - must shorten the tape,
  - “complete” the window from the left
  - a new state is entered,
- restart

RLWW-Automaton

Possible Steps



- move right and change the state
- move left and change the state
- rewrite
  - must shorten the tape,
  - “complete” the window from the left
  - a new state is entered,
- restart
- accept

RLWW-Automaton

How It Computes

- general RLWW-automata are nondeterministic

RLWW-Automaton

How It Computes

- general RLWW-automata are nondeterministic
- if for a given state and contents of the read/write window the automaton has no instruction, then the automaton halts and rejects

RLWW-Automaton

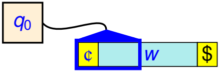
How It Computes

- general RLWW-automata are nondeterministic
- if for a given state and contents of the read/write window the automaton has no instruction, then the automaton halts and rejects
- rewrite and restart steps must alternate

RLWW-Automaton

How It Computes

- general RLWW-automata are nondeterministic
- if for a given state and contents of the read/write window the automaton has no instruction, then the automaton halts and rejects
- rewrite and restart steps must alternate
- a word w ($\in \Sigma^*$) is accepted if there exists a computation starting in the initial state q_0 with w on the tape

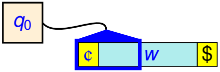


and ending with an *accept* step

RLWW-Automaton

How It Computes

- general RLWW-automata are nondeterministic
- if for a given state and contents of the read/write window the automaton has no instruction, then the automaton halts and rejects
- rewrite and restart steps must alternate
- a word w ($\in \Sigma^*$) is accepted if there exists a computation starting in the initial state q_0 with w on the tape



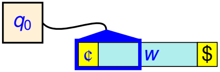
and ending with an *accept* step
- the input language accepted by $\mathcal{M}$

$$L(\mathcal{M}) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

RLWW-Automaton

How It Computes

- general RLWW-automata are nondeterministic
- if for a given state and contents of the read/write window the automaton has no instruction, then the automaton halts and rejects
- rewrite and restart steps must alternate
- a word w ($\in \Sigma^*$) is accepted if there exists a computation starting in the initial state q_0 with w on the tape



and ending with an *accept* step
- the input language accepted by $\mathcal{M}$

$$L(\mathcal{M}) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

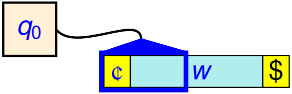
- the basic language accepted by $\mathcal{M}$

$$L_C(\mathcal{M}) = \{w \in \Gamma^* \mid M \text{ accepts } w\}$$

RLWW-Automaton

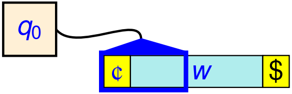
Cycles

- a restarting configuration:



RLWW-Automaton

Cycles



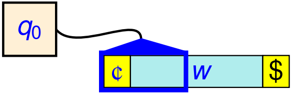
- a restarting configuration:
- a cycle – each part of a computation:

  $\langle$*restarting configuration*$_1\rangle \rightsquigarrow \langle$*restarting configuration*$_2\rangle$

  notation: $w \Rightarrow^c_{\mathcal{M}} w'$ if there is a cycle from restarting configuration with w on the tape and ending by the restart with w' on the tape

RLWW-Automaton

Cycles



- a restarting configuration:
- a cycle – each part of a computation:

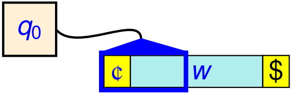
  $\langle$*restarting configuration*$_1\rangle \rightsquigarrow \langle$*restarting configuration*$_2\rangle$

  notation: $w \Rightarrow^{c}_{\mathcal{M}} w'$ if there is a cycle from restarting configuration with w on the tape and ending by the restart with w' on the tape
- a tail – the last part of a computation:

  $\langle$*restarting configuration*$\rangle \rightsquigarrow \langle$*halting configuration*$\rangle$

RLWW-Automaton

Cycles



- a restarting configuration:
- a cycle – each part of a computation:

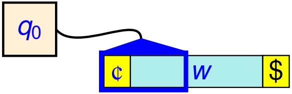
  $\langle$*restarting configuration*$_1\rangle \rightsquigarrow \langle$*restarting configuration*$_2\rangle$

  notation: $w \Rightarrow^c_{\mathcal{M}} w'$ if there is a cycle from restarting configuration with w on the tape and ending by the restart with w' on the tape
- a tail – the last part of a computation:

  $\langle$*restarting configuration*$\rangle \rightsquigarrow \langle$*halting configuration*$\rangle$
- a computation: a sequence of cycles finished by a tail

RLWW-Automaton

Cycles



- a restarting configuration:
- a cycle – each part of a computation:

  $$\langle \textit{restarting configuration}_1 \rangle \rightsquigarrow \langle \textit{restarting configuration}_2 \rangle$$

  notation: $w \Rightarrow^{c}_{\mathcal{M}} w'$ if there is a cycle from restarting configuration with w on the tape and ending by the restart with w' on the tape
- a tail – the last part of a computation:

  $$\langle \textit{restarting configuration} \rangle \rightsquigarrow \langle \textit{halting configuration} \rangle$$

- a computation: a sequence of cycles finished by a tail
- a RLWW(i)-automaton: can execute at most i rewrite instructions per cycle

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \to \Sigma$

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \rightarrow \Sigma$

- maps *working symbol* $\rightarrow$ *input symbol*

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton

h is a homomorphism: $\Gamma \to \Sigma$

- maps *working symbol* $\to$ *input symbol*
- $h(a) = a$ for all input symbols

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

- M is an RLWW(i)-automaton
- h is a homomorphism: $\Gamma \to \Sigma$
  - maps *working symbol* $\to$ *input symbol*
  - $h(a) = a$ for all input symbols
- the input language $L(\widehat{M}) = L(M)$

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \to \Sigma$

- maps *working symbol* $\to$ *input symbol*
- $h(a) = a$ for all input symbols

- the input language $L(\widehat{M}) = L(M)$
- the basic language $L_{\mathrm{C}}(\widehat{M}) = L_{\mathrm{C}}(M)$

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \to \Sigma$

- maps *working symbol* $\to$ *input symbol*
- $h(a) = a$ for all input symbols

- the input language $L(\widehat{M}) = L(M)$
- the basic language $L_{\mathrm{C}}(\widehat{M}) = L_{\mathrm{C}}(M)$
- the h-proper language accepted by $\widehat{M}$

$$L_{\mathrm{hP}}(\widehat{M}) = h(L_{\mathrm{C}}(M))$$

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \to \Sigma$

- maps *working symbol* $\to$ *input symbol*
- $h(a) = a$ for all input symbols

- the input language $L(\widehat{M}) = L(M)$
- the basic language $L_{\mathrm{C}}(\widehat{M}) = L_{\mathrm{C}}(M)$
- the h-proper language accepted by $\widehat{M}$

$$L_{\mathrm{hP}}(\widehat{M}) = h(L_{\mathrm{C}}(M))$$

- h-lexicalized syntactic analysis

$$L_{\mathrm{A}}(\widehat{M}) = \{(h(w), w) \mid w \in L_{\mathrm{C}}(M)\}$$

h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

$\widehat{M} = (M, h)$ where

M is an RLWW(i)-automaton
h is a homomorphism: $\Gamma \to \Sigma$

- maps *working symbol* $\to$ *input symbol*
- $h(a) = a$ for all input symbols

- the input language $L(\widehat{M}) = L(M)$
- the basic language $L_{\mathrm{C}}(\widehat{M}) = L_{\mathrm{C}}(M)$
- the h-proper language accepted by $\widehat{M}$

$$L_{\mathrm{hP}}(\widehat{M}) = h(L_{\mathrm{C}}(M))$$

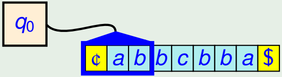
- h-lexicalized syntactic analysis

$$L_{\mathrm{A}}(\widehat{M}) = \{(h(w), w) \mid w \in L_{\mathrm{C}}(M)\}$$

- obviously $L(\widehat{M}) \subseteq L_{\mathrm{hP}}(\widehat{M}) = h(L_{\mathrm{C}}(\widehat{M}))$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

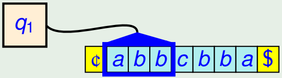
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
& \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
& \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
& \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

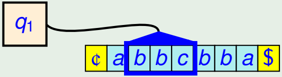
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\le 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{\, wcw^R \mid w \in \{a,b\}^* \,\}$

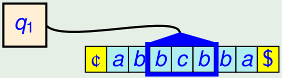
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\},\ z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{wcw^R \mid w \in \{a,b\}^*\}$

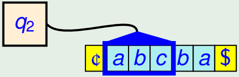
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

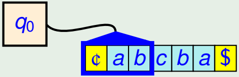
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

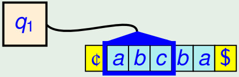
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\},\ z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{\, wcw^R \mid w \in \{a,b\}^* \,\}$

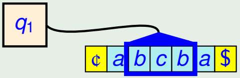
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

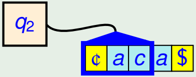
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\},\ z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

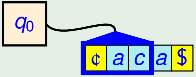
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

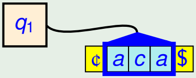
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

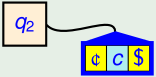
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{\, wcw^R \mid w \in \{a, b\}^* \,\}$

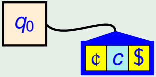
Example 1



$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a, b\}, y \in \{a, b, c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a, b\}, z \in \{a, b, c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

Example 1

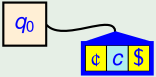


$L(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$

$$
\begin{array}{ll}
(1) & \delta(q_0, ¢c\$) = \{\text{Accept}\}, \\
(2) & \delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}, \\
 & \quad \text{for all } x \in \{a,b\}, y \in \{a,b,c\}, \\
(3) & \delta(q_1, aca) = \{(q_2, c)\}, \\
(4) & \delta(q_1, bcb) = \{(q_2, c)\}, \\
(5) & \delta(q_1, xyz) = \{(q_1, \text{MVR})\}, \\
 & \text{for all } x, y \in \{a,b\}, z \in \{a,b,c\}, \\
(6) & \delta(q_2, u) = \{\text{Restart}\}, \\
 & \quad \text{for all } u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}.
\end{array}
$$

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

Example 1

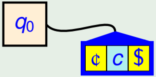


$$L(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$$
$$L_{\mathrm{C}}(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$$

(1) $\delta(q_0, ¢c\$) = \{\text{Accept}\}$,
(2) $\delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}$,
for all $x \in \{a,b\}, y \in \{a,b,c\}$,
(3) $\delta(q_1, aca) = \{(q_2, c)\}$,
(4) $\delta(q_1, bcb) = \{(q_2, c)\}$,
(5) $\delta(q_1, xyz) = \{(q_1, \text{MVR})\}$,
for all $x, y \in \{a,b\}$, $z \in \{a,b,c\}$,
(6) $\delta(q_2, u) = \{\text{Restart}\}$,
for all $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}$.

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a, b\}^* \}$

Example 1



$L(M) = \{ wcw^R \mid w \in \{a, b\}^* \}$

$L_{\mathrm{C}}(M) = \{ wcw^R \mid w \in \{a, b\}^* \}$

$L_{\mathrm{hP}}(M) = \{ wcw^R \mid w \in \{a, b\}^* \}$

(1) $\delta(q_0, ¢c\$) = \{\text{Accept}\}$,

(2) $\delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}$, for all $x \in \{a, b\}, y \in \{a, b, c\}$,

(3) $\delta(q_1, aca) = \{(q_2, c)\}$,

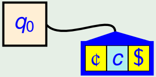
(4) $\delta(q_1, bcb) = \{(q_2, c)\}$,

(5) $\delta(q_1, xyz) = \{(q_1, \text{MVR})\}$, for all $x, y \in \{a, b\}$, $z \in \{a, b, c\}$,

(6) $\delta(q_2, u) = \{\text{Restart}\}$, for all $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}$.

RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\mathrm{pal},c} = \{ wcw^R \mid w \in \{a,b\}^* \}$

Example 1



$L(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$

$L_{\mathrm{C}}(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$

$L_{\mathrm{hP}}(M) = \{ wcw^R \mid w \in \{a,b\}^* \}$

(1) $\delta(q_0, ¢c\$) = \{\text{Accept}\}$,

(2) $\delta(q_0, ¢xy) = \{(q_1, \text{MVR})\}$,
for all $x \in \{a,b\}, y \in \{a,b,c\}$,

(3) $\delta(q_1, aca) = \{(q_2, c)\}$,

(4) $\delta(q_1, bcb) = \{(q_2, c)\}$,

(5) $\delta(q_1, xyz) = \{(q_1, \text{MVR})\}$,
for all $x, y \in \{a,b\}$, $z \in \{a,b,c\}$,

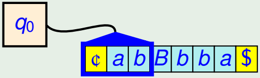
(6) $\delta(q_2, u) = \{\text{Restart}\}$,
for all $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}$.

only contextual rewritings = deletes only, at most 2 factors in one rewrite step

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

Example 2

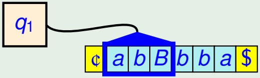
$$\begin{aligned}\Sigma &= \{a,b\},\\ \Gamma &= \{a,b,A,B\},\\ &\quad h(A) = a,\ h(B) = b,\end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a, b\}^*\}$

Example 2

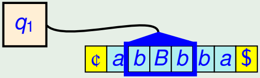
$$\begin{aligned} \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, A, B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

Example 2

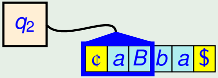
$$\begin{aligned} \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, A, B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

Example 2

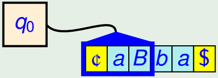
$$\Sigma = \{a, b\},$$
$$\Gamma = \{a, b, A, B\},$$
$$h(A) = a,\ h(B) = b,$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

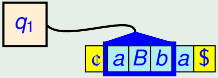
Example 2

$$\begin{aligned} \Sigma &= \{a,b\}, \\ \Gamma &= \{a,b,A,B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

Example 2

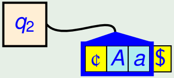


$$\begin{aligned} \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, A, B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ ww^R \mid w \in \{a, b\}^* \}$

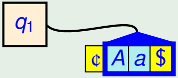
Example 2

$$\begin{aligned} \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, A, B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ww^R \mid w \in \{a,b\}^*\}$

Example 2

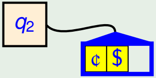


$$\begin{aligned} \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, A, B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{\, ww^R \mid w \in \{a,b\}^* \,\}$

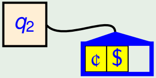
Example 2

$$\begin{aligned} \Sigma &= \{a,b\}, \\ \Gamma &= \{a,b,A,B\}, \\ & h(A) = a,\ h(B) = b, \end{aligned}$$



RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \{ ww^R \mid w \in \{a, b\}^* \}$

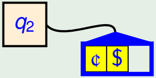
Example 2



$$
\begin{aligned}
\Sigma &= \{a, b\}, \\
\Gamma &= \{a, b, A, B\}, \\
& h(A) = a,\ h(B) = b, \\
L(M) &= \{\lambda\}
\end{aligned}
$$

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \left\{ ww^R \mid w \in \{a, b\}^* \right\}$

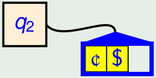
Example 2



$$
\begin{aligned}
\Sigma &= \{a, b\}, \\
\Gamma &= \{a, b, A, B\}, \\
& h(A) = a,\ h(B) = b, \\
L(M) &= \{\lambda\} \\
L_{\mathrm{C}}(M) &= \left\{ wAaw^R, wBbw^R \mid w \in \{a, b\}^* \right\} \cup \{\lambda\}
\end{aligned}
$$

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \left\{ ww^R \mid w \in \{a, b\}^* \right\}$

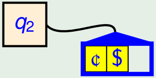
Example 2



$$\begin{aligned}
\Sigma &= \{a, b\}, \\
\Gamma &= \{a, b, A, B\}, \\
&\quad h(A) = a,\ h(B) = b, \\
L(M) &= \{\lambda\} \\
L_{\mathrm{C}}(M) &= \left\{ wAaw^R, wBbw^R \mid w \in \{a, b\}^* \right\} \cup \{\lambda\} \\
L_{\mathrm{hP}}(M) &= \left\{ ww^R \mid w \in \{a, b\}^* \right\}
\end{aligned}$$

RLWW-Automaton Accepting Even Palindromes $L_{\mathrm{pal}} = \left\{ ww^R \mid w \in \{a, b\}^* \right\}$

Example 2



$$\begin{aligned}
\Sigma &= \{a, b\}, \\
\Gamma &= \{a, b, A, B\}, \\
& h(A) = a,\ h(B) = b, \\
L(M) &= \{\lambda\} \\
L_{\mathrm{C}}(M) &= \left\{ wAaw^R, wBbw^R \mid w \in \{a, b\}^* \right\} \cup \{\lambda\} \\
L_{\mathrm{hP}}(M) &= \left\{ ww^R \mid w \in \{a, b\}^* \right\}
\end{aligned}$$

- only contextual rewritings = deletes only, at most 2 factors in one rewrite step

The Power of RLWW(i)-automata

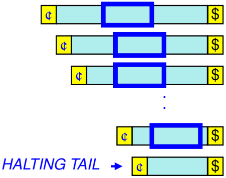
- $\mathcal{L}$(RLWW(i)) – the class of input languages accepted by RLWW(i)-automata – properly includes CFL

The Power of RLWW(i)-automata

- $\mathcal{L}(\mathrm{RLWW}(i))$ – the class of input languages accepted by RLWW(i)-automata – properly includes CFL
- the class of growing context sensitive languages is a proper subclass of $\mathcal{L}(\mathrm{RLWW}(1))$

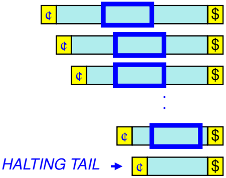
The Power of RLWW(i)-automata

- $\mathcal{L}(\mathrm{RLWW}(i))$ – the class of input languages accepted by RLWW(i)-automata – properly includes CFL
- the class of growing context sensitive languages is a proper subclass of $\mathcal{L}(\mathrm{RLWW}(1))$
- a monotone computation: the places of rewriting do not increase their distance from the right sentinel



The Power of RLWW(i)-automata

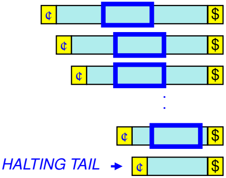
- $\mathcal{L}(\mathrm{RLWW}(i))$ – the class of input languages accepted by RLWW(i)-automata – properly includes CFL
- the class of growing context sensitive languages is a proper subclass of $\mathcal{L}(\mathrm{RLWW}(1))$
- a monotone computation: the places of rewriting do not increase



their distance from the right sentinel
- a monotone automaton: all its computations are monotone

The Power of RLWW(i)-automata

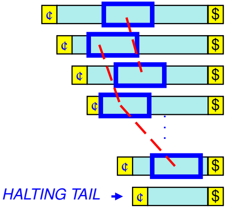
- $\mathcal{L}$(RLWW(i)) – the class of input languages accepted by RLWW(i)-automata – properly includes CFL
- the class of growing context sensitive languages is a proper subclass of $\mathcal{L}$(RLWW(1))
- a monotone computation: the places of rewriting do not increase their distance from the right sentinel



- a monotone automaton: all its computations are monotone
- $\mathcal{L}(\text{mon-RLWW}(1)) = \text{CFL}$

j-Monotone Automata

- a *j*-monotone computation: the places of rewriting can be partitioned into at most *j* (noncontiguous) monotone subsequences



- a *j*-monotone automaton: all its computations are *j*-monotone

Basic Correctness Preserving Property

- an hRLWW(i)-automaton is basically correctness preserving if $u \Rightarrow_M^{c^*} v$ and $u \in L_{\mathrm{C}}(M)$ induce that $v \in L_{\mathrm{C}}(M)$, and therewith $h(v) \in L_{\mathrm{hP}}(M)$ and $(h(v), v) \in L_{\mathrm{A}}(M)$
- **Fact:** each deterministic hRLWW(i)-automaton is basically correctness preserving.

Strong Cyclic Form

An hRLWW-automaton M is in strong cyclic form if it does not halt on any word of length greater than the size of its read/write window

Lemma 3

Each RLWW(i)*-automaton* M *can be transformed into* scf-RLWW(i)*-automaton* M_{scf} *such that*

- $L_{\mathrm{C}}(M) = L_{\mathrm{C}}(M_{\mathrm{scf}})$,
- $u \Rightarrow^{c^*}_{M} v$ *implies* $u \Rightarrow^{c^*}_{M_{\mathrm{scf}}} v$, *for all words* u, v,
- *all reductions of* M_{scf} *that are not possible for* M *are in contextual form – they do not rewrite, delete at most two factors,*
- *if* M *is deterministic and/or* j*-monotone, then* M_{scf} *is deterministic and/or* j*-monotone.*

Strong Cyclic Form

Context-Free Constructions

- LRR $=$ the class of left-to-right regular languages
- syn-RLWW(i) means j-mon-RLWW(i) where $j \leq i$

Theorem 4

Let $X \in \{\text{hRLWW}(1), \text{hRLWWD}(1), \text{hRLWWC}(1)\}$*. Then*

- LRR $= \mathcal{L}_{\mathrm{C}}$(scf-det-syn-$X$) *and*
- CFL $= \mathcal{L}_{\mathrm{hP}}$(scf-det-syn-$X$).

Sensitivity to the Size of Window

- Basic and h-proper languages of scf-hRLWW(i)-automata are sensitive to the size of their windows, to the number of deletions by a reduction, and to the degree of monotonicity.
- small finite witness languages

Lemma 5

For $k \geq 2$:

(a) $\{a^k\} \in \mathcal{L}_C(k\text{-scf-fin}(0)\text{-det-mon-RLWC})$.

- RLWC*-automata: no auxiliary (non-input) symbols, contextual instructions only*
- fin(0)*- at most* 0 *cycles in each accepting computation*
- k *is the length of window*

(b) $\{a^k\} \notin \mathcal{L}_C((k-1)\text{-scf-hRLWW}) \cup \mathcal{L}_{hP}((k-1)\text{-scf-hRLWW})$.

Sensitivity to the Number of Rewrites per Cycle

- witness languages of cardinality two

Lemma 6

Let $k, j \geq 1$, *let* $L_2(j,k) = \{a^{k\cdot(j+1)}, a^k\}$.

(a) $L_2(j,k) \in \mathcal{L}_{\mathrm{C}}(k\text{-scf-fin(1)-det-mon-RLWC}(j))$.

(b) $L_2(j,k) \notin \mathcal{L}_{\mathrm{C}}(k\text{-scf-hRLWW}(j')) \cup \mathcal{L}_{\mathrm{hP}}(k\text{-scf-hRLWW}(j'))$ *for any* $j' < j$.

Sensitivity to the Degree of Monotonicity

- finite witness languages

Lemma 7

Let $k, j \geq 2$. *There exist languages* $L_3(j,k) \subset \{a,b,c\}^*$ *of cardinality* $j^2 + j + 1$ *such that:*

(a) $L_3(j,k) \in \mathcal{L}_{\mathrm{C}}(k\text{-scf-fin}(j+j^2)\text{-det-mon}(j)\text{-RLWC}(j))$.

(b) $L_3(j,k) \notin \mathcal{L}_{\mathrm{C}}(k\text{-scf-mon}(j')\text{-hRLWW}(j)) \cup \mathcal{L}_{\mathrm{hP}}(k\text{-scf-mon}(j')\text{-hRLWW}(j))$ *for any* $j' < j$.

(c) $L_3(j,k) \notin \mathcal{L}_{\mathrm{C}}(k\text{-scf-hRLWW}(j')) \cup \mathcal{L}_{\mathrm{hP}}(k\text{-scf-hRLWW}(j'))$ *for any* $j' < j$.

Hierarchy of Contextually Transparent Language Classes

CTL(i) = the class of h-proper languages accepted by hRLWW(i)-automata that are

- deterministic, contextual, in the strong cyclic form
- synchronized – mon-(i)

CTLA(i) = the class of lexicalized analyses corresponding to CTL(i)

$$\begin{tikzcd}
\mathrm{CTL} \arrow[r] & \mathrm{CSL} \\
\vdots \arrow[u] & \\
\mathrm{CTL}(3) \arrow[u] & \\
\mathrm{CTL}(2) \arrow[u] & \\
\mathrm{CTL}(1) \arrow[u] \arrow[r, phantom, "="] & \mathrm{CFL} \arrow[uuuu]
\end{tikzcd}$$

$$\begin{tikzcd}
\mathrm{CTLA} & \\
\vdots \arrow[u] & \\
\mathrm{CTLA}(3) \arrow[u] & \\
\mathrm{CTLA}(2) \arrow[u] & \\
\mathrm{CTLA}(1) \arrow[u] \arrow[r, phantom, "="] & \mathrm{CFLA}
\end{tikzcd}$$

CTL $\subsetneq$ CSL

$\subseteq$: easy

$\subsetneq$: $L_{\mathrm{e}} = \{a^{2^n} \mid n \geq 0\} \notin$ CTL by contradiction

- if $a^{2^n} \in L_{\mathrm{hP}}(M)$ for some k-hRLWW(i)-automaton, then $a^{2^n} = h(w)$ for some $w \in L_{\mathrm{C}}(M)$ and there exists an accepting computation of M on w; the accepting computation contains at least one cycle
- the cycle starts by a reduction $w \Rightarrow_M w'$, where $|w| > |w'| \geq |w| - k \cdot i$ and $h(w') \in L_{\mathrm{hP}}(M)$
- for sufficiently large n, the length of $h(w')$ cannot be a power of 2 $\Rightarrow$ $h(w') \notin L_{\mathrm{hp}}(M)$ – a contradiction

A Refinement With Respect to the Window Size

h-Proper Language Classes

k-CTL(i) $=$ the class of h-proper languages accepted by hRLWW(i)-automata that are

- deterministic, contextual, in the strong cyclic form
- synchronized – mon-(i)
- of window size k

$$
\begin{tikzcd}
\text{1-CTL} \arrow[r] & \text{2-CTL} \arrow[r] & \text{3-CTL} \arrow[r] & \cdots \arrow[r] & \text{CTL} \arrow[r] & \text{CSL} \\
\vdots \arrow[u] & \vdots \arrow[u] & \vdots \arrow[u] & & \vdots \arrow[u] & \\
\text{1-CTL(3)} \arrow[r] \arrow[u] & \text{2-CTL(3)} \arrow[r] \arrow[u] & \text{3-CTL(3)} \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTL(3)} \arrow[u] & \\
\text{1-CTL(2)} \arrow[r] \arrow[u] & \text{2-CTL(2)} \arrow[r] \arrow[u] & \text{3-CTL(2)} \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTL(2)} \arrow[u] & \\
\text{1-CTL(1)} \arrow[r] \arrow[u] & \text{2-CTL(1)} \arrow[r] \arrow[u] & \text{3-CTL(1)} \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTL(1)} \arrow[u] \arrow[r, phantom, "="] & \text{CFL} \arrow[uuuu]
\end{tikzcd}
$$

A Refinement With Respect to the Window Size

Lexicalized Analyses

k-CTLA(i) $=$ the class of lexicalized analyses by hRLWW(i)-automata that are

- deterministic, contextual, in the strong cyclic form
- synchronized – mon-(i)
- of window size k

$$
\begin{tikzcd}
1\text{-CTLA} \arrow[r] & 2\text{-CTLA} \arrow[r] & 3\text{-CTLA} \arrow[r] & \cdots \arrow[r] & \text{CTLA} \\
\vdots \arrow[u] & \vdots \arrow[u] & \vdots \arrow[u] & & \vdots \arrow[u] \\
1\text{-CTLA}(3) \arrow[r] \arrow[u] & 2\text{-CTLA}(3) \arrow[r] \arrow[u] & 3\text{-CTLA}(3) \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTLA}(3) \arrow[u] \\
1\text{-CTLA}(2) \arrow[r] \arrow[u] & 2\text{-CTLA}(2) \arrow[r] \arrow[u] & 3\text{-CTLA}(2) \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTLA}(2) \arrow[u] \\
1\text{-CTLA}(1) \arrow[r] \arrow[u] & 2\text{-CTLA}(1) \arrow[r] \arrow[u] & 3\text{-CTLA}(1) \arrow[r] \arrow[u] & \cdots \arrow[r] & \text{CTLA}(1) = \text{CFLA} \arrow[u]
\end{tikzcd}
$$

Conclusions (1)

- det-hRLWW(i)-automata are correctness preserving with respect to their basic and h-proper languages, lexicalized syntactic analysis and analysis by reduction.

Conclusions (1)

- det-hRLWW(i)-automata are correctness preserving with respect to their basic and h-proper languages, lexicalized syntactic analysis and analysis by reduction.
- The basic correctness preserving property enforces the sensitivity to the degree of synchronization, number of rewrites in a cycle, and to the size of the window.

Conclusions (1)

- det-hRLWW(i)-automata are correctness preserving with respect to their basic and h-proper languages, lexicalized syntactic analysis and analysis by reduction.
- The basic correctness preserving property enforces the sensitivity to the degree of synchronization, number of rewrites in a cycle, and to the size of the window.
- **Conjecture:** The class 12-CTLA(2) is strong enough to model the lexicalized surface syntax of Czech (lexicalized sentence analysis based on PDT).

Conclusions (1)

- det-hRLWW(i)-automata are correctness preserving with respect to their basic and h-proper languages, lexicalized syntactic analysis and analysis by reduction.
- The basic correctness preserving property enforces the sensitivity to the degree of synchronization, number of rewrites in a cycle, and to the size of the window.
- **Conjecture:** The class 12-CTLA(2) is strong enough to model the lexicalized surface syntax of Czech (lexicalized sentence analysis based on PDT).
- Long term goal: to propose and support a formal (and possibly also software) environment for a further study and development of Functional Generative Description (FGD) of Czech.

Conclusions (1)

- det-hRLWW(i)-automata are correctness preserving with respect to their basic and h-proper languages, lexicalized syntactic analysis and analysis by reduction.
- The basic correctness preserving property enforces the sensitivity to the degree of synchronization, number of rewrites in a cycle, and to the size of the window.
- **Conjecture:** The class 12-CTLA(2) is strong enough to model the lexicalized surface syntax of Czech (lexicalized sentence analysis based on PDT).
- Long term goal: to propose and support a formal (and possibly also software) environment for a further study and development of Functional Generative Description (FGD) of Czech.
- **Conjecture:** The lexicalized syntactic analysis of full (four level) FGD can be described by tools very close to 24-CTLA(4).

Conclusions (2)

- Phrase-structure grammars:

Conclusions (2)

- Phrase-structure grammars:
  - categories bound to individual constituents (parts of a sentence) $\Rightarrow$ not any correctness preserving property is possible,

Conclusions (2)

- Phrase-structure grammars:
  - categories bound to individual constituents (parts of a sentence) $\Rightarrow$ not any correctness preserving property is possible,
  - not sensitive to the size of individual grammar rules – cf. Chomsky normal form,

Conclusions (2)

- Phrase-structure grammars:
  - categories bound to individual constituents (parts of a sentence) $\Rightarrow$ not any correctness preserving property is possible,
  - not sensitive to the size of individual grammar rules – cf. Chomsky normal form,
  - not any kind of classification of finite syntactic constructions of (natural) languages.

Conclusions (2)

- Phrase-structure grammars:
  - categories bound to individual constituents (parts of a sentence) $\Rightarrow$ not any correctness preserving property is possible,
  - not sensitive to the size of individual grammar rules – cf. Chomsky normal form,
  - not any kind of classification of finite syntactic constructions of (natural) languages.
- In traditional and corpus linguistics, only finite language phenomena can be directly observed. The basic and h-proper languages of hRLWWC(i)-automata in strong cyclic form with constraints on the window size allow common classifications of finite phenomena as well as their infinite relaxations.

Conclusions (2)

- Phrase-structure grammars:
  - categories bound to individual constituents (parts of a sentence) $\Rightarrow$ not any correctness preserving property is possible,
  - not sensitive to the size of individual grammar rules – cf. Chomsky normal form,
  - not any kind of classification of finite syntactic constructions of (natural) languages.
- In traditional and corpus linguistics, only finite language phenomena can be directly observed. The basic and h-proper languages of hRLWWC(i)-automata in strong cyclic form with constraints on the window size allow common classifications of finite phenomena as well as their infinite relaxations.
- Many practical problems in computational and corpus linguistic become decidable when we only consider languages parametrized by the size of the windows, or even easier when they are parametrized by a finite number of reductions.

Thank you for your attention!