Lexicalized Syntactic Analysis by Restarting Automata

František Mráz\textsuperscript{a}  Friedrich Otto\textsuperscript{a}
Dana Pardubská\textsuperscript{b1}  Martin Plátek\textsuperscript{a2}

\textsuperscript{a}Charles University, Prague, Czech Republic
\textsuperscript{b}Comenius University, Bratislava, Slovakia

PSC 2019, Prague, August 27, 2019

\textsuperscript{1}The research was partially supported by VEGA 2/0165/16.
\textsuperscript{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.
Lexicalized Syntactic Analysis

1 input sentence

PSC means an interesting conference
Lexicalized Syntactic Analysis

1. input sentence
2. lexicalization

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

← 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, and shortens.

← each reduction
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
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.*

*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
- detecting (in)dependencies within a sentence
Restarting Automaton

RLWW-automaton

- Definitions
- A finite set of states $Q$
- A finite control with the current state $q_0$
- An input word $\$ a b b b b a\$
Restarting Automaton

RLWW-automaton

- a finite set of states $Q$
- an input alphabet $\Sigma$

Definitions
Restarting Automaton

RLWW-automaton

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

Definitions

Finite control with the current state

An input word

Read/write window
Restarting Automaton

RLWW-automaton

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

Definitions

- $\ddagger a b b b b a \$$
- read/write window
- finite control with the current state
- $q_0$
- an input word

- $\ddagger a b b b b a \$$
- finite control with the current state
- $q_0$
- an input word
Restarting Automaton

RLWW-automaton

- a finite set of states $Q$
- an input alphabet $\Sigma$
- a working alphabet $\Gamma(\supseteq \Sigma)$
- the left and right sentinels $\mathdollar{\text{\$}}$ and $\mathdollar{\text{\$}}$
- the initial state $q_0$


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

**RLWW-automaton**

- a finite set of states $Q$
- an input alphabet $\Sigma$
- a working alphabet $\Gamma(\supseteq \Sigma)$
- the left and right sentinels $c$ and $\$$
- the initial state $q_0$
- a read/write window of length $k$
- a partial transition function $\delta$
RLWW-Automaton
Possible Steps

Definitions

F. Mráz, F. Otto, D. Pardubská, M. Plátek
Lexicalized Syntactic Analysis
PSC 2019, Aug 27, 2019
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
Definitions

**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**
general RLWW-automata are nondeterministic
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
Definitions

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
Definitions

**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

![Diagram of RLWW-Automaton]

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 $M$

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

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

- the basic language accepted by $M$

$$L_C(M) = \{ w \in \Gamma^* \mid M \text{ accepts } w \}$$
a restarting configuration:

\[ q_0 \]
a restarting configuration:

a cycle – each part of a computation:

\[ \langle \text{restarting configuration}_1 \rangle \sim \langle \text{restarting configuration}_2 \rangle \]

notation: \( w \xrightarrow{c}_{M} w' \) if there is a cycle from restarting configuration with \( w \) on the tape and ending by the restart with \( w' \) on the tape
a restarting configuration:

a cycle – each part of a computation:

\[ \langle \text{restarting configuration}_1 \rangle \rightsquigarrow \langle \text{restarting configuration}_2 \rangle \]

notation: \( w \Rightarrow^c_M w' \) 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:

\[ \langle \text{restarting configuration} \rangle \rightsquigarrow \langle \text{halting configuration} \rangle \]
RLWW-Automaton

Cycles

- a restarting configuration:
- a cycle – each part of a computation:
  \[ \langle \text{restarting configuration}_1 \rangle \rightsquigarrow \langle \text{restarting configuration}_2 \rangle \]
  notation: \( w \Rightarrow^c_M w' \) 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:
  \[ \langle \text{restarting configuration} \rangle \rightsquigarrow \langle \text{halting configuration} \rangle \]
- a computation: a sequence of cycles finished by a tail
RLWW-Automaton

Cycles

- a restarting configuration:
- a cycle – each part of a computation:
  \( \langle \text{restarting configuration}_1 \rangle \leadsto \langle \text{restarting configuration}_2 \rangle \)
  notation: \( w \Rightarrow^c_M w' \) 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:
  \( \langle \text{restarting configuration} \rangle \leadsto \langle \text{halting configuration} \rangle \)
- 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

\( \hat{M} = (M, h) \) where

\( M \) is an RLWW(i)-automaton
h-Lexicalized RLWW(i)-automaton

hRLWW(i)-Automaton

\( \hat{M} = (M, h) \) where

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

\( \hat{M} = (M, h) \) where

- \( M \) is an \( RLWW(i) \)-automaton
- \( h \) is a homomorphism: \( \Gamma \rightarrow \Sigma \)
  - maps working symbol \( \rightarrow \) input symbol
\[ \hat{M} = (M, h) \] where

- \( M \) is an RLWW(i)-automaton
- \( h \) is a homomorphism: \( \Gamma \rightarrow \Sigma \)
  - maps working symbol \( \rightarrow \) input symbol
  - \( h(a) = a \) for all input symbols
**h-Lexicalized RLWW(i)-automaton**

\[ \hat{M} = (M, h) \] where

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

\( \hat{M} = (M, h) \) 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 \( L(\hat{M}) = L(M) \)
- the basic language \( L_C(\hat{M}) = L_C(M) \)
**Definitions**

**h-Lexicalized RLWW(i)-automaton**

**hRLWW(i)-Automaton**

\( \hat{M} = (M, h) \) where

- \( M \) is an RLWW(i)-automaton
- \( h \) is a homomorphism: \( \Gamma \rightarrow \Sigma \)
  - maps *working symbol* → *input symbol*
  - \( h(a) = a \) for all input symbols
- the input language \( L(\hat{M}) = L(M) \)
- the basic language \( L_C(\hat{M}) = L_C(M) \)
- the h-proper language accepted by \( \hat{M} \)

\[
L_{\text{hP}}(\hat{M}) = h(L_C(M))
\]
Definitions

h-Lexicalized RLWW(i)-automaton

$hRLWW(i)$-Automaton

$\hat{M} = (M, h)$ 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 $L(\hat{M}) = L(M)$
- the basic language $L_C(\hat{M}) = L_C(M)$
- the $h$-proper language accepted by $\hat{M}$

$$L_{hP}(\hat{M}) = h(L_C(M))$$

- $h$-lexicalized syntactic analysis

$$L_A(\hat{M}) = \{(h(w), w) \mid w \in L_C(M)\}$$
h-Lexicalized RLWW(i)-automaton

\( \hat{M} = (M, h) \) 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 \( L(\hat{M}) = L(M) \)
- the basic language \( L_C(\hat{M}) = L_C(M) \)
- the h-proper language accepted by \( \hat{M} \)

\[
L_{hP}(\hat{M}) = h(L_C(M))
\]

- h-lexicalized syntactic analysis

\[
L_A(\hat{M}) = \{ (h(w), w) \mid w \in L_C(M) \}
\]

- obviously \( L(\hat{M}) \subseteq L_{hP}(\hat{M}) = h(L_C(\hat{M})) \)
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\text{pal},c} = \{wcw^R \mid w \in \{a, b\}^*\}$

Example 1

1. $\delta(q_0, cc\$) = \{\text{Accept}\}$,
2. $\delta(q_0, cxy) = \{(q_1, \text{MVR})\}$, 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 $L_{pal,c} = \{ wcw^R \mid w \in \{a, b\}^* \}$

Example 1

(1) $\delta(q_0, cc$) = $\{\text{Accept}\}$,
(2) $\delta(q_0, cxy) = \{(q_1, \text{MVR})\}$,
  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 $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{$}.  

F. Mráz, F. Otto, D. Pardubská, M. Plátek
Lexicalized Syntactic Analysis
PSC 2019, Aug 27, 2019 10 / 26
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{pal,c} = \{wcw^R | w \in \{a, b\}^*\}$

Example 1

(1) $\delta(q_0, cc$) = $\{\text{Accept}\}$,
(2) $\delta(q_0, cxy) = \{(q_1, \text{MVR})\}$,
   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 $u \in \Gamma^3 \cup \Gamma^\leq2 \cdot \{\$\}$. 
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{pal,c} = \{wcw^R | w \in \{a, b\}^*\}$

Example 1

(1) $\delta(q_0, cc$) = \{Accept\},
(2) $\delta(q_0, cxy) = \{(q_1, MVR)\}$, 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) $\delta(q_1, xyz) = \{(q_1, MVR)\}$, for all $x, y \in \{a, b\}, z \in \{a, b, c\}$,
(6) $\delta(q_2, u) = \{Restart\}$, for all $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\}$.
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{pal,c} = \{wcw^R | w \in \{a, b\}^*\}$

Example 1

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

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

Example 1

(1) $\delta(q_0, cc) = \{\text{Accept}\}$,
(2) $\delta(q_0, cxy) = \{(q_1, \text{MVR})\}$,
    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 $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cup \{\$\}$. 
Definitions

**RLWW-Automaton Accepting Palindromes with Marked Centers**

$L_{\text{pal},c} = \{wcw^R \mid w \in \{a, b\}^*\}$

**Example 1**

1. $\delta(q_0, \text{cc}$) = \{Accept\},
2. $\delta(q_0, \text{cxy}) = \{(q_1, \text{MVR})\}$, for all $x \in \{a, b\}, y \in \{a, b, c\}$,
3. $\delta(q_1, \text{aca}) = \{(q_2, c)\}$,
4. $\delta(q_1, \text{bcb}) = \{(q_2, c)\}$,
5. $\delta(q_1, \text{xyz}) = \{(q_1, \text{MVR})\}$, 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 $L_{pal,c} = \{wcw^R \mid w \in \{a, b\}^*\}$

Example 1

\begin{itemize}
  \item[(1)] $\delta(q_0, \text{cc}$) = $\{\text{Accept}\}$,
  \item[(2)] $\delta(q_0, \text{cxy}) = \{(q_1, \text{MVR})\}$,
    \hspace{1cm} for all $x \in \{a, b\}$, $y \in \{a, b, c\}$,
  \item[(3)] $\delta(q_1, \text{aca}) = \{(q_2, c)\}$,
  \item[(4)] $\delta(q_1, \text{bcb}) = \{(q_2, c)\}$,
  \item[(5)] $\delta(q_1, \text{xyz}) = \{(q_1, \text{MVR})\}$,
    \hspace{1cm} for all $x, y \in \{a, b\}$, $z \in \{a, b, c\}$,
  \item[(6)] $\delta(q_2, u) = \{\text{Restart}\}$,
    \hspace{1cm} for all $u \in \Gamma^3 \cup \Gamma^2 \cdot \{$}.
\end{itemize}
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{pal,c} = \{ wcw^R \mid w \in \{a, b\}^* \}$

Example 1

(1) $\delta(q_0, cc\$) = \{\text{Accept}\}$,
(2) $\delta(q_0, cxy) = \{(q_1, \text{MVR})\}$,
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 $u \in \Gamma^3 \cup \Gamma^{\leq 2} \cdot \{\$\}$. 
Definitions

**RLWW-Automaton Accepting Palindromes with Marked Centers**

\[ L_{pal,c} = \{ wcw^R \mid w \in \{a, b\}^* \} \]

**Example 1**

\[ q_0 \]

\[ \$ a c a c \$

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

(2) \[ \delta(q_0, cxy) = \{(q_1, \text{MVR})\} , \]

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 \( L_{\text{pal},c} = \{ wcw^R | w \in \{ a, b \}^* \} \)

Example 1

\( q_1 \)  
\( \text{a} \quad \text{c} \quad \text{a} \quad \text{c} \quad \$ \)

1. \( \delta(q_0, \text{c$c$}) = \{ \text{Accept} \} \),
2. \( \delta(q_0, \text{c$xy$}) = \{ (q_1, \text{MVR}) \} \),
   for all \( x \in \{ a, b \}, y \in \{ a, b, c \} \),
3. \( \delta(q_1, \text{aca}) = \{ (q_2, c) \} \),
4. \( \delta(q_1, \text{bcb}) = \{ (q_2, c) \} \),
5. \( \delta(q_1, \text{xyz}) = \{ (q_1, \text{MVR}) \} \),
   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 \( L_{\text{pal},c} = \{ wcw^R \mid w \in \{ a, b \}^* \} \)

Example 1

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

**RLWW-Automaton Accepting Palindromes with Marked Centers**

$L_{\text{pal}, \text{c}} = \{wcw^R \mid w \in \{a, b\}^*\}$

**Example 1**

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

1. \(\delta(q_0, cc\$) = \{\text{Accept}\},\)
2. \(\delta(q_0, cxy) = \{(q_1, \text{MVR})\},\)
   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 \( L_{\text{pal},c} = \{wcw^R \mid w \in \{a, b\}^*\} \)

**Example 1**

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

\[ L_C(M) = \{wcw^R \mid w \in \{a, b\}^*\} \]

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

**RLWW-Automaton Accepting Palindromes with Marked Centers**

$$L_{\text{pal}, c} = \{ wcw^R \mid w \in \{a, b\}^* \}$$

### Example 1

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$$cc$</td>
<td>$L(M) = { wcw^R \mid w \in {a, b}^* }$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$L_C(M) = { wcw^R \mid w \in {a, b}^* }$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$L_{\text{hp}}(M) = { wcw^R \mid w \in {a, b}^* }$</td>
</tr>
<tr>
<td></td>
<td>$\delta(q_0, $c) = { \text{Accept} }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta(q_0, cxy) = { (q_1, \text{MVR}) }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for all $x \in {a, b}$, $y \in {a, b, c}$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta(q_1, aca) = { (q_2, c) }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta(q_1, bcb) = { (q_2, c) }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta(q_1, xyz) = { (q_1, \text{MVR}) }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for all $x, y \in {a, b}$, $z \in {a, b, c}$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta(q_2, u) = { \text{Restart} }$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for all $u \in \Gamma^3 \cup \Gamma^\leq 2 \cdot {$}$.</td>
<td></td>
</tr>
</tbody>
</table>
RLWW-Automaton Accepting Palindromes with Marked Centers $L_{\text{pal},c} = \{ wcw^R \mid w \in \{ a, b \}^* \}$

Example 1

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

(1) $\delta(q_0, cc$) = $\{ \text{Accept} \}$,
(2) $\delta(q_0, cxy) = \{ (q_1, \text{MVR}) \}$,
   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
### Definitions

**RLWW-Automaton Accepting Even Palindromes**

\[ L_{\text{pal}} = \{ ww^R \mid w \in \{a, b\}^* \} \]

### Example 2

\[ \Sigma = \{a, b\}, \quad \Gamma = \{a, b, A, B\}, \quad h(A) = a, \ h(B) = b, \]

\[ q_0 \rightarrow \text{a b B b b a $} \]
RLWW-Automaton Accepting Even Palindromes

\[ L_{pal} = \{ ww^R \mid w \in \{a, b\}^* \} \]

Example 2

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

\[ q_1 \]

\[ \angle a b B b b a $ \]
Definitions

**RLWW-Automaton Accepting Even Palindromes**

\[ L_{\text{pal}} = \{ w w^R \mid w \in \{a, b\}^* \} \]

**Example 2**

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

\[ \begin{array}{c}
q_1 \\
\$ a b B b b a \$
\end{array} \]
RLWW-Automaton Accepting Even Palindromes

$L_{\text{pal}} = \{ww^R | w \in \{a, b\}^*\}$

Example 2

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

$q_2$

$\& a B b a \$
RLWW-Automaton Accepting Even Palindromes

\[ L_{\text{pal}} = \{ ww^R \mid w \in \{a, b\}^* \} \]

Example 2

\[ \Sigma = \{a, b\}, \]
\[ \Gamma = \{a, b, A, B\}, \]
\[ h(A) = a, \quad h(B) = b, \]

\[ q_0 \quad \epsilon \quad a \quad B \quad b \quad a \quad \$ \]
RLWW-Automaton Accepting Even Palindromes

\[ L_{\text{pal}} = \{ ww^R \mid w \in \{a, b\}^* \} \]

Example 2

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

\[ q_1 \]
RLWW-Automaton Accepting Even Palindromes

$L_{pal} = \{ww^R \mid w \in \{a, b\}^*\}$

Example 2

$\Sigma = \{a, b\}$,
$\Gamma = \{a, b, A, B\}$,
$h(A) = a$, $h(B) = b$,
RLWW-Automaton Accepting Even Palindromes

\[ L_{\text{pal}} = \{ ww^R \mid w \in \{a, b\}^* \} \]

Example 2

\[ \Sigma = \{a, b\}, \]
\[ \Gamma = \{a, b, A, B\}, \]
\[ h(A) = a, \quad h(B) = b, \]

\[ q_1 \quad \epsilon \quad A \quad a \quad \$ \]
RLWW-Automaton Accepting Even Palindromes

$L_{\text{pal}} = \{ww^R \mid w \in \{a, b\}^*\}$

Example 2

\[
\begin{align*}
\Sigma &= \{a, b\}, \\
\Gamma &= \{a, b, A, B\}, \\
&\quad h(A) = a, h(B) = b,
\end{align*}
\]
Definitions

**RLWW-Automaton Accepting Even Palindromes**

\[ L_{pal} = \{ w w^R \mid w \in \{a, b\}^* \} \]

**Example 2**

\[ \Sigma = \{a, b\}, \]
\[ \Gamma = \{a, b, A, B\}, \]
\[ h(A) = a, \ h(B) = b, \]
\[ L(M) = \{\lambda\} \]
RLWW-Automaton Accepting Even Palindromes

\[ L_{\text{pal}} = \{ ww^R \mid w \in \{a, b\}^* \} \]

Example 2

\[ \Sigma = \{a, b\}, \quad \Gamma = \{a, b, A, B\}, \quad h(A) = a, h(B) = b, \]
\[ L(M) = \{\lambda\} \]
\[ L_C(M) = \{wAaw^R, wBbw^R \mid w \in \{a, b\}^*\} \cup \{\lambda\} \]
Definitions

RLWW-Automaton Accepting Even Palindromes

\( L_{\text{pal}} = \{ ww^R \mid w \in \{ a, b \}^* \} \)

Example 2

\[ \Sigma = \{ a, b \}, \]
\[ \Gamma = \{ a, b, A, B \}, \]
\[ h(A) = a, \quad h(B) = b, \]
\[ L(M) = \{ \lambda \} \]
\[ L_{\text{C}}(M) = \{ wAaw^R, wBbw^R \mid w \in \{ a, b \}^* \} \cup \{ \lambda \} \]
\[ L_{\text{hP}}(M) = \{ ww^R \mid w \in \{ a, b \}^* \} \]
RLWW-Automaton Accepting Even Palindromes

\[ L_{\text{pal}} = \{ w w^R \mid w \in \{a, b\}^* \} \]

Example 2

\[ \Sigma = \{a, b\}, \]
\[ \Gamma = \{a, b, A, B\}, \]
\[ h(A) = a, h(B) = b, \]
\[ L(M) = \{\lambda\} \]
\[ L_C(M) = \{wAaw^R, wBbw^R \mid w \in \{a, b\}^*\} \cup \{\lambda\} \]
\[ L_{hP}(M) = \{ww^R \mid w \in \{a, b\}^*\} \]

- only contextual rewritings = deletes only, at most 2 factors in one rewrite step
The Power of RLWW\((i)\)-automata

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

- $\mathcal{L}(\text{RLWW}(i))$ – 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 $\mathcal{L}(\text{RLWW}(1))$
The Power of $\mathcal{L}(\text{RLWW}(i))$-automata

- $\mathcal{L}(\text{RLWW}(i))$ – 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 $\mathcal{L}(\text{RLWW}(1))$
- a monotone computation: the places of rewriting do not increase

their distance from the right sentinel
The Power of RLWW(i)-automata

- $\mathcal{L}(\text{RLWW}(i))$ – 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 $\mathcal{L}(\text{RLWW}(1))$
- 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

- \( \mathcal{L}(\text{RLWW}(i)) \) – 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 \( \mathcal{L}(\text{RLWW}(1)) \)
- 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.
an \( \text{hRLWW}(i) \)-automaton is **basically correctness preserving** if \( u \Rightarrow^*_{c_M} v \) and \( u \in L_C(M) \) induce that \( v \in L_C(M) \), and therewith \( h(v) \in L_{hP}(M) \) and \( (h(v), v) \in L_A(M) \).

**Fact:** each deterministic \( \text{hRLWW}(i) \)-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

- $L_C(M) = L_C(M_{scf})$,
- $u \Rightarrow^* M v$ implies $u \Rightarrow^* M_{scf} v$, 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 $X \in \{hRLWW(1), hRLWWD(1), hRLWWC(1)\}$. Then
- $\text{LRR} = \mathcal{L}_C(\text{scf-det-syn-}X)$ and
- $\text{CFL} = \mathcal{L}_{hP}(\text{scf-det-syn-}X)$. 
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) \( \{a^k\} \not\in \mathcal{L}_C((k - 1)\text{-scf-hRLWW}) \cup \mathcal{L}_hP((k - 1)\text{-scf-hRLWW}) \).
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) $L_2(j, k) \in \mathcal{L}_C(k\text{-scf-fin}(1)\text{-det-mon-RLWC}(j))$.

(b) $L_2(j, k) \not\in \mathcal{L}_C(k\text{-scf-hRLWW}(j')) \cup \mathcal{L}_{hP}(k\text{-scf-hRLWW}(j'))$ 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) $L_3(j, k) \in \mathcal{L}_C(k$-scf-fin$(j + j^2)$-det-mon$(j)$-RLWC$(j)$).

(b) $L_3(j, k) \not\in \mathcal{L}_C(k$-scf-mon$(j')$-hRLWW$(j)$) $\cup \mathcal{L}_{hp}(k$-scf-mon$(j')$-hRLWW$(j)$) for any $j' < j$.

(c) $L_3(j, k) \not\in \mathcal{L}_C(k$-scf-hRLWW$(j')$) $\cup \mathcal{L}_{hp}(k$-scf-hRLWW$(j')$) for any $j' < j$. 
Hierarchy of Contextually Transparent Language Classes

\[ \text{CTL}(i) = \text{the class of h-proper languages accepted by } \text{hRLWW}(i)-\text{automata that are} \]

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

\[ \text{CTL} \rightarrow \text{CSL} \]

\[ \vdots \]

\[ \text{CTL}(3) \]

\[ \text{CTL}(2) \]

\[ \text{CTL}(1) = \text{CFL} \]

\[ \text{CTL}(3) \]

\[ \text{CTL}(2) \]

\[ \text{CTL}(1) = \text{CFLA} \]

\[ \text{CTL} \rightarrow \text{CTLA} \]

\[ \vdots \]

\[ \text{CTLA}(3) \]

\[ \text{CTLA}(2) \]

\[ \text{CTLA}(1) = \text{CFLA} \]
\(\subseteq\): easy

\(\not\subseteq\): \(L_e = \{ a^{2^n} \mid n \geq 0 \} \not\subseteq\) CTL by contradiction

- if \(a^{2^n} \in L_{hP}(M)\) for some \(k\)-hRLWW(i)-automaton, then \(a^{2^n} = h(w)\) for some \(w \in L_C(M)\) 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 \(h(w') \in L_{hP}(M)\)
- for sufficiently large \(n\), the length of \(h(w')\) cannot be a power of 2 \(\Rightarrow h(w') \not\in L_{hP}(M)\) – 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 $h$-RLWW($i$)-automata that are

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

\[ \begin{align*}
1-\text{CTL} & \rightarrow 2-\text{CTL} \rightarrow 3-\text{CTL} \rightarrow \cdots \rightarrow \text{CTL} \rightarrow \text{CSL} \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
& \vdots \quad \vdots \quad \vdots \quad \cdots \quad \vdots \quad \vdots \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
1-\text{CTL}(3) & \rightarrow 2-\text{CTL}(3) \rightarrow 3-\text{CTL}(3) \rightarrow \cdots \rightarrow \text{CTL}(3) \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
& \vdots \quad \vdots \quad \vdots \quad \cdots \quad \vdots \quad \vdots \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
1-\text{CTL}(2) & \rightarrow 2-\text{CTL}(2) \rightarrow 3-\text{CTL}(2) \rightarrow \cdots \rightarrow \text{CTL}(2) \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
& \vdots \quad \vdots \quad \vdots \quad \cdots \quad \vdots \quad \vdots \\
& \uparrow \quad \uparrow \quad \uparrow \quad \cdots \quad \uparrow \quad \uparrow \\
1-\text{CTL}(1) & \rightarrow 2-\text{CTL}(1) \rightarrow 3-\text{CTL}(1) \rightarrow \cdots \rightarrow \text{CTL}(1) = \text{CFL}
\end{align*} \]
A Refinement With Respect to the Window Size

Lexicalized Analyses

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

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

1-CTLA $\rightarrow$ 2-CTLA $\rightarrow$ 3-CTLA $\rightarrow$ · · · $\rightarrow$ CTLA

↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑∪
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.

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 (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 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.
Conclusions (1)

- \textit{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.

- \textbf{Conjecture:} The class 12-\textit{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,
Phrase-structure grammars:
- categories bound to individual constituents (parts of a sentence) ⇒ 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) ⇒
    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) ⇒ 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 $h\text{RLWWC}(i)$-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!