A Measure for The Degree of Nondeterminism of Context-Free Languages[1]

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

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

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

Conference on Implementation and Application of Automata, 2007

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

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...
  } additional information is inserted into the input sentence – auxiliary symbols

  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - "make it as efficient (deterministic) as possible"
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...
  syntactic analysis

  additional information is inserted into the input sentence – auxiliary symbols
- syntactic analysis
  - often nondeterministic – high complexity
  - “make it as efficient (deterministic) as possible”
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...

  additional information is inserted into the input sentence – auxiliary symbols

  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - "make it as efficient (deterministic) as possible"
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases
  a dictionary lookup
  morphological analysis
  disambiguation, ...
  syntactic analysis

  additional information is inserted into the input sentence – auxiliary symbols
- syntactic analysis
  - often nondeterministic – high complexity
  - “make it as efficient (deterministic) as possible”
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases
  a dictionary lookup
  morphological analysis
  disambiguation, ...
  } additional information is inserted into the input sentence – auxiliary symbols
  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - "make it as efficient (deterministic) as possible"
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...
  } additional information is inserted into the input sentence – auxiliary symbols

  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - “make it as efficient (deterministic) as possible”
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...
  } additional information is inserted into the input sentence – auxiliary symbols

  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - "make it as efficient (deterministic) as possible"
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

An analysis of a sentence

- it is very complex and has several phases

  a dictionary lookup
  morphological analysis
  disambiguation, ...

  } additional information is inserted into the input sentence – auxiliary symbols

  syntactic analysis
- syntactic analysis
  - often nondeterministic – high complexity
  - "make it as efficient (deterministic) as possible"
  - many different models are used – among others restarting automata
- Questions:
  1. How many symbols must be added in order to get a subsequent analysis deterministic?
  2. Do we need many different auxiliary symbols?

Our model

an artificial example:

The language $L_e = \{ww^R \mid w \in \{a, b\}^*\}$ is not deterministic context-free, i.e. deterministic for a pushdown automaton, but an insertion of a single symbol in the middle of an input word – $w\#w^R$ – makes its parsing simple.

- categories are added and disambiguated by a nondeterministic process followed by
- a deterministic analysis by reduction modelled by a restarting automaton

Outline

1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
2. Lexicalized restarting automata
   - Deterministic and lexicalized restarting automata
   - Monotone restarting automata
   - How to measure nondeterminism
   - Word-alphabet expansion hierarchy
3. Conclusions

Outline

1. Restarting automaton
   - Definition
   - Meta-instructions
   - Languages defined by restarting automata
2. Lexicalized restarting automata
   - Deterministic and lexicalized restarting automata
   - Monotone restarting automata
   - How to measure nondeterminism
   - Word-alphabet expansion hierarchy
3. Conclusions

Outline

Restarting automaton
a tool for modeling **analysis by reduction**

Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

Jane works.

- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton

a tool for modeling **analysis by reduction**

Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

Jane works.

- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton
a tool for modeling **analysis by reduction**

Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

Jane works.

- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton

a tool for modeling **analysis by reduction**

Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

Jane works.

- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton
a tool for modeling **analysis by reduction**

Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

Jane works.

- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton

a tool for modeling **analysis by reduction**

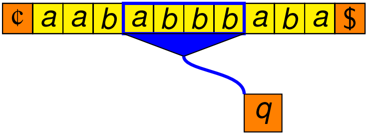
Martin, Peter and Jane work very slowly.

Peter and Jane work very slowly.

Jane works very slowly.

Jane works slowly.

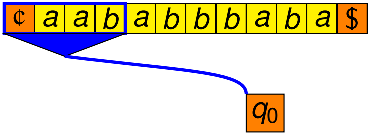
Jane works.



- restarting automaton consists of:
  - finite control
  - read/write window of fixed size
  - operations: move right, rewrite, restart, accept

Restarting automaton

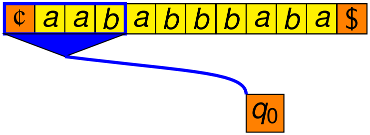
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

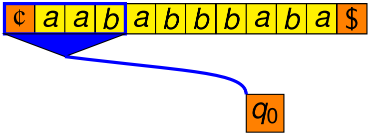
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

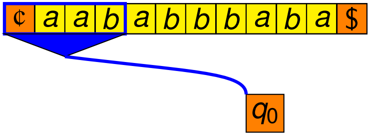
a tool for modeling **analysis by reduction**



- computation
    - starts on the left end
    - consists of cycles = parts between two restarts
    - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

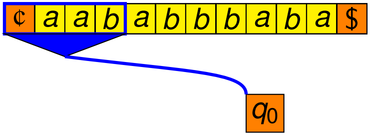
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

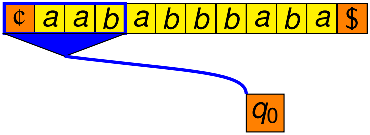
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

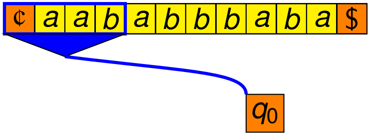
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

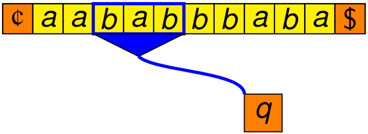
a tool for modeling **analysis by reduction**



- computation
  - starts on the left end
  - consists of cycles = parts between two restarts
  - in one cycle exactly one rewriting of the content of its window must occur (local change), it must shorten the tape
- this model emphasized locality of one reduction step
- even for CFL recognition there are needed additional (non-input) working symbols

Restarting automaton

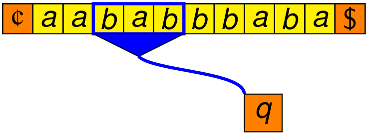
briefly RRWW-automaton



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

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

Restarting automaton
briefly RRWW-automaton

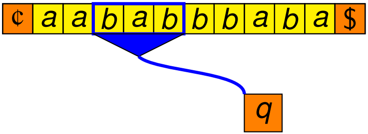


$M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$:

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

Restarting automaton

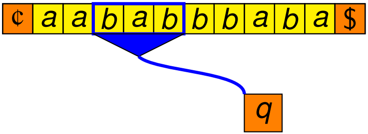
briefly RRWW-automaton



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

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

Restarting automaton
briefly RRWW-automaton

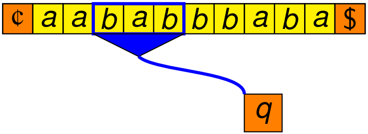


$M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$:

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

Restarting automaton

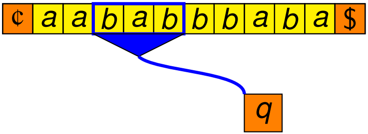
briefly RRWW-automaton



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

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

Restarting automaton
briefly RRWW-automaton



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

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

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

$w_1 \vdash^c_M w_2 \vdash^c_M \cdots \vdash^c_M w_{n-1} \vdash^c_M w_n$

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

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

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

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

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

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

$w_1 =$

$w_2 =$

$w_3 =$

$\dots$

$w_{n-1} =$

$w_n =$

$$w_1 \vdash^c_M w_2 \vdash^c_M \cdots \vdash^c_M w_{n-1} \vdash^c_M w_n$$

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

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

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

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

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

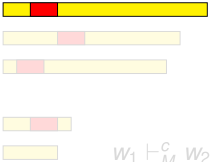
Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

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



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

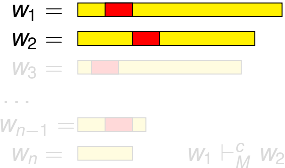
Note: if w_n is accepted by M then all $w_1, \dots, w_n$ are accepted by M.

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).



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

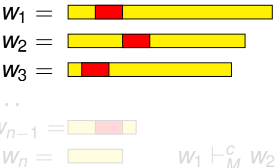
Note: if w_n is accepted by M then all $w_1, \ldots, w_n$ are accepted by M.

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).



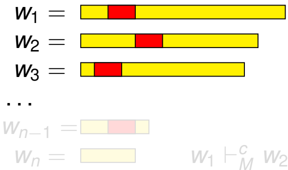
$w_1 \vdash_M^c w_2 \vdash_M^c \cdots \vdash_M^c w_{n-1} \vdash_M^c w_n$

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

Computation of a restarting automaton
Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).



$w_1 \vdash^c_M w_2 \vdash^c_M \cdots \vdash^c_M w_{n-1} \vdash^c_M w_n$

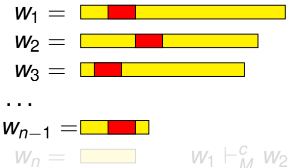
Note: if w_n is accepted by M then all $w_1, \ldots, w_n$ are accepted by M.

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).



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

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

Computation of a restarting automaton

Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).

$w_1 =$

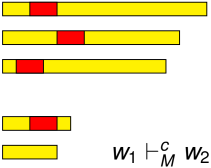
$w_2 =$

$w_3 =$

$\dots$

$w_{n-1} =$

$w_n =$



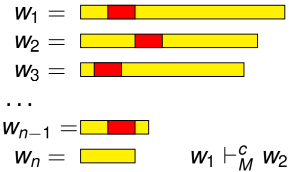
$$w_1 \vdash_M^c w_2 \vdash_M^c \cdots \vdash_M^c w_{n-1} \vdash_M^c w_n$$

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

Computation of a restarting automaton
Cycles, tails

a cycle the part of an computation between two restarting configurations

a tail the part of an computation after the last restart configuration and halt (accepting or rejecting).



$$w_1 \vdash^c_M w_2 \vdash^c_M \cdots \vdash^c_M w_{n-1} \vdash^c_M w_n$$

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

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word *abb#bba*:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a,b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word *abb#bba*:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Example

- A restarting automaton for the language

$$L'_e = \{w\#w^R | w \in \{a, b\}^*\}$$

- Sample computation on the word $abb\#bba$:
  - $¢abb\#bba\$$
  - $¢ab\#ba\$$
  - $¢a\#a\$$
  - $¢\#\$$
  - Accept
- It can be done by a deterministic restarting automaton.

Meta-instructions

A more convenient representation

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

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

$$w = \boxed{w_1}\boxed{u}\boxed{w_2}$$

$$w' = \boxed{w_1}\boxed{v}\boxed{w_2}$$

(E, Accept) an accepting meta-instruction

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

Meta-instructions

A more convenient representation

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

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

$$w = \boxed{w_1}\boxed{u}\boxed{w_2}$$
$$\top$$
$$w' = \boxed{w_1}\boxed{v}\boxed{w_2}$$

(E, Accept) an accepting meta-instruction

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

Languages defined by a RRWW-automaton

- A word w is accepted by M if there exists a computation which starts by the initial configuration $q_0¢w\$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that w is an expanded version of v.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of M – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if v has an expanded version u such that $u \in L_C(M)$.

Languages defined by a RRWW-automaton

- A word *w* is accepted by *M* if there exists a computation which starts by the initial configuration $q_0¢w\$$ and ends by an accepting configuration.
- The set of *all* words accepted by *M* is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton *M*.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by *M*
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that *w* is an expanded version of *v*.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of *M* – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if *v* has an expanded version *u* such that $u \in L_C(M)$.

Languages defined by a RRWW-automaton

- A word w is accepted by M if there exists a computation which starts by the initial configuration $q_0 ¢ w \$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that w is an expanded version of v.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of M – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if v has an expanded version u such that $u \in L_C(M)$.

$L(M)$ $L_P(M)$ $L_C(M)$

Languages defined by a RRWW-automaton

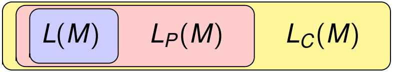
- A word w is accepted by M if there exists a computation which starts by the initial configuration $q_0 ¢ w \$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that w is an expanded version of v.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of M – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if v has an expanded version u such that $u \in L_C(M)$.

Languages defined by a RRWW-automaton

- A word w is accepted by M if there exists a computation which starts by the initial configuration $q_0¢w\$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that w is an expanded version of v.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of M – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if v has an expanded version u such that $u \in L_C(M)$.

Languages defined by a RRWW-automaton

- A word w is accepted by M if there exists a computation which starts by the initial configuration $q_0¢w\$$ and ends by an accepting configuration.
- The set of *all* words accepted by M is denoted as $L_C(M)$ and it is called the characteristic language accepted by the RRWW-automaton M.
- $L(M) = L_C(M) \cap \Sigma^*$ denotes the *input language* accepted by M
- By Pr^Σ we denote the projection from Γ^* onto Σ^* which leaves out all the auxiliary symbols.
- If $v = \mathrm{Pr}^\Sigma(w)$ we say that w is an expanded version of v.
- $L_P(M) := \mathrm{Pr}^\Sigma(L_C(M))$ denotes the proper language of M – a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if v has an expanded version u such that $u \in L_C(M)$.



Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

abbCbba
abCba
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Sample restarting automaton

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

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

abbCbba
abCba
aCa
C
Accept

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

Sample restarting automaton

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

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

$abbCbba$
$abCba$
aCa
C
Accept

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

Deterministic and lexicalized RRWW-automata

An RRWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is

- deterministic if its transition relation is deterministic (i.e. after transforming the meta-instructions into a low level instructions we get a deterministic automaton).

Fact

If M is a deterministic RRWW*-automaton, then the membership problem for the language $L_C(M)$ is solvable in linear time.*

- lexicalized if it is *deterministic* and there exists a constant $j \in \mathbb{N}_+$ such that M rejects immediately all words containing a subsequence of non-input symbols of length greater than j.

Deterministic and lexicalized RRWW-automata

An RRWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is

- deterministic if its transition relation is deterministic (i.e. after transforming the meta-instructions into a low level instructions we get a deterministic automaton).

Fact

If M is a deterministic RRWW*-automaton, then the membership problem for the language $L_C(M)$ is solvable in linear time.*

- lexicalized if it is *deterministic* and there exists a constant $j \in \mathbb{N}_+$ such that M rejects immediately all words containing a subsequence of non-input symbols of length greater than j.

Deterministic and lexicalized RRWW-automata

An RRWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is

- deterministic if its transition relation is deterministic (i.e. after transforming the meta-instructions into a low level instructions we get a deterministic automaton).

Fact

If M is a deterministic RRWW*-automaton, then the membership problem for the language $L_C(M)$ is solvable in linear time.*

- lexicalized if it is *deterministic* and there exists a constant $j \in \mathbb{N}_+$ such that M rejects immediately all words containing a subsequence of non-input symbols of length greater than j.

Deterministic and lexicalized RRWW-automata

An RRWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is

- deterministic if its transition relation is deterministic (i.e. after transforming the meta-instructions into a low level instructions we get a deterministic automaton).

Fact

If M is a deterministic RRWW*-automaton, then the membership problem for the language $L_C(M)$ is solvable in linear time.*

- lexicalized if it is *deterministic* and there exists a constant $j \in \mathbb{N}_+$ such that M rejects immediately all words containing a subsequence of non-input symbols of length greater than j.

Lexicalized RRWW-automata

- If M is a lexicalized RRWW-automaton, and if w is an extended version of an input word $v = \mathrm{Pr}^{\Sigma}(w)$ such that w is not immediately rejected by M, then $|w| \leq (j+1) \cdot |v| + j$ for some constant $j > 0$.

Corollary

If M is a lexicalized RRWW-automaton, then the proper language $L_P(M)$ is context-sensitive.

Lexicalized RRWW-automata

- If M is a lexicalized RRWW-automaton, and if w is an extended version of an input word $v = \mathrm{Pr}^{\Sigma}(w)$ such that w is not immediately rejected by M, then $|w| \leq (j+1) \cdot |v| + j$ for some constant $j > 0$.

Corollary

If M is a lexicalized RRWW-automaton, then the proper language $L_P(M)$ is context-sensitive.

Lexicalized RRWW-automata

- If M is a lexicalized RRWW-automaton, and if w is an extended version of an input word $v = \mathrm{Pr}^{\Sigma}(w)$ such that w is not immediately rejected by M, then $|w| \leq (j+1) \cdot |v| + j$ for some constant $j > 0$.

Corollary

If M is a lexicalized RRWW*-automaton, then the proper language $L_P(M)$ is context-sensitive.*

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.

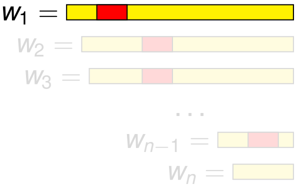
- monotone RRWW-automaton – all its computations are monotone

Theorem

CFL = $\mathcal{L}_P$(mon-lex-RRWW).

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



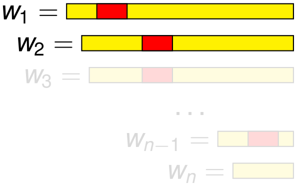
- monotone RRWW-automaton – all its computations are monotone

Theorem

CFL $= \mathcal{L}_P$(mon-lex-RRWW).

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



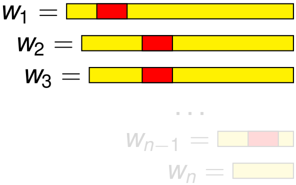
- monotone RRWW-automaton – all its computations are monotone

Theorem

CFL = $\mathcal{L}_P$(mon-lex-RRWW).

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



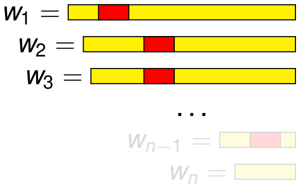
- monotone RRWW-automaton – all its computations are monotone

Theorem

$CFL = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



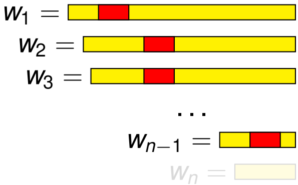
- monotone RRWW-automaton – all its computations are monotone

Theorem

$\mathrm{CFL} = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



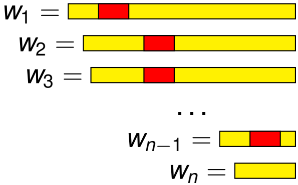
- monotone RRWW-automaton – all its computations are monotone

Theorem

$\mathrm{CFL} = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



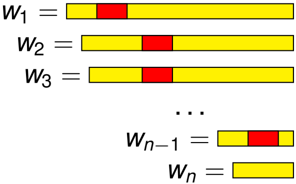
- monotone RRWW-automaton – all its computations are monotone

Theorem

$\mathrm{CFL} = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



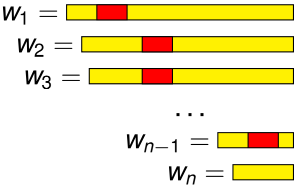
- monotone RRWW-automaton – all its computations are monotone

Theorem

$\mathrm{CFL} = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



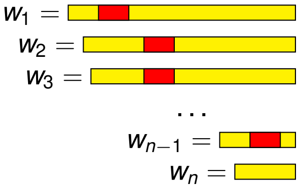
- monotone RRWW-automaton – all its computations are monotone

Theorem

$\mathrm{CFL} = \mathcal{L}_P(\text{mon-lex-RRWW})$.

Monotone RRWW-automata

- monotone computation – the places of rewriting in cycles do not increase their distance from the right sentinel. Tails are not considered here.



- monotone RRWW-automaton – all its computations are monotone

Theorem

CFL $= \mathcal{L}_P$(mon-lex-RRWW).

Word and word-alphabet expansion

- M has word-expansion m (denoted by $\mathsf{W}(m)$-) if each word from $L_C(M)$ contains at most m occurrences of auxiliary symbols.
- M has word-alphabet-expansion (m, j) (denoted by $\mathsf{WA}(m, j)$-) if M has word-expansion m, and Γ contains at most j different auxiliary symbols ($|\Gamma \setminus \Sigma| \leq j$).

Theorem

If M is a $\mathsf{W}(m)$-RRWW-automaton for some constant $m \geq 0$, then the membership problem for the language $L_P(M)$ is solvable in deterministic polynomial time.

$O(n^{m+1})$

Word and word-alphabet expansion

- M has word-expansion m (denoted by $\mathrm{W}(m)$-) if each word from $L_C(M)$ contains at most m occurrences of auxiliary symbols.
- M has word-alphabet-expansion (m, j) (denoted by $\mathrm{WA}(m, j)$-) if M has word-expansion m, and Γ contains at most j different auxiliary symbols ($|\Gamma \setminus \Sigma| \leq j$).

Theorem

If M is a $\mathrm{W}(m)$-RRWW-automaton for some constant $m \geq 0$, then the membership problem for the language $L_P(M)$ is solvable in deterministic polynomial time.

$O(n^{m+1})$

Word and word-alphabet expansion

- M has word-expansion m (denoted by $\mathrm{W}(m)$-) if each word from $L_C(M)$ contains at most m occurrences of auxiliary symbols.
- M has word-alphabet-expansion (m, j) (denoted by $\mathrm{WA}(m, j)$-) if M has word-expansion m, and Γ contains at most j different auxiliary symbols ($|\Gamma \setminus \Sigma| \leq j$).

Theorem

If M is a $\mathrm{W}(m)$-RRWW-automaton for some constant $m \geq 0$, then the membership problem for the language $L_P(M)$ is solvable in deterministic polynomial time.

$O(n^{m+1})$

Word and word-alphabet expansion

- M has word-expansion m (denoted by $\mathrm{W}(m)$-) if each word from $L_C(M)$ contains at most m occurrences of auxiliary symbols.
- M has word-alphabet-expansion (m, j) (denoted by $\mathrm{WA}(m, j)$-) if M has word-expansion m, and Γ contains at most j different auxiliary symbols ($|\Gamma \setminus \Sigma| \leq j$).

Theorem

If M is a $\mathrm{W}(m)$-RRWW-automaton for some constant $m \geq 0$, then the membership problem for the language $L_P(M)$ is solvable in deterministic polynomial time.

$O(n^{m+1})$

Theorem

DCFL $= \mathcal{L}_{\mathrm{P}}$(W(0)-mon-RRWW) $= \mathcal{L}_C$(W(0)-mon-RRWW).

- let $L_3 := \bigcup_{1 \leq k \leq 3} \{ a^n(b^k)^n \mid n \geq 0 \}$ over $\Sigma_0 := \{a, b\}$.

Lemma

$L_3 \in \mathcal{L}_{\mathrm{P}}(\text{WA}(1,1)\text{-mon-RRWW}) \setminus \mathcal{L}_P(\text{W}(0)\text{-RRWW})$.

- L_3 is the proper language of the automaton with the following meta-instructions (C is an auxiliary symbol):

(1)	$(¢ \cdot a^*, ab \to \lambda, b^* \cdot \$)$,	(3)	$(¢ \cdot aCa^*, abbb \to \lambda, b^+ \cdot \$)$,
(2)	$(¢ \cdot Ca^*, abb \to \lambda, b^* \cdot \$)$,	(4)	$(¢ \cdot (C + aCbbb + \lambda) \cdot \$, \text{Accept})$.

Theorem

DCFL $= \mathcal{L}_{\mathrm{P}}$(W(0)-mon-RRWW) $= \mathcal{L}_C$(W(0)-mon-RRWW).

- let $L_3 := \bigcup_{1 \leq k \leq 3} \{\, a^n (b^k)^n \mid n \geq 0 \,\}$ over $\Sigma_0 := \{a, b\}$.

Lemma

$L_3 \in \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,1)\text{-mon-RRWW}) \smallsetminus \mathcal{L}_P(\mathrm{W}(0)\text{-RRWW})$.

- L_3 is the proper language of the automaton with the following meta-instructions (C is an auxiliary symbol):

(1)	$(\text{¢} \cdot a^*, ab \to \lambda, b^* \cdot \$)$,	(3)	$(\text{¢} \cdot aCa^*, abbb \to \lambda, b^+ \cdot \$)$,
(2)	$(\text{¢} \cdot Ca^*, abb \to \lambda, b^* \cdot \$)$,	(4)	$(\text{¢} \cdot (C + aCbbb + \lambda) \cdot \$, \mathrm{Accept})$.

Theorem

DCFL $= \mathcal{L}_{\mathrm{P}}(\text{W(0)-mon-RRWW}) = \mathcal{L}_C(\text{W(0)-mon-RRWW})$.

- let $L_3 := \bigcup_{1 \leq k \leq 3} \{\, a^n (b^k)^n \mid n \geq 0 \,\}$ over $\Sigma_0 := \{a, b\}$.

Lemma

$L_3 \in \mathcal{L}_{\mathrm{P}}(\text{WA}(1,1)\text{-mon-RRWW}) \smallsetminus \mathcal{L}_P(\text{W(0)-RRWW})$.

- L_3 is the proper language of the automaton with the following meta-instructions (C is an auxiliary symbol):

$$
\begin{array}{llll}
(1) & (\text{¢} \cdot a^*, ab \to \lambda, b^* \cdot \$), & (3) & (\text{¢} \cdot aCa^*, abbb \to \lambda, b^+ \cdot \$), \\
(2) & (\text{¢} \cdot Ca^*, abb \to \lambda, b^* \cdot \$), & (4) & (\text{¢} \cdot (C + aCbbb + \lambda) \cdot \$, \text{Accept}).
\end{array}
$$

- Let L_{pal} denote the language of palindromes $L_{\mathrm{pal}} := \{\, w \in \Sigma_0^* \mid w = w^R \,\}$.

Lemma

L_{pal} belongs to the class $\mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,1)\text{-mon-RRWW})$, but it is not contained in the class $\mathcal{L}_{\mathrm{P}}(\mathrm{W}(0)\text{-RRWW})$.

- For $j \in \mathbb{N}_+$, let $L_j := \bigcup_{1 \le i \le j} \{\, a^n (b^i)^n \mid n \ge 0 \,\}$, and

$$L_p(1,j) := \{\, ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \ge |w| \ge 0, \text{ and } v \in L_j \}.$$

Based on the construction of M_{pal} we can construct a monotone RRWW-automaton $M^p_{1,j}$ that has word-alphabet-expansion $(1,j)$ and that satisfies $L_{\mathrm{P}}(M^p_{1,j}) = L_p(1,j)$.

Lemma

For all $j \in \mathbb{N}_+$, $L_p(1,j) \in \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,j)\text{-mon-RRWW})$, while for $j > 1$, $L_p(1,j) \notin \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,j-1)\text{-RRWW})$.

- Let L_{pal} denote the language of palindromes
  $L_{\mathrm{pal}} := \{ w \in \Sigma_0^* \mid w = w^R \}$.

Lemma

L_{pal} belongs to the class $\mathcal{L}_{\mathrm{P}}$(WA(1, 1)-mon-RRWW), but it is not contained in the class $\mathcal{L}_{\mathrm{P}}$(W(0)-RRWW).

- For $j \in \mathbb{N}_+$, let $L_j := \bigcup_{1 \le i \le j} \{ a^n (b^i)^n \mid n \ge 0 \}$, and

  $$L_p(1, j) := \{ ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \ge |w| \ge 0, \text{ and } v \in L_j \}.$$

  Based on the construction of M_{pal} we can construct a monotone RRWW-automaton $M^p_{1,j}$ that has word-alphabet-expansion $(1, j)$ and that satisfies $L_{\mathrm{P}}(M^p_{1,j}) = L_p(1, j)$.

Lemma

For all $j \in \mathbb{N}_+$, $L_p(1, j) \in \mathcal{L}_{\mathrm{P}}$(WA(1, j)-mon-RRWW), while for $j > 1$, $L_p(1, j) \notin \mathcal{L}_{\mathrm{P}}$(WA(1, $j-1$)-RRWW).

- Let L_{pal} denote the language of palindromes $L_{\mathrm{pal}} := \{ w \in \Sigma_0^* \mid w = w^R \}$.

Lemma

L_{pal} belongs to the class $\mathcal{L}_{\mathrm{P}}(\mathsf{WA}(1,1)\text{-mon-RRWW})$, but it is not contained in the class $\mathcal{L}_{\mathrm{P}}(\mathsf{W}(0)\text{-RRWW})$.

- For $j \in \mathbb{N}_+$, let $L_j := \bigcup_{1 \le i \le j} \{ a^n (b^i)^n \mid n \ge 0 \}$, and

$$L_p(1,j) := \{ ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \ge |w| \ge 0,\ \text{and } v \in L_j \}.$$

Based on the construction of M_{pal} we can construct a monotone RRWW-automaton $M_{1,j}^p$ that has word-alphabet-expansion $(1,j)$ and that satisfies $L_{\mathrm{P}}(M_{1,j}^p) = L_p(1,j)$.

Lemma

For all $j \in \mathbb{N}_+$, $L_p(1,j) \in \mathcal{L}_{\mathrm{P}}(\mathsf{WA}(1,j)\text{-mon-RRWW})$, while for $j > 1$, $L_p(1,j) \notin \mathcal{L}_{\mathrm{P}}(\mathsf{WA}(1,j-1)\text{-RRWW})$.

- Let L_{pal} denote the language of palindromes
  $L_{\mathrm{pal}} := \{\, w \in \Sigma_0^* \mid w = w^R \,\}$.

Lemma

L_{pal} belongs to the class $\mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,1)\text{-mon-RRWW})$, but it is not contained in the class $\mathcal{L}_{\mathrm{P}}(\mathrm{W}(0)\text{-RRWW})$.

- For $j \in \mathbb{N}_+$, let $L_j := \bigcup_{1 \le i \le j} \{\, a^n (b^i)^n \mid n \ge 0 \,\}$, and

  $$L_p(1,j) := \{\, ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \ge |w| \ge 0,\ \text{and } v \in L_j \}.$$

  Based on the construction of M_{pal} we can construct a monotone RRWW-automaton $M^p_{1,j}$ that has word-alphabet-expansion $(1,j)$ and that satisfies $L_{\mathrm{P}}(M^p_{1,j}) = L_p(1,j)$.

Lemma

For all $j \in \mathbb{N}_+$, $L_p(1,j) \in \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,j)\text{-mon-RRWW})$, while for $j > 1$, $L_p(1,j) \notin \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(1,j-1)\text{-RRWW})$.

$$L_p(1,j) := \{\, ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \geq |w| \geq 0, \text{ and } v \in L_j \}.$$

Now, for all $m, j \in \mathbb{N}_+$, let $L_p(m,j) := L_p(1,j) \cdot (\{d\} \cdot L_{\mathrm{pal}})^{m-1}$.

Lemma

(a) *For all* $m, j \geq 1$, $L_p(m,j) \in \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(m,j)\text{-mon-RRWW})$.

(b) *For all* $m, j \geq 1$, $L_p(m,j) \notin \mathcal{L}_{\mathrm{P}}(\mathrm{W}(m-1)\text{-RRWW})$.

(c) *For all* $m \geq 1$ *and* $j \geq 2$,
$L_p(m,j) \notin \mathcal{L}_{\mathrm{P}}(\mathrm{WA}(m,j-1)\text{-RRWW})$.

$$L_p(1,j) := \{\, ucvcw \mid uw \in L_{\mathrm{pal}},\ |u| \geq |w| \geq 0,\ \text{and } v \in L_j \}.$$

$$L_p(m,j) := L_p(1,j) \cdot (\{d\} \cdot L_{\mathrm{pal}})^{m-1}.$$

In processing a word from $L_p(m,j)$, a lexicalized RRWW-automaton M' must be able to distinguish between the j possible values for the parameter i in the factor $c \cdot a^n \cdot b^{i \cdot n} \cdot c$ inserted within the first of the above palindromes. As it scans its tape strictly from left to right, it only has the first of the above-mentioned auxiliary symbols to encode the correct value. It follows that j different auxiliary symbols must be available to M'.

Word-alphabet expansion hierarchy

Theorem

For all $\mathrm{X} \in \{\mathrm{RRWW}, \text{mon-RRWW}\}$*, we have the following proper inclusions:*

(a) $\mathcal{L}_P(\mathrm{W}(m)\text{-X}) \subset \mathcal{L}_P(\mathrm{W}(m+1)\text{-X})$ *for all* $m \geq 0$.

(b) $\mathcal{L}_P(\mathrm{WA}(m,j)\text{-X}) \subset \mathcal{L}_P(\mathrm{WA}(m+1,j)\text{-X})$ *for all* $m \geq 0, j \geq 1$.

(b) $\mathcal{L}_P(\mathrm{WA}(m,j)\text{-X}) \subset \mathcal{L}_P(\mathrm{WA}(m,j+1)\text{-X})$ *for all* $m, j \geq 1$.

Corollary

$\bigcup_{m\geq 0} \mathcal{L}_P(\mathrm{W}(m)\text{-mon-RRWW}) \subset \mathcal{L}_P(\text{lex-mon-RRWW}) = \mathrm{CFL}$.

$\bigcup_{m\geq 0} \mathcal{L}_P(\mathrm{W}(m)\text{-RRWW}) \subset \mathcal{L}_P(\text{lex-RRWW})$.

Word-alphabet expansion hierarchy

Theorem

For all $\mathsf{X} \in \{\mathsf{RRWW}, \text{mon-}\mathsf{RRWW}\}$*, we have the following proper inclusions:*

(a) $\mathcal{L}_P(\mathsf{W}(m)\text{-}\mathsf{X}) \subset \mathcal{L}_P(\mathsf{W}(m+1)\text{-}\mathsf{X})$ *for all* $m \geq 0$.

(b) $\mathcal{L}_P(\mathsf{WA}(m,j)\text{-}\mathsf{X}) \subset \mathcal{L}_P(\mathsf{WA}(m+1,j)\text{-}\mathsf{X})$ *for all* $m \geq 0, j \geq 1$.

(b) $\mathcal{L}_P(\mathsf{WA}(m,j)\text{-}\mathsf{X}) \subset \mathcal{L}_P(\mathsf{WA}(m,j+1)\text{-}\mathsf{X})$ *for all* $m, j \geq 1$.

Corollary

$\bigcup_{m\geq 0} \mathcal{L}_P(\mathsf{W}(m)\text{-mon-}\mathsf{RRWW}) \subset \mathcal{L}_P(\text{lex-mon-}\mathsf{RRWW}) = \mathsf{CFL}$.

$\bigcup_{m\geq 0} \mathcal{L}_P(\mathsf{W}(m)\text{-}\mathsf{RRWW}) \subset \mathcal{L}_P(\text{lex-}\mathsf{RRWW})$.

Conclusions

- The degree of word-expansion and the word-alphabet-expansion are new measures for the degree of nondeterminism for proper languages of restarting automata.
- we have obtained infinite hierarchies of language classes for monotone and for non-monotone RRWW-automata that are lexicalized.
- In the monotone case these classes form an infinite hierarchy between DCFL and CFL.
- Moreover, for each finite degree m of word-expansion, the number of available different auxiliary symbols yields an infinite hierarchy within $\mathcal{L}_{\mathrm{P}}(\mathrm{W}(m)\text{(-mon)-RRWW})$.
- Any lexicalized RRWW-automaton has word-expansion that is bounded from above by a linear function. It remains to study those languages that are obtained as proper languages of lexicalized RRWW-automata for which the word-expansion is non-constant, but bounded from above by a slowly growing function.

Conclusions

- The degree of word-expansion and the word-alphabet-expansion are new measures for the degree of nondeterminism for proper languages of restarting automata.
- we have obtained infinite hierarchies of language classes for monotone and for non-monotone RRWW-automata that are lexicalized.
- In the monotone case these classes form an infinite hierarchy between DCFL and CFL.
- Moreover, for each finite degree m of word-expansion, the number of available different auxiliary symbols yields an infinite hierarchy within $\mathcal{L}_{\mathrm{P}}(\mathrm{W}(m)$(-mon)-RRWW).
- Any lexicalized RRWW-automaton has word-expansion that is bounded from above by a linear function. It remains to study those languages that are obtained as proper languages of lexicalized RRWW-automata for which the word-expansion is non-constant, but bounded from above by a slowly growing function.

Conclusions

- The degree of word-expansion and the word-alphabet-expansion are new measures for the degree of nondeterminism for proper languages of restarting automata.
- we have obtained infinite hierarchies of language classes for monotone and for non-monotone RRWW-automata that are lexicalized.
- In the monotone case these classes form an infinite hierarchy between DCFL and CFL.
- Moreover, for each finite degree m of word-expansion, the number of available different auxiliary symbols yields an infinite hierarchy within $\mathcal{L}_{\mathrm{P}}(\mathrm{W}(m)$(-mon)-RRWW).
- Any lexicalized RRWW-automaton has word-expansion that is bounded from above by a linear function. It remains to study those languages that are obtained as proper languages of lexicalized RRWW-automata for which the word-expansion is non-constant, but bounded from above by a slowly growing function.

Conclusions

- The degree of word-expansion and the word-alphabet-expansion are new measures for the degree of nondeterminism for proper languages of restarting automata.
- we have obtained infinite hierarchies of language classes for monotone and for non-monotone RRWW-automata that are lexicalized.
- In the monotone case these classes form an infinite hierarchy between DCFL and CFL.
- Moreover, for each finite degree m of word-expansion, the number of available different auxiliary symbols yields an infinite hierarchy within $\mathcal{L}_{\mathrm{P}}(\mathsf{W}(m)(\text{-mon})\text{-RRWW})$.
- Any lexicalized RRWW-automaton has word-expansion that is bounded from above by a linear function. It remains to study those languages that are obtained as proper languages of lexicalized RRWW-automata for which the word-expansion is non-constant, but bounded from above by a slowly growing function.

Conclusions

- The degree of word-expansion and the word-alphabet-expansion are new measures for the degree of nondeterminism for proper languages of restarting automata.
- we have obtained infinite hierarchies of language classes for monotone and for non-monotone RRWW-automata that are lexicalized.
- In the monotone case these classes form an infinite hierarchy between DCFL and CFL.
- Moreover, for each finite degree m of word-expansion, the number of available different auxiliary symbols yields an infinite hierarchy within $\mathcal{L}_{\mathrm{P}}$(W(m)(-mon)-RRWW).
- Any lexicalized RRWW-automaton has word-expansion that is bounded from above by a linear function. It remains to study those languages that are obtained as proper languages of lexicalized RRWW-automata for which the word-expansion is non-constant, but bounded from above by a slowly growing function.