Jumping Pure Grammars

Alexander Meduna and Zbyněk Křivka

meduna@fit.vutbr.cz, krivka@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Talk at LTA 2018, Brno,
December 12, 2018

Outline of this talk

- Based on

  Křivka, Z., Kučera, J., and Meduna, A.: Jumping Pure Grammars.
  In: The Computer Journal, 2018.

Contents of this talk:

Introduction

Motivation

- Classical grammars and automata work strictly continuously

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
    - strongly-scattered information processing (bioinformatics, DNA computing)

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
  - strongly-scattered information processing (bioinformatics, DNA computing)
- Keep the structure of classical models unchanged. Change the way they work so they jump on strings.

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
  - strongly-scattered information processing (bioinformatics, DNA computing)
- Keep the structure of classical models unchanged. Change the way they work so they jump on strings.

Why jumping on strings

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
  - strongly-scattered information processing (bioinformatics, DNA computing)
- Keep the structure of classical models unchanged. Change the way they work so they jump on strings.

Related important publications

- Meduna, A. and Zemek, P.: Jumping Finite Automata, Int. J. Found. Comput. Sci., 2012 (citations: 34)

Why jumping on strings

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
  - strongly-scattered information processing (bioinformatics, DNA computing)
- Keep the structure of classical models unchanged. Change the way they work so they jump on strings.

Related important publications

- Meduna, A. and Zemek, P.: Jumping Finite Automata, Int. J. Found. Comput. Sci., 2012 (citations: 34)
- Křivka, Z. and Meduna, A.: Jumping Grammars, Int. J. Found. Comput. Sci., 2015 (citations: 5)

Why jumping on strings

Motivation

- Classical grammars and automata work strictly continuously
- Adaptation of classical models to work on strings discontinuously
  - strongly-scattered information processing (bioinformatics, DNA computing)
- Keep the structure of classical models unchanged. Change the way they work so they jump on strings.

Related important publications

- Meduna, A. and Zemek, P.: Jumping Finite Automata, Int. J. Found. Comput. Sci., 2012 (citations: 34)
- Křivka, Z. and Meduna, A.: Jumping Grammars, Int. J. Found. Comput. Sci., 2015 (citations: 5)
- Křivka, Z., Kučera, J. and Meduna, A.: Jumping Pure Grammars, Computer Journal, 2018

- grammar G is based on rules of the form

$$x \rightarrow y$$

- grammar G is based on rules of the form

$$x \to y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \to y$, G rewrites uxv to uyv.

- grammar G is based on rules of the form

$$x \rightarrow y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \rightarrow y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;

- grammar G is based on rules of the form

$$x \to y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \to y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;

- grammar G is based on rules of the form

$$x \rightarrow y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \rightarrow y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y at the same position where x was.

- grammar G is based on rules of the form

$$x \rightarrow y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \rightarrow y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y at the same position where x was.

Definition (Jumping grammars)

Let $z = uxv$. By using $x \rightarrow y$, G performs:

- grammar G is based on rules of the form

$$x \to y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \to y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y at the same position where x was.

Definition (Jumping grammars)

Let $z = uxv$. By using $x \to y$, G performs:

1. selects an occurrence of x in z;

- grammar G is based on rules of the form

$$x \to y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \to y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y at the same position where x was.

Definition (Jumping grammars)

Let $z = uxv$. By using $x \to y$, G performs:

1. selects an occurrence of x in z;
2. erase x from z;

- grammar G is based on rules of the form

$$x \rightarrow y$$

Definition (Classical grammars)

Let $z = uxv$. By using $x \rightarrow y$, G rewrites uxv to uyv.

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y at the same position where x was.

Definition (Jumping grammars)

Let $z = uxv$. By using $x \rightarrow y$, G performs:

1. selects an occurrence of x in z;
2. erase x from z;
3. G inserts y anywhere in uv.

Definition (Pure grammars)

1. they use only terminals;

Pure Grammars

Definition (Pure grammars)

1. they use only terminals;
2. derivations start from a starting string (axiom) σ;

Definition (Pure grammars)

1. they use only terminals;
2. derivations start from a starting string (axiom) σ;
3. every string they derive from σ belongs to the generated language.

Example: Classical Grammar

Example

Classical Grammar

1. Starting nonterminal S. Rules:

$$S \rightarrow a, S \rightarrow aa$$

Trivially, generated language is $\{a, aa\}$.

Example

Pure Grammars

1. Starting string S. Rules:

$$S \rightarrow a, S \rightarrow aa$$

Then, generated language is $\{S, a, aa\}$.

Example

Pure Grammars

1. Starting string S. Rules:

$$S \rightarrow a, S \rightarrow aa$$

   Then, generated language is $\{S, a, aa\}$.

2. Starting string a. Rules:

$$a \rightarrow aa$$

   Generated language is $\{a\}^{+}$.

Example

Pure Grammars

1. Starting string S. Rules:

$$S \rightarrow a, S \rightarrow aa$$

   Then, generated language is $\{S, a, aa\}$.

2. Starting string a. Rules:

$$a \rightarrow aa$$

   Generated language is $\{a\}^+$.

3. Starting string aa. Rules:

$$a \rightarrow \varepsilon$$

   Generated language is $\{\varepsilon, a, aa\}$.

Example: Jumping Grammar

Example

Jumping Grammar

1. Starting nonterminal S. Rules:

$$S \rightarrow aS, S \rightarrow b$$

   Trivially, generated language is $\{a\}^*\{b\}\{a\}^*$.

Example

Jumping Pure Grammar

1. Starting string ab. Rules:

$$a \rightarrow a$$

Generated language is $\{ab, ba\}$.

Preliminaries & Definitions

- For an alphabet of symbols Σ, Σ^* denotes the set of all strings over Σ.
- Algebraically, Σ^* represents the free monoid generated by Σ under concatenation.
- The unit of Σ^* is denoted by ε (the empty string).
- $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.
- Any $L \subseteq \Sigma^*$ is a language over Σ.
- Let $a \in \Sigma$ and $w \in L$,
  - $|w|$ denotes the length of w and
  - $|w|_a$ denotes the number of occurrences of a in w.
- **REG** $\subset$ **CF** $\subset$ **CS**
  - **REG**, **CF**, and **CS** denote the families of regular, context-free, and context-sensitive languages, respectively.

Definition (Pure Grammars)

A pure grammar (PG for short) is a triplet, $G = (\Sigma, P, \sigma)$, where

- Σ is an alphabet;
- P is a finite relation from Σ^+ to Σ^*;
- $\sigma \in \Sigma^+$ is the start string.

Any member $(x, y) \in P$ is called a rule and written as $x \rightarrow y$

Definition (Pure Grammars)

A pure grammar (PG for short) is a triplet, $G = (\Sigma, P, \sigma)$, where

- Σ is an alphabet;
- P is a finite relation from Σ^+ to Σ^*;
- $\sigma \in \Sigma^+$ is the start string.

Any member $(x, y) \in P$ is called a rule and written as $x \rightarrow y$

Definition (Propagating Pure Grammars)

If for every $x \rightarrow y \in P$, $y \neq \varepsilon$, G is propagating.

Definition (Pure Grammars)

A pure grammar (PG for short) is a triplet, $G = (\Sigma, P, \sigma)$, where

- Σ is an alphabet;
- P is a finite relation from Σ^+ to Σ^*;
- $\sigma \in \Sigma^+$ is the start string.

Any member $(x, y) \in P$ is called a rule and written as $x \rightarrow y$

Definition (Propagating Pure Grammars)

If for every $x \rightarrow y \in P$, $y \neq \varepsilon$, G is propagating.

Definition (Context-Free Pure Grammars)

If for every $x \rightarrow y \in P$, $|x| = 1$, G is context-free (CFPG for short).

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \ {}_s\!\Rightarrow uyv$ in G iff there exists $x \rightarrow y \in P$;

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \,{}_{s}\!\!\Rightarrow uyv$ in G iff there exists $x \rightarrow y \in P$;
2. Jumping mode: Let $w \in \Sigma^*$.
   (a) right mode: $uxwv \,{}_{j}\!\!\Rightarrow uwyv$ in G iff $x \rightarrow y \in P$ or
   (b) left mode: $uwxv \,{}_{j}\!\!\Rightarrow uywv$ in G iff $x \rightarrow y \in P$ in G;

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \ {}_s\!\Rightarrow uyv$ in G iff there exists $x \rightarrow y \in P$;
2. Jumping mode: Let $w \in \Sigma^*$.
   (a) right mode: $uxwv \ {}_j\!\Rightarrow uwyv$ in G iff $x \rightarrow y \in P$ or
   (b) left mode: $uwxv \ {}_j\!\Rightarrow uywv$ in G iff $x \rightarrow y \in P$ in G;
3. Parallel mode: $u \ {}_p\!\Rightarrow v$ in G iff there exist $x_1 \rightarrow y_1, x_2 \rightarrow y_2, \ldots, x_n \rightarrow y_n \in P$ such that $u = x_1 x_2 \cdots x_n$ and $v = y_1 y_2 \cdots y_n$, where $n \geq 0$;

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \ {}_s\!\Rightarrow uyv$ in G iff there exists $x \to y \in P$;
2. Jumping mode: Let $w \in \Sigma^*$.
   (a) right mode: $uxwv \ {}_j\!\Rightarrow uwyv$ in G iff $x \to y \in P$ or
   (b) left mode: $uwxv \ {}_j\!\Rightarrow uywv$ in G iff $x \to y \in P$ in G;
3. Parallel mode: $u \ {}_p\!\Rightarrow v$ in G iff there exist $x_1 \to y_1, x_2 \to y_2, \dots, x_n \to y_n \in P$ such that $u = x_1x_2 \cdots x_n$ and $v = y_1y_2 \cdots y_n$, where $n \geq 0$;
4. Jumping Parallel mode: $u \ {}_{jp}\!\Rightarrow v$ in G iff there exist $x_1 \to y_1, x_2 \to y_2, \dots, x_n \to y_n \in P$ such that $u = x_1x_2 \cdots x_n$ and $v = z_1z_2 \cdots z_n$, where $(z_1, z_2, \dots, z_n)$ is a permutation of $(y_1, y_2, \dots, y_n)$, $n \geq 0$.

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \ {}_s\!\Rightarrow uyv$ in G iff there exists $x \rightarrow y \in P$;
2. Jumping mode: Let $w \in \Sigma^*$.
   (a) right mode: $uxwv \ {}_j\!\Rightarrow uwyv$ in G iff $x \rightarrow y \in P$ or
   (b) left mode: $uwxv \ {}_j\!\Rightarrow uywv$ in G iff $x \rightarrow y \in P$ in G;
3. Parallel mode: $u \ {}_p\!\Rightarrow v$ in G iff there exist $x_1 \rightarrow y_1, x_2 \rightarrow y_2, \ldots, x_n \rightarrow y_n \in P$ such that $u = x_1 x_2 \cdots x_n$ and $v = y_1 y_2 \cdots y_n$, where $n \geq 0$;
4. Jumping Parallel mode: $u \ {}_{jp}\!\Rightarrow v$ in G iff there exist $x_1 \rightarrow y_1, x_2 \rightarrow y_2, \ldots, x_n \rightarrow y_n \in P$ such that $u = x_1 x_2 \cdots x_n$ and $v = z_1 z_2 \cdots z_n$, where $(z_1, z_2, \ldots, z_n)$ is a permutation of $(y_1, y_2, \ldots, y_n)$, $n \geq 0$.

Definition (Derivation Modes)

Let $u, v \in \Sigma^*$.
Derivation step according to a mode:

1. Sequential mode: $uxv \;{}_{s}\!\Rightarrow uyv$ in G iff there exists $x \to y \in P$;
2. Jumping mode: Let $w \in \Sigma^*$.
   (a) right mode: $uxwv \;{}_{j}\!\Rightarrow uwyv$ in G iff $x \to y \in P$ or
   (b) left mode: $uwxv \;{}_{j}\!\Rightarrow uywv$ in G iff $x \to y \in P$ in G;
3. Parallel mode: $u \;{}_{p}\!\Rightarrow v$ in G iff there exist $x_1 \to y_1, x_2 \to y_2, \dots, x_n \to y_n \in P$ such that $u = x_1x_2 \cdots x_n$ and $v = y_1y_2 \cdots y_n$, where $n \geq 0$;
4. Jumping Parallel mode: $u \;{}_{jp}\!\Rightarrow v$ in G iff there exist $x_1 \to y_1, x_2 \to y_2, \dots, x_n \to y_n \in P$ such that $u = x_1x_2 \cdots x_n$ and $v = z_1z_2 \cdots z_n$, where $(z_1, z_2, \dots, z_n)$ is a permutation of $(y_1, y_2, \dots, y_n)$, $n \geq 0$.

Definition (Generated Language)

For $h \in \{s, j, p, jp\}$, $L(G, {}_{h}\!\Rightarrow) = \{x \mid \sigma \;{}_{h}\!\Rightarrow^* x\}$.

Example (1)

Consider CFPG $G = (\{a, b, c, d\}, P, a)$ with

$$P = \{a \to abcd, a \to a, b \to b, c \to c, d \to d\}$$

Modes ${}_s\Rightarrow$ and ${}_p\Rightarrow$:

$$a \;{}_s\Rightarrow abcd \;{}_s\Rightarrow abcd \;{}_s\Rightarrow abcdbcd \;{}_s\Rightarrow abcdbcdbcd$$
$$a \;{}_p\Rightarrow abcd \;{}_p\Rightarrow abcd \;{}_p\Rightarrow abcdbcd \;{}_p\Rightarrow abcdbcdbcd$$

$L(G, {}_s\Rightarrow) = L(G, {}_p\Rightarrow) = \{a\}\{bcd\}^* \in$ **REG**

Modes ${}_j\Rightarrow$ and ${}_{jp}\Rightarrow$:

$$a \;{}_j\Rightarrow abcd \;{}_j\Rightarrow bacd \;{}_j\Rightarrow badc \;{}_j\Rightarrow bdabcdc$$
$$a \;{}_{jp}\Rightarrow abcd \;{}_{jp}\Rightarrow badc \;{}_{jp}\Rightarrow bdabcdc$$

$L(G, {}_j\Rightarrow) = L(G, {}_{jp}\Rightarrow) = \{w \mid |w|_a = 1, |w|_b = |w|_c = |w|_d\} \in$ **CS** $-$ **CF**

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

Examples

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with $_s\Rightarrow$:

   $S \rightarrow aA, S \rightarrow bB$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

   $S \rightarrow aA, S \rightarrow bB$
   $A \rightarrow \varepsilon, A \rightarrow aA$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

$$S \rightarrow aA, S \rightarrow bB$$
$$A \rightarrow \varepsilon, A \rightarrow aA$$
$$B \rightarrow \varepsilon, B \rightarrow bB$$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

$$S \rightarrow aA, S \rightarrow bB$$
$$A \rightarrow \varepsilon, A \rightarrow aA$$
$$B \rightarrow \varepsilon, B \rightarrow bB$$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

   $S \rightarrow aA$, $S \rightarrow bB$
   $A \rightarrow \varepsilon$, $A \rightarrow aA$
   $B \rightarrow \varepsilon$, $B \rightarrow bB$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

   $a \rightarrow aa$, $b \rightarrow bb$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

   $S \rightarrow aA$, $S \rightarrow bB$
   $A \rightarrow \varepsilon$, $A \rightarrow aA$
   $B \rightarrow \varepsilon$, $B \rightarrow bB$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

   $a \rightarrow aa$, $b \rightarrow bb$
   $a \rightarrow b$ or $c \rightarrow a$, $c \rightarrow b$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

   $S \rightarrow aA$, $S \rightarrow bB$
   $A \rightarrow \varepsilon$, $A \rightarrow aA$
   $B \rightarrow \varepsilon$, $B \rightarrow bB$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

   $a \rightarrow aa$, $b \rightarrow bb$
   $a \rightarrow b$ or $c \rightarrow a$, $c \rightarrow b$

3. In classical CF grammar with jumping, simply use G_1 with ${}_j\Rightarrow$. For instance,

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

$$S \rightarrow aA, S \rightarrow bB$$
$$A \rightarrow \varepsilon, A \rightarrow aA$$
$$B \rightarrow \varepsilon, B \rightarrow bB$$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

$$a \rightarrow aa, b \rightarrow bb$$
$$a \rightarrow b \text{ or } c \rightarrow a, c \rightarrow b$$

3. In classical CF grammar with jumping, simply use G_1 with ${}_j\Rightarrow$. For instance,

$$S \;{}_j\Rightarrow aA \;{}_j\Rightarrow aAa \;{}_j\Rightarrow aaaA \;{}_j\Rightarrow aaa$$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

1. In classical CF grammar, G_1 with ${}_s\Rightarrow$:

   $S \rightarrow aA$, $S \rightarrow bB$
   $A \rightarrow \varepsilon$, $A \rightarrow aA$
   $B \rightarrow \varepsilon$, $B \rightarrow bB$

2. In CFPG, $G_2 = (\{a, b\}, P, \sigma)$ with ${}_s\Rightarrow$; $\sigma = ?$. P :

   $a \rightarrow aa$, $b \rightarrow bb$
   $a \rightarrow b$ or $c \rightarrow a$, $c \rightarrow b$

3. In classical CF grammar with jumping, simply use G_1 with ${}_j\Rightarrow$. For instance,

   $S\ {}_j\Rightarrow aA\ {}_j\Rightarrow aAa\ {}_j\Rightarrow aaaA\ {}_j\Rightarrow aaa$

   $L(G_1, {}_j\Rightarrow) = \{a\}^+ \cup \{b\}^+$.

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

   $a \rightarrow b$, $a \rightarrow bb$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

$$a \rightarrow b,\ a \rightarrow bb$$
$$b \rightarrow a,\ b \rightarrow aa$$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

   $a \rightarrow b$, $a \rightarrow bb$
   $b \rightarrow a$, $b \rightarrow aa$

   For instance, $a \;{}_p\Rightarrow\; b \;{}_p\Rightarrow\; aa \;{}_p\Rightarrow\; bbb \;{}_p\Rightarrow\; aaa \;{}_p\Rightarrow \cdots$

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

   $a \rightarrow b$, $a \rightarrow bb$
   $b \rightarrow a$, $b \rightarrow aa$

   For instance, $a \;{}_p\Rightarrow b \;{}_p\Rightarrow aa \;{}_p\Rightarrow bbb \;{}_p\Rightarrow aaa \;{}_p\Rightarrow \cdots$

   $L(G_3, {}_p\Rightarrow) = \{a\}^+ \cup \{b\}^+$. ☺

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

   $a \rightarrow b,\ a \rightarrow bb$
   $b \rightarrow a,\ b \rightarrow aa$

   For instance, $a\ {}_p\Rightarrow\ b\ {}_p\Rightarrow\ aa\ {}_p\Rightarrow\ bbb\ {}_p\Rightarrow\ aaa\ {}_p\Rightarrow \cdots$

   $L(G_3, {}_p\Rightarrow) = \{a\}^+ \cup \{b\}^+$. ☺

4. In CFPG grammar with ${}_{jp}\Rightarrow$, simply use G_3 with ${}_{jp}\Rightarrow$.

Example (2)

Our goal: Language $L = \{a\}^+ \cup \{b\}^+$ with $\Sigma = \{a, b\}$:

3. In CFPG, $G_3 = (\{a, b\}, P, a)$ with ${}_p\Rightarrow$ and P:

   $a \rightarrow b,\ a \rightarrow bb$
   $b \rightarrow a,\ b \rightarrow aa$

   For instance, $a\ {}_p\Rightarrow b\ {}_p\Rightarrow aa\ {}_p\Rightarrow bbb\ {}_p\Rightarrow aaa\ {}_p\Rightarrow \cdots$

   $L(G_3, {}_p\Rightarrow) = \{a\}^+ \cup \{b\}^+$. ☺

4. In CFPG grammar with ${}_{jp}\Rightarrow$, simply use G_3 with ${}_{jp}\Rightarrow$.

   For instance, $a\ {}_{jp}\Rightarrow b\ {}_{jp}\Rightarrow aa\ {}_{jp}\Rightarrow bbb\ {}_{jp}\Rightarrow aaa\ {}_{jp}\Rightarrow \cdots$
   $L(G_3, {}_{jp}\Rightarrow) = \{a\}^+ \cup \{b\}^+$.

J $=$ Jumping, **S** $=$ Sequential, **P** $=$ Parallel, **CF** $=$ Context-Free.
For a language family **X**, its propagating variant is $\mathbf{X}^{-\varepsilon}$.

1. **S** $= \{L(G, {}_s\!\Rightarrow) \mid G$ is a PG$\}$;
2. **JS** $= \{L(G, {}_j\!\Rightarrow) \mid G$ is a PG$\}$;
3. **P** $= \{L(G, {}_p\!\Rightarrow) \mid G$ is a PG$\}$;
4. **JP** $= \{L(G, {}_{jp}\!\Rightarrow) \mid G$ is a PG$\}$;
5. **SCF** $= \{L(G, {}_s\!\Rightarrow) \mid G$ is a CFPG$\}$;
6. **JSCF** $= \{L(G, {}_j\!\Rightarrow) \mid G$ is a CFPG$\}$;
7. **PCF** $= \{L(G, {}_p\!\Rightarrow) \mid G$ is a CFPG$\}$;
8. **JPCF** $= \{L(G, {}_{jp}\!\Rightarrow) \mid G$ is a CFPG$\}$;

Results

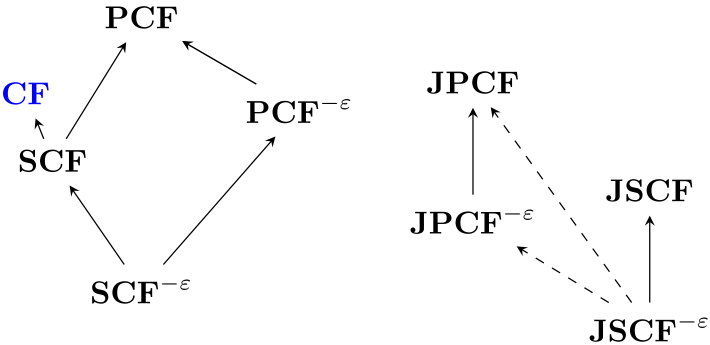
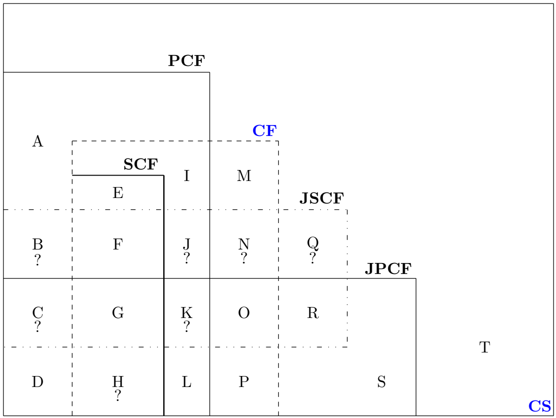
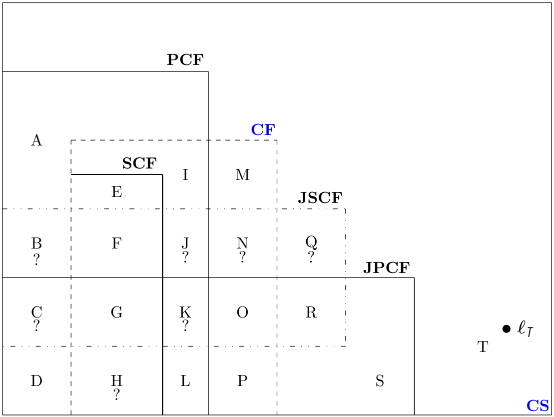


Figure: Dashed arrow = open problem. No connection = incomparability.

Note: Two language families X and Y are incomparable iff $X \not\subseteq Y$ and $Y \not\subseteq X$.

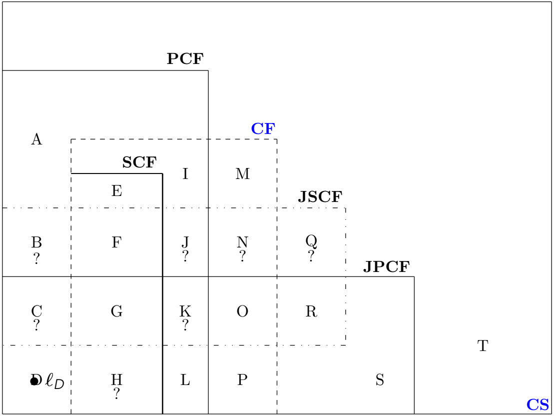

PCF
CF
SCF
JSCF
JPCF
CS
A
B ?
C ?
D
E
F
G
H ?
I
J ?
K ?
L
M
N ?
O
P
Q ?
R
S
T



$\ell_T = \{a^p \mid p \text{ is a prime}\} \in$ **CS** $-$ (**PCF** $\cup$ **CF** $\cup$ **JSCF** $\cup$ **JPCF**)

Idea $\ell_T \notin$ **JPCF**: $a \to \varepsilon \notin P$, so $\sigma = a^2$. We need $a^2 \ {}_{jp}\!\Rightarrow a^3$ by $a \to aa$ and $a \to a$, but we get a^4 as well.

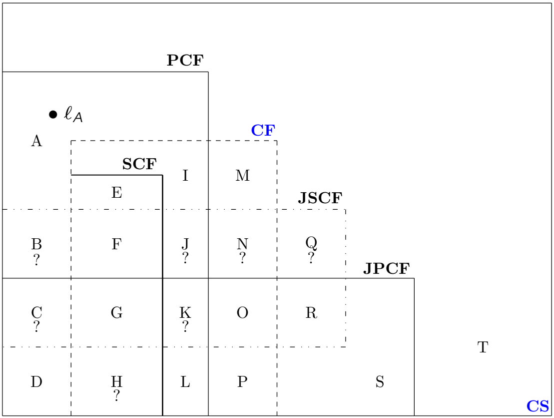
Then, $\mathbf{JPCF}_u = \mathbf{PCF}_u \supset \mathbf{JSCF}_u = \mathbf{SCF}_u$.



$\ell_D = \{a^{2^n} \mid n \geq 0\} = (\mathbf{PCF} \cap \mathbf{JPCF}) - (\mathbf{CF} \cup \mathbf{JSCF})$

Idea $\ell_D \in \mathbf{PCF} \cap \mathbf{JPCF}$: Take rule $a \to aa$ with $\sigma = a$.

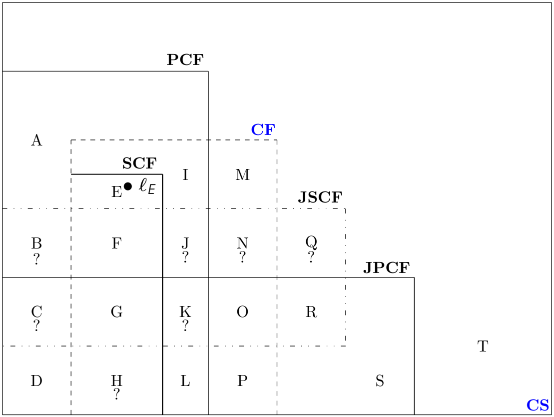
Idea $\ell_D \notin \mathbf{CF} \cup \mathbf{JSCF}$: ℓ_D is not semilinear.



$\ell_A = \{a^{2^n} b^{2^n} \mid n \geq 0\} = \mathbf{PCF} - (\mathbf{CF} \cup \mathbf{JSCF} \cup \mathbf{JPCF})$

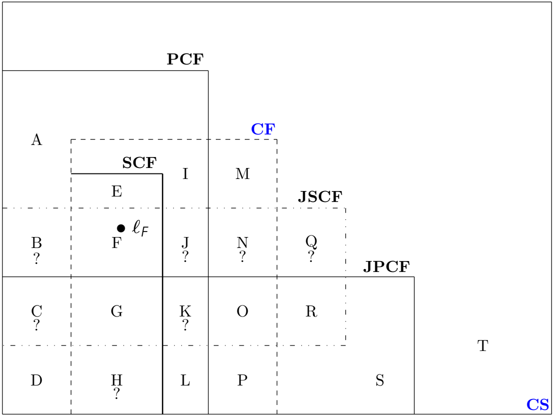
Idea $\ell_A \in$ **PCF**: Take rules $a \rightarrow aa$ and $b \rightarrow bb$ with $\sigma = ab$.

Idea $\ell_A \notin$ **JPCF**: Show the proof by contradiction.



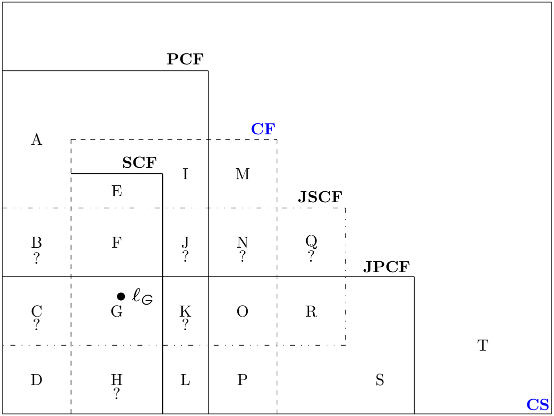
$\ell_E = \{a^n c b^n \mid n \geq 0\} \in$ **SCF** $-$ (**JSCF** $\cup$ **JPCF**)

Idea $\ell_E \in$ **SCF**: Take rule $c \rightarrow acb$ with $\sigma = c$.

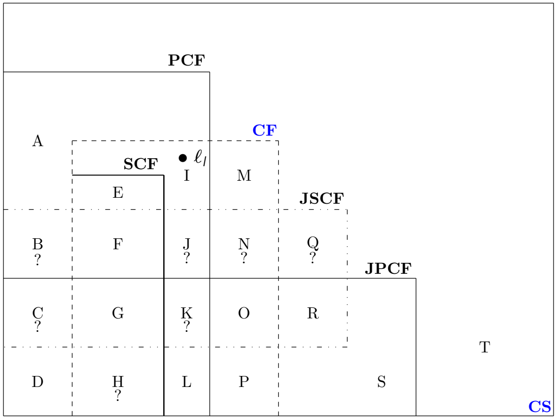


$\ell_F = \{aa, aab, aac, aabc\} \in (\mathbf{SCF} \cap \mathbf{JSCF}) - \mathbf{JPCF}$

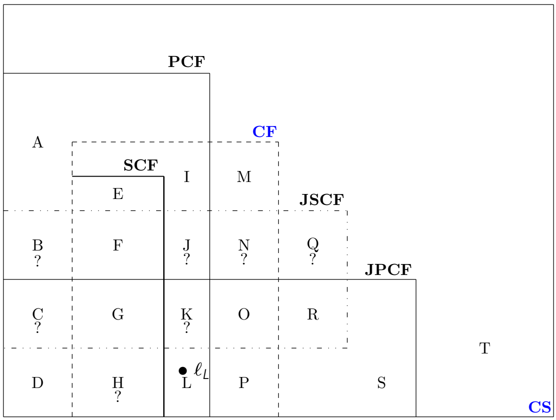
Idea for $\ell_F \in \mathbf{SCF} \cap \mathbf{JSCF}$: Take $\sigma = aabc$ and rules $b \to \varepsilon$ and $c \to \varepsilon$.



$$\ell_G = \{a\}^+ \in \mathbf{SCF} \cap \mathbf{JSCF} \cap \mathbf{JPCF}$$



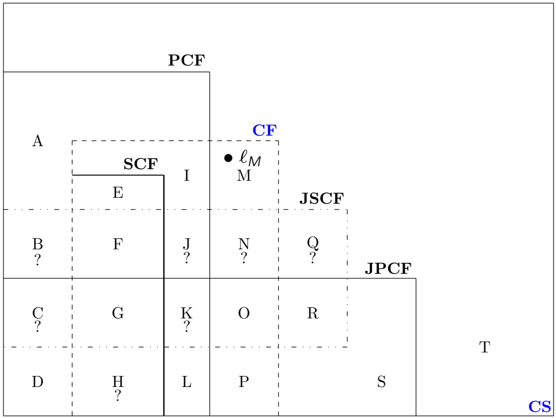
$$\ell_I = \{aabb, ccdd\} \in (\mathbf{PCF} \cap \mathbf{CF}) - (\mathbf{SCF} \cup \mathbf{JSCF} \cup \mathbf{JPCF})$$



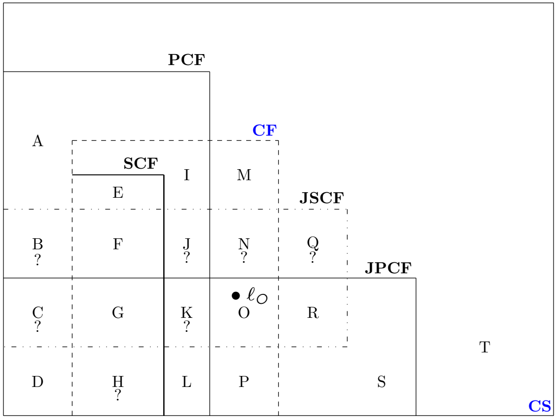
$\ell_L = \{ab, cd, dc\} \in$ (**PCF** $\cap$ **CF** $\cap$ **JPCF**) $-$ (**SCF** $\cup$ **JSCF**)

We need to rewrite two symbols in parallel such as with $a \to c, b \to d, c \to d, d \to c$ with $\sigma = ab$.

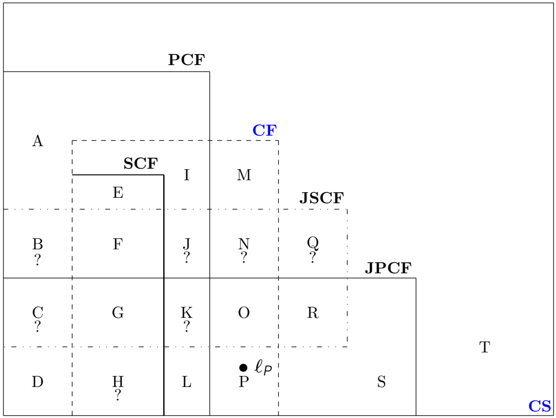
For instance, $ab \ {}_{p}\!\Rightarrow cd \ {}_{p}\!\Rightarrow dc$ or $ab \ {}_{jp}\!\Rightarrow dc$.



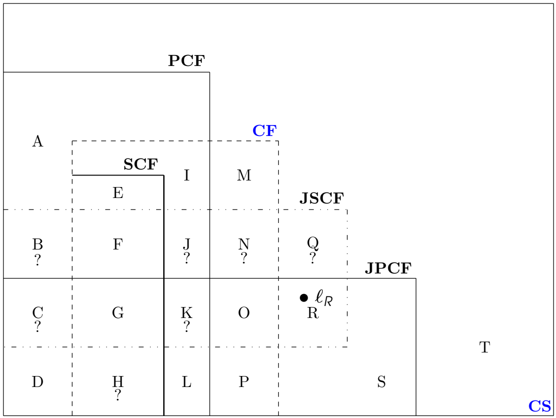
$$\ell_M = \{a^n b^n \mid n \geq 1\} \in \mathbf{CF} - (\mathbf{PCF} \cup \mathbf{JSCF} \cup \mathbf{JPCF})$$



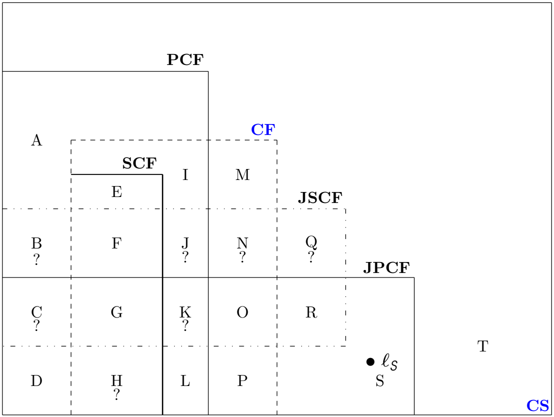
$\ell_O = \{aabb, abab, abba, baab, baba, bbaa\} \in$ (**CF** $\cap$ **JSCF** $\cap$ **JPCF**) $-$ **PCF**



$\ell_P = \{aabb, ccdd, cdcd, cddc, dccd, dcdc, ddcc\} \in$ $(\mathbf{CF} \cap \mathbf{JPCF}) - (\mathbf{PCF} \cup \mathbf{JSCF})$



$$\ell_R = \left\{ w \;\middle|\; \begin{array}{l} |w|_a - 1 = |w|_b = |w|_c, \\ w \in \{a,b,c\}^+ \end{array} \right\} \in (\mathbf{JSCF} \cap \mathbf{JPCF}) - (\mathbf{CF} \cup \mathbf{PCF})$$



$$\ell_S = \{\hat{a}\hat{b}\hat{c}\} \cup \left\{ w \;\middle|\; \begin{array}{l} |w|_a - 1 = |w|_b = |w|_c, \\ w \in \{a, b, c\}^+ \end{array} \right\} \in$$

JPCF $-$ (**CF** $\cup$ **PCF** $\cup$ **JSCF**)

$$
\begin{tikzcd}
\mathbf{PCF}_u^{-\varepsilon} \arrow[r, dash] \arrow[d] & \mathbf{JPCF}_u^{-\varepsilon} \arrow[d] \\
\mathbf{PCF}_u \arrow[r, dash] & \mathbf{JPCF}_u \\
\mathbf{SCF}_u \arrow[u] \arrow[r, dash] & \mathbf{JSCF}_u \arrow[u] \\
\mathbf{SCF}_u^{-\varepsilon} \arrow[u] \arrow[uuu, bend right] \arrow[r, dash] & \mathbf{JSCF}_u^{-\varepsilon} \arrow[u] \arrow[uuu, bend right]
\end{tikzcd}
$$

$$
\begin{tikzcd}
\mathbf{PCF}_u^{-\varepsilon} \arrow[r, dash] \arrow[d] & \mathbf{JPCF}_u^{-\varepsilon} \arrow[d] \\
\mathbf{PCF}_u \arrow[r, dash] & \mathbf{JPCF}_u \\
\mathbf{SCF}_u \arrow[r, dash] \arrow[u] & \mathbf{JSCF}_u \arrow[u] \\
\mathbf{SCF}_u^{-\varepsilon} \arrow[r, dash] \arrow[u] \arrow[uuu, bend right] & \mathbf{JSCF}_u^{-\varepsilon} \arrow[u] \arrow[uuu, bend right]
\end{tikzcd}
$$

Note: $\mathbf{SCF}_u$ and $\mathbf{PCF}_u^{-\varepsilon}$ are incomparable.

Conclusion

Conclusion, Future Work

- Open Problems
- Closure Properties
- Decidability (Emptiness, Universality, ...)
- Left-jumps and Right-jumps in Pure Grammars

Thank You For Your Attention!