Вычислительная сложность формальных грамматик

Александр Охотин

Кафедра математики, университет Турку, Финляндия

25 сентября 2011 г. от Р. Х.

Часть I

Назначение и математическое определение грамматик

Определение синтаксиса

- Сложные данные передаются последовательностью символов.

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:
   - Число — это выражение.

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:
   - Число — это выражение.
   - Если e и e' — выражения, то $e + e'$, $e * e'$ и (e) — тоже выражения.

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:
   - Число — это выражение.
   - Если e и e' — выражения, то $e + e'$, $e * e'$ и (e) — тоже выражения.
2. В естественных языках:

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:
   - Число — это выражение.
   - Если e и e' — выражения, то $e + e'$, $e * e'$ и (e) — тоже выражения.
2. В естественных языках:
   - *Подлежащее* и следующее за ним *сказуемое* образуют *предложение*.

Определение синтаксиса

- Сложные данные передаются последовательностью символов.
- Индуктивные определения правильных записей.

1. В математике, языках программирования, и т.д.:
   - Число — это выражение.
   - Если e и e' — выражения, то $e + e'$, $e * e'$ и (e) — тоже выражения.
2. В естественных языках:
   - *Подлежащее* и следующее за ним *сказуемое* образуют *предложение*.
   - *Прилагательное* и *существительное* образуют *подлежащее*.

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

N: вспомогательный символы для синтаксических понятий, таких, как *выражение* или *подлежащее*.

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

N: вспомогательный символы для синтаксических понятий, таких, как *выражение* или *подлежащее*.

- Всякая строка $w \in \Sigma^*$ может иметь или не иметь каждое свойство $A \in N$.

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

N: вспомогательный символы для синтаксических понятий, таких, как *выражение* или *подлежащее*.

- Всякая строка $w \in \Sigma^*$ может иметь или не иметь каждое свойство $A \in N$.

R: множество правил вида

$$A \to X_1 \dots X_\ell \quad (A \in N,\ \ell \geqslant 0,\ X_i \in \Sigma \cup N)$$

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

N: вспомогательный символы для синтаксических понятий, таких, как *выражение* или *подлежащее*.

- Всякая строка $w \in \Sigma^*$ может иметь или не иметь каждое свойство $A \in N$.

R: множество правил вида

$$A \to X_1 \dots X_\ell \quad (A \in N,\ \ell \geqslant 0,\ X_i \in \Sigma \cup N)$$

$S \in N$: множество правильных предложений.

Формальная запись для таких описаний

Четвёрка $G = (\Sigma, N, R, S)$, где

Σ: алфавит целевого языка (a,b,c,. . .)

N: вспомогательный символы для синтаксических понятий, таких, как *выражение* или *подлежащее*.

- Всякая строка $w \in \Sigma^*$ может иметь или не иметь каждое свойство $A \in N$.

R: множество правил вида

$$A \to X_1 \dots X_\ell \quad (A \in N,\ \ell \geqslant 0,\ X_i \in \Sigma \cup N)$$

$S \in N$: множество правильных предложений.

Если $w \in \Sigma^*$ представима, как $X_1 \cdot \ldots \cdot X_\ell$, то w имеет свойство A.

Определение через дедукцию

- Утверждения вида “w имеет свойство X”, где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.

Определение через дедукцию

- Утверждения вида “w имеет свойство X”, где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.

Определение через дедукцию

- Утверждения вида “w имеет свойство X”, где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\vdash [a, a] \qquad (\textit{аксиома})$$
$$\vdash [b, b] \qquad (\textit{аксиома})$$

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\begin{aligned} &\vdash [a, a] && (\textit{аксиома}) \\ &\vdash [b, b] && (\textit{аксиома}) \\ &\vdash [S, \varepsilon] && (S \to \varepsilon) \end{aligned}$$

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\begin{aligned}
&\vdash [a, a] && (\textit{аксиома}) \\
&\vdash [b, b] && (\textit{аксиома}) \\
&\vdash [S, \varepsilon] && (S \to \varepsilon) \\
[a, a], [S, \varepsilon], [b, b] &\vdash [S, ab] && (S \to aSb)
\end{aligned}$$

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\begin{aligned}
&\vdash [a, a] && (\textit{аксиома}) \\
&\vdash [b, b] && (\textit{аксиома}) \\
&\vdash [S, \varepsilon] && (S \to \varepsilon) \\
[a, a], [S, \varepsilon], [b, b] &\vdash [S, ab] && (S \to aSb) \\
[a, a], [S, ab], [b, b] &\vdash [S, aabb] && (S \to aSb)
\end{aligned}$$

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\begin{aligned}
&\vdash [a, a] && (\textit{аксиома}) \\
&\vdash [b, b] && (\textit{аксиома}) \\
&\vdash [S, \varepsilon] && (S \to \varepsilon) \\
[a, a], [S, \varepsilon], [b, b] &\vdash [S, ab] && (S \to aSb) \\
[a, a], [S, ab], [b, b] &\vdash [S, aabb] && (S \to aSb)
\end{aligned}$$

Pro: Это истинный смысл грамматик.

Определение через дедукцию

- Утверждения вида "w имеет свойство X", где $w \in \Sigma^*$ и $X \in \Sigma \cup N$. Обозначение: $[X, w]$.
- Аксиомы: $\vdash [a, a]$ для всех $a \in \Sigma$.
- Правила вывода:

$$[X_1, w_1], \ldots, [X_k, w_k] \vdash [A, w_1 \ldots w_k] \quad (A \to X_1 \ldots X_k \in R,\ w_i \in \Sigma^*)$$

Example: $S \to aSb \mid \varepsilon$

$$\begin{aligned}
&\vdash [a, a] && (\textit{аксиома}) \\
&\vdash [b, b] && (\textit{аксиома}) \\
&\vdash [S, \varepsilon] && (S \to \varepsilon) \\
[a, a], [S, \varepsilon], [b, b] &\vdash [S, ab] && (S \to aSb) \\
[a, a], [S, ab], [b, b] &\vdash [S, aabb] && (S \to aSb)
\end{aligned}$$

Pro: Это истинный смысл грамматик.

Contra: Не столь удобно в использовании.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

  где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.
- Всегда имеет решение, и среди них есть *наименьшее*.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

  где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.
- Всегда имеет решение, и среди них есть *наименьшее*.
- Записывает счётное множество шагов дедукции одновременно.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

  где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.
- Всегда имеет решение, и среди них есть *наименьшее*.
- Записывает счётное множество шагов дедукции одновременно.

Example: $S \to aSb \mid \varepsilon$

$S = \{a\}S\{b\} \cup \{\varepsilon\}$, *единственное решение* $S = \{a^n b^n \mid n \geqslant 0\}$.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

  где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.
- Всегда имеет решение, и среди них есть *наименьшее*.
- Записывает счётное множество шагов дедукции одновременно.

Example: $S \to aSb \mid \varepsilon$

$S = \{a\}S\{b\} \cup \{\varepsilon\}$, *единственное решение* $S = \{a^n b^n \mid n \geqslant 0\}$.

Pro: Легко проверить решение.

Определение через языковые уравнения

- Всякий $A \in N$: неизвестный язык над Σ.
  - Множество строк $w \in \Sigma^*$, имеющих свойство A.
- Система уравнений

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad \text{(для всех } A \in N\text{)},$$

  где $X_j = a \in \Sigma$ — постоянный язык $\{a\}$.
- Всегда имеет решение, и среди них есть *наименьшее*.
- Записывает счётное множество шагов дедукции одновременно.

Example: $S \to aSb \mid \varepsilon$

$S = \{a\}S\{b\} \cup \{\varepsilon\}$, *единственное решение* $S = \{a^n b^n \mid n \geqslant 0\}$.

Pro: Легко проверить решение.

Contra: Трудно получить непринадлежность наименьшему решению.

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
- Перезапись строк над $(\Sigma \cup N)^*$.

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
- Перезапись строк над $(\Sigma \cup N)^*$.
- Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
- Перезапись строк над $(\Sigma \cup N)^*$.
- Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

- $w \in \Sigma^*$ имеет свойство $A \in N$, если $A \Longrightarrow \ldots \Longrightarrow w$

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
- Перезапись строк над $(\Sigma \cup N)^*$.
- Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

- $w \in \Sigma^*$ имеет свойство $A \in N$, если $A \Longrightarrow \ldots \Longrightarrow w$
- $a \in \Sigma$: “терминальные символы”,
  $A \in N$: “нетерминальные символы”.

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
- Перезапись строк над $(\Sigma \cup N)^*$.
- Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

- $w \in \Sigma^*$ имеет свойство $A \in N$, если $A \Longrightarrow \ldots \Longrightarrow w$
- $a \in \Sigma$: “терминальные символы”,
  $A \in N$: “нетерминальные символы”.

Example: $S \to aSb \mid \varepsilon$

$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow aabb$

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
  - Перезапись строк над $(\Sigma \cup N)^*$.
  - Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

  - $w \in \Sigma^*$ имеет свойство $A \in N$, если $A \Longrightarrow \ldots \Longrightarrow w$
  - $a \in \Sigma$: “терминальные символы”, $A \in N$: “нетерминальные символы”.

Example: $S \to aSb \mid \varepsilon$

$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow aabb$

Pro: Просто, можно преподавать даже школьникам.

Определение через перезапись строк

- ✓ Н. Хомский, “О некоторых формальных свойствах грамматик” (1959).
  - Перезапись строк над $(\Sigma \cup N)^*$.
  - Правила — как правила перезаписи:

$$\mu A \nu \Longrightarrow \mu \alpha \nu \quad (A \to \alpha \in R)$$

  - $w \in \Sigma^*$ имеет свойство $A \in N$, если $A \Longrightarrow \ldots \Longrightarrow w$
  - $a \in \Sigma$: “терминальные символы”, $A \in N$: “нетерминальные символы”.

Example: $S \to aSb \mid \varepsilon$

$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow aabb$

Pro: Просто, можно преподавать даже школьникам.

Contra: Не передаёт истинный смысл грамматик.

Часть II

Разновидности грамматик

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

w имеет свойство S

$\Updownarrow$

$w = aub$, где u имеет свойство S

$\vee$

$w = \varepsilon$

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

$$w \text{ имеет свойство } S$$
$$\Updownarrow$$
$$w = aub, \text{ где } u \text{ имеет свойство } S$$
$$\vee$$
$$w = \varepsilon$$

- Несколько правил для одного символа: дизъюнкция.

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

$$w \text{ имеет свойство } S$$
$$\Updownarrow$$
$$w = aub, \text{ где } u \text{ имеет свойство } S$$
$$\vee$$
$$w = \varepsilon$$

- Несколько правил для одного символа: дизъюнкция.
  - “w удовлетворяет условию A или условию B”

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

$$w \text{ имеет свойство } S$$
$$\Updownarrow$$
$$w = aub, \text{ где } u \text{ имеет свойство } S$$
$$\vee$$
$$w = \varepsilon$$

- Несколько правил для одного символа: дизъюнкция.
  - “w удовлетворяет условию A или условию B”
- Где конъюнкция и отрицание?

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

$$w \text{ имеет свойство } S$$
$$\Updownarrow$$
$$w = aub, \text{ где } u \text{ имеет свойство } S$$
$$\vee$$
$$w = \varepsilon$$

- Несколько правил для одного символа: дизъюнкция.
  - “w удовлетворяет условию A или условию B”
- Где конъюнкция и отрицание?
  - “w удовлетворяет обоим условиям A и B”

Булевы действия в грамматиках

$S \to aSb \mid \varepsilon$

$$w \text{ имеет свойство } S$$
$$\Updownarrow$$
$$w = aub, \text{ где } u \text{ имеет свойство } S$$
$$\vee$$
$$w = \varepsilon$$

- Несколько правил для одного символа: дизъюнкция.
  - “w удовлетворяет условию A или условию B”
- Где конъюнкция и отрицание?
  - “w удовлетворяет обоим условиям A и B”
  - “w удовлетворяет A, но не B”

Конъюнктивные грамматики (Охотин, 2000)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad (\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*)$$

“Если w представимо в виде α_1, …, α_m, то w имеет свойство A”

Конъюнктивные грамматики (Охотин, 2000)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad (\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*)$$

“Если w представимо в виде α_1, . . . , α_m, то w имеет свойство A”

- Определение через дедукцию утверждений $[A, w]$.

Конъюнктивные грамматики (Охотин, 2000)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad (\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*)$$

"Если w представимо в виде α_1, ..., α_m, то w имеет свойство A"

- Определение через дедукцию утверждений $[A, w]$.
- Определение через языковые уравнения.

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in R} \bigcap_{i=1}^{m} \alpha_i$$

Конъюнктивные грамматики (Охотин, 2000)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad (\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*)$$

"Если w представимо в виде α_1, ..., α_m, то w имеет свойство A"

- Определение через дедукцию утверждений $[A, w]$.
- Определение через языковые уравнения.

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in R} \bigcap_{i=1}^{m} \alpha_i$$

- Определение через перезапись термов.

$$sAs' \Longrightarrow s(\alpha_1 \& \ldots \& \alpha_m)s' \Longrightarrow \ldots \Longrightarrow s(w \& \ldots \& w)s' \Longrightarrow sws'$$

Конъюнктивные грамматики (Охотин, 2000)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad (\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*)$$

"Если w представимо в виде α_1, ..., α_m, то w имеет свойство A"

- Определение через дедукцию утверждений $[A, w]$.
- Определение через языковые уравнения.

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in R} \bigcap_{i=1}^{m} \alpha_i$$

- Определение через перезапись термов.

$$sAs' \Longrightarrow s(\alpha_1 \& \ldots \& \alpha_m)s' \Longrightarrow \ldots \Longrightarrow s(w \& \ldots \& w)s' \Longrightarrow sws'$$

- Могут задать $\{a^n b^n c^n \mid n \geqslant 0\}$, $\{wcw \mid w \in \Sigma^*\}$, $\{a^{4^n} \mid n \geqslant 0\}$, и т.д.

Булевы грамматики (Охотин, 2003)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n$$

“Если w представимо в виде α_1, …, α_m и непредставимо в виде β_1, …, β_n, то w имеет свойство A”.

Булевы грамматики (Охотин, 2003)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n$$

“Если w представимо в виде α_1, …, α_m и непредставимо в виде β_1, …, β_n, то w имеет свойство A”.

- Можно выразить противоречие: $S \to \neg S$.

Булевы грамматики (Охотин, 2003)

Правила вида

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n$$

“Если w представимо в виде α_1, …, α_m и непредставимо в виде β_1, …, β_n, то w имеет свойство A”.

- Можно выразить противоречие: $S \to \neg S$.
- Определение через языковые уравнения.

Краткий обзор грамматик

- ✓ Индуктивное определение.

Краткий обзор грамматик

- ✓ Индуктивное определение.
- ✓ Длинные строки получаются сцеплением коротких.

Краткий обзор грамматик

- ✓ Индуктивное определение.
- ✓ Длинные строки получаются сцеплением коротких.
- ✓ Сцепления образуют дерево разбора.

Краткий обзор грамматик

- ✓ Индуктивное определение.
- ✓ Длинные строки получаются сцеплением коротких.
- ✓ Сцепления образуют дерево разбора.
- Строка длины n зависит от n^2 её подстрок.

Краткий обзор грамматик

- ✓ Индуктивное определение.
- ✓ Длинные строки получаются сцеплением коротких.
- ✓ Сцепления образуют дерево разбора.
- Строка длины n зависит от n^2 её подстрок.

1. Логика для определения синтаксиса.

Краткий обзор грамматик

- ✓ Индуктивное определение.
- ✓ Длинные строки получаются сцеплением коротких.
- ✓ Сцепления образуют дерево разбора.
- Строка длины n зависит от n^2 её подстрок.

1. Логика для определения синтаксиса.
2. Обыкновенная бесконтекстная грамматика — дизъюнктивный фрагмент.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.
3. Детерминированные или LR(k) грамматики: правило/разбиение можно определить, читая слева направо.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.
3. Детерминированные или LR(k) грамматики: правило/разбиение можно определить, читая слева направо.
4. LL(k) грамматики: можно определить, начиная читать подстроку.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.
3. Детерминированные или LR(k) грамматики: правило/разбиение можно определить, читая слева направо.
4. LL(k) грамматики: можно определить, начиная читать подстроку.
5. Грамматики, управлямые входом (input-driven; также “visibly pushdown” на ломаном английском):

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.
3. Детерминированные или LR(k) грамматики: правило/разбиение можно определить, читая слева направо.
4. LL(k) грамматики: можно определить, начиная читать подстроку.
5. Грамматики, управлямые входом (input-driven; также “visibly pushdown” на ломаном английском):
   - Алфавит $\Sigma_{+1} \cup \Sigma_0 \cup \Sigma_{-1}$.

Важные особые случаи грамматик

1. Линейные грамматики: правила вида $A \to uBv$ и $A \to w$, где $u, v, w \in \Sigma^*$.
   - $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$: линейная конъюнктивная.
2. Однозначные грамматики:
   - Правила $A \to \mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ порождают непересекающиеся языки.
   - Для всякого сцепления BC в правилах, $w \in L(B)L(C) \Longrightarrow \exists!$ разбиение.
3. Детерминированные или LR(k) грамматики: правило/разбиение можно определить, читая слева направо.
4. LL(k) грамматики: можно определить, начиная читать подстроку.
5. Грамматики, управлямые входом (input-driven; также “visibly pushdown” на ломаном английском):
   - Алфавит $\Sigma_{+1} \cup \Sigma_0 \cup \Sigma_{-1}$.
   - Правила $A \to aDbE$ ($a \in \Sigma_{+1}$, $b \in \Sigma_{-1}$), $A \to cD$ ($c \in \Sigma_0$), $A \to \varepsilon$.

Классификация формальных грамматик

- Общего вида — однозначные — детерминированные (LR) — сильно детерминированные (LL).

Классификация формальных грамматик

- Общего вида — однозначные — детерминированные (LR) — сильно детерминированные (LL).
- Общего вида — с линейным сцеплением — управляемые входом.

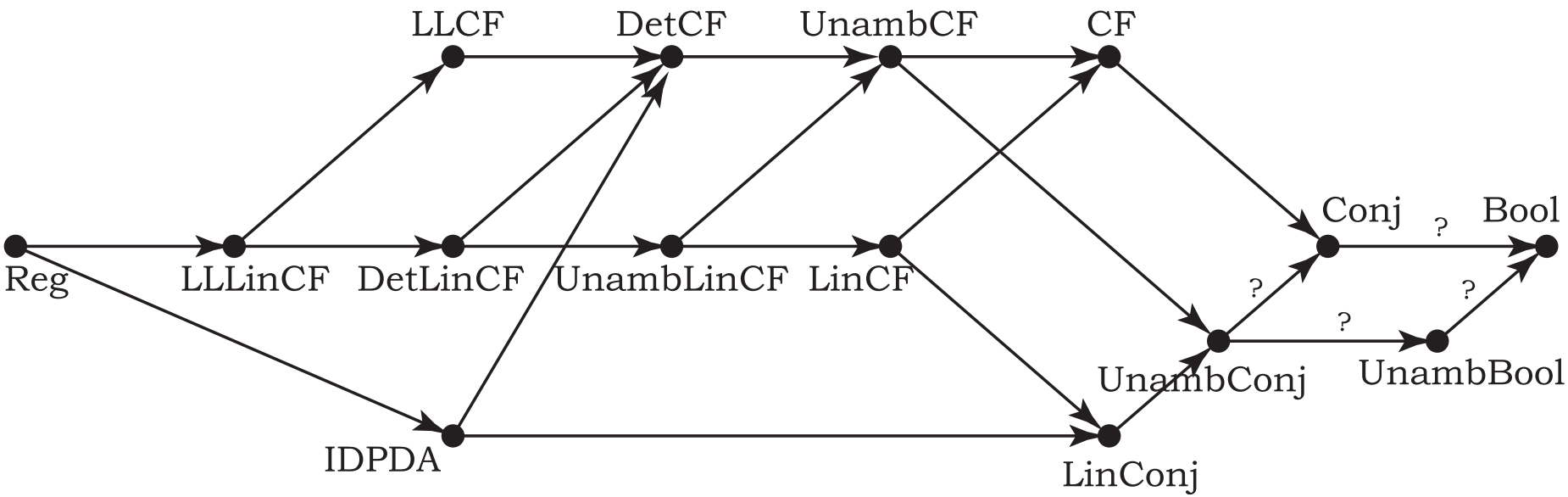
Классификация формальных грамматик

- Общего вида — однозначные — детерминированные (LR) — сильно детерминированные (LL).
- Общего вида — с линейным сцеплением — управляемые входом.
- Булевы действия: $\{\vee, \wedge, \neg\}$ — $\{\vee, \wedge\}$ — $\{\vee\}$.

Классификация формальных грамматик

- Общего вида — однозначные — детерминированные (LR) — сильно детерминированные (LL).
- Общего вида — с линейным сцеплением — управляемые входом.
- Булевы действия: $\{\vee, \wedge, \neg\}$ — $\{\vee, \wedge\}$ — $\{\vee\}$.

* * *



Часть III

Сложность языков

Время работы известных алгоритмов

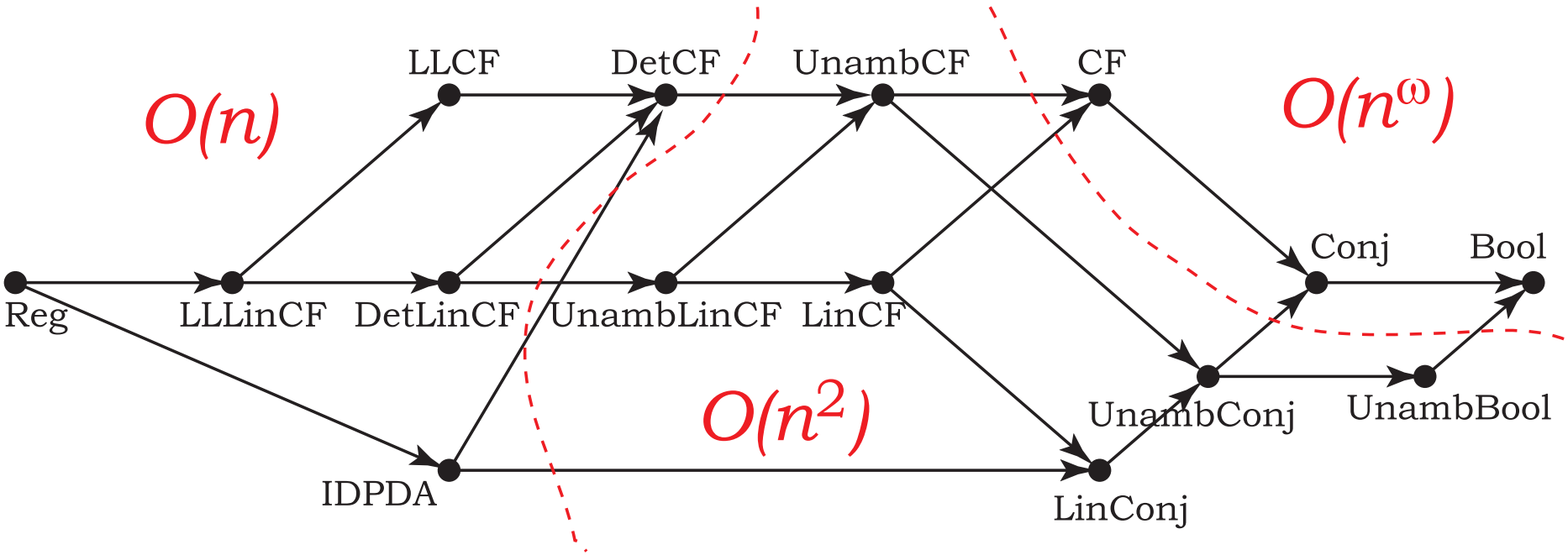
Задача принадлежности для семейства языков $\mathcal{L}$

Пусть $L \in \mathcal{L}$. Для данной строки w, определить, принадлежит ли она L.

Время работы известных алгоритмов

Задача принадлежности для семейства языков $\mathcal{L}$

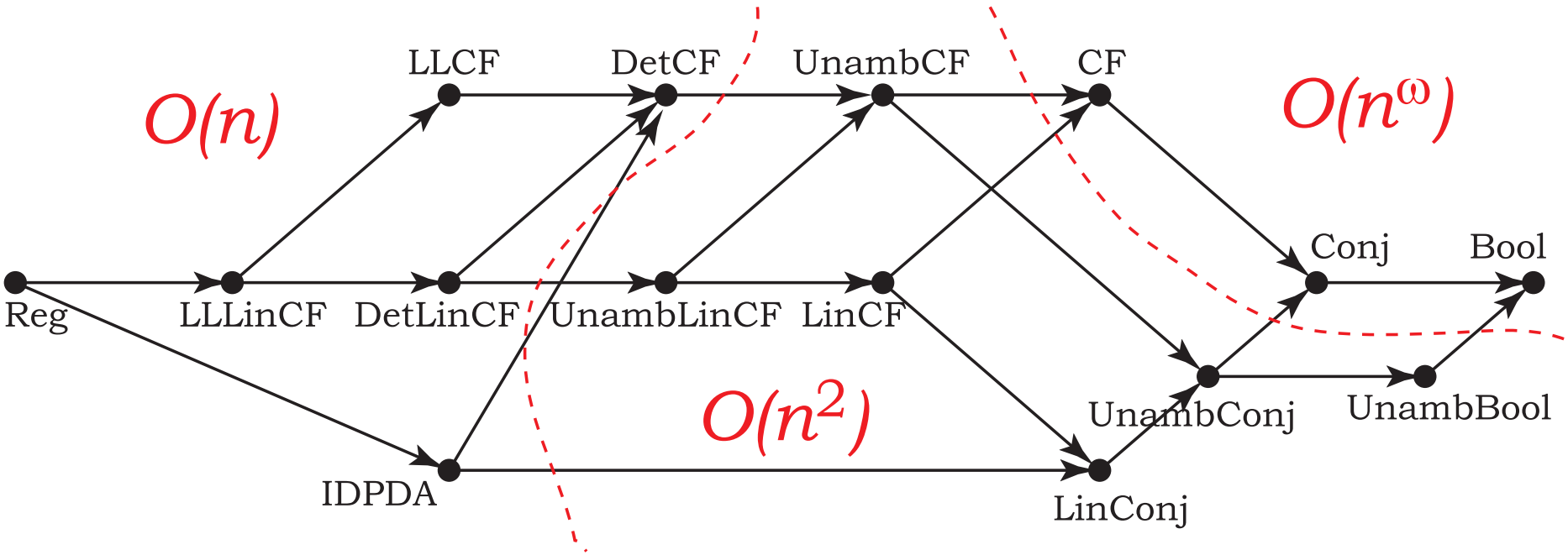
Пусть $L \in \mathcal{L}$. Для данной строки w, определить, принадлежит ли она L.



Время работы известных алгоритмов

Задача принадлежности для семейства языков $\mathcal{L}$

Пусть $L \in \mathcal{L}$. Для данной строки w, определить, принадлежит ли она L.

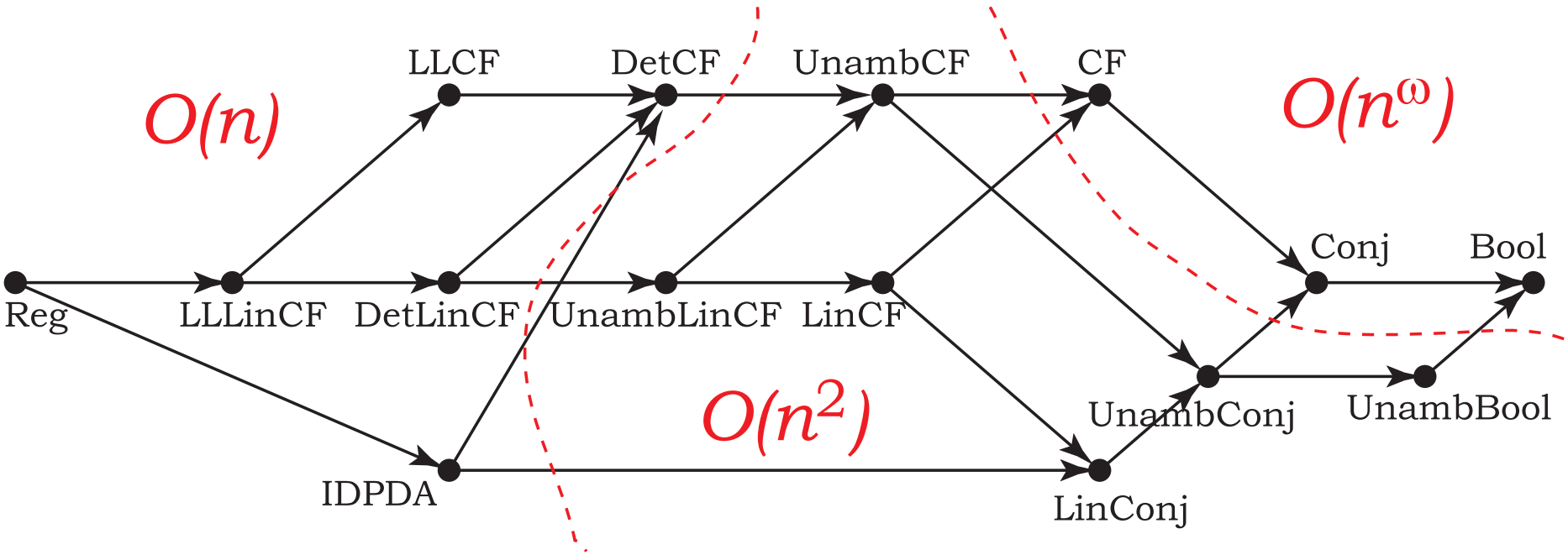


- Булевы грамматики общего вида: за время $O(\mathrm{BMM}(n) \log n)$ (Валиант, 1975; Охотин, 2010).

Время работы известных алгоритмов

Задача принадлежности для семейства языков $\mathcal{L}$

Пусть $L \in \mathcal{L}$. Для данной строки w, определить, принадлежит ли она L.

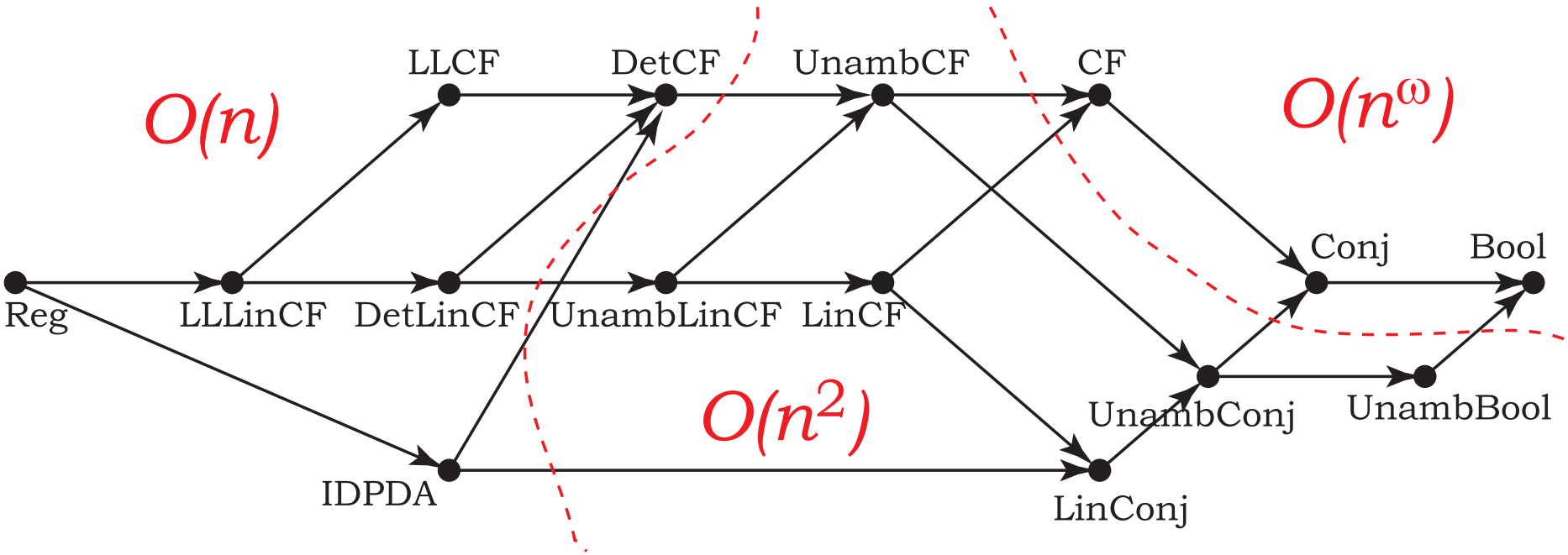


- Булевы грамматики общего вида: за время $O(\mathrm{BMM}(n) \log n)$ (Валиант, 1975; Охотин, 2010).
- Однозначные грамматики: за время $O(n^2)$ (Касами и Тории, 1967; Охотин, 2008).

Время работы известных алгоритмов

Задача принадлежности для семейства языков $\mathcal{L}$

Пусть $L \in \mathcal{L}$. Для данной строки w, определить, принадлежит ли она L.



- Булевы грамматики общего вида: за время $O(\mathrm{BMM}(n) \log n)$ (Валиант, 1975; Охотин, 2010).
- Однозначные грамматики: за время $O(n^2)$ (Касами и Тории, 1967; Охотин, 2008).
- LR(k)-грамматики (DetCF): за время $O(n)$ (Кнут, 1965).

NL-полнота линейных бесконтекстных грамматик

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$; NL-полная задача: достижимость в графе.

NL-полнота линейных бесконтекстных грамматик

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$; NL-полная задача: достижимость в графе.

Теорема (Сэдборо, 1975)

1. *LinCF* $\subseteq \mathrm{NL}$,
2. *LinCF содержит NL-полный язык.*

NL-полнота линейных бесконтекстных грамматик

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$; NL-полная задача: достижимость в графе.

Теорема (Сэдборо, 1975)

1. *LinCF* $\subseteq \mathrm{NL}$,
2. *LinCF содержит NL-полный язык.*

- Правила $A \to uBv$, разбор угадывается с двух концов.

NL-полнота линейных бесконтекстных грамматик

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$; NL-полная задача: достижимость в графе.

Теорема (Сэдборо, 1975)

1. *LinCF* $\subseteq \mathrm{NL}$,
2. *LinCF содержит NL-полный язык.*

- Правила $A \to uBv$, разбор угадывается с двух концов.
- Построение NL-полного языка: $f\left(\{w\$w^R \mid w \in \{a,b\}\}\right)$, где:

$$f(L) = \Big\{ [w_{1,1}\# \ldots \# w_{1,\ell_1}] \ldots [w_{k,1}\# \ldots \# w_{k,\ell_k}] \;\Big|\; \exists i_1, \ldots, i_k : \; w_{1,i_1} \ldots w_{k,i_k} \in L \Big\}$$

NL-полнота линейных бесконтекстных грамматик

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$; NL-полная задача: достижимость в графе.

Теорема (Сэдборо, 1975)

1. *LinCF* $\subseteq \mathrm{NL}$,
2. *LinCF содержит NL-полный язык.*

- Правила $A \to uBv$, разбор угадывается с двух концов.
- Построение NL-полного языка: $f\left(\{w\$w^R \mid w \in \{a,b\}\}\right)$, где:

$$f(L) = \Big\{ [w_{1,1}\# \ldots \# w_{1,\ell_1}] \ldots [w_{k,1}\# \ldots \# w_{k,\ell_k}] \;\Big|\; \exists i_1, \ldots, i_k : \; w_{1,i_1} \ldots w_{k,i_k} \in L \Big\}$$

- Кодирование достижимости в графе:

$[a^1 b]$

$[\#a^1 b a^{t_{1,1}} b\# \ldots \#a^1 b a^{t_{1,m_1}} b] \ldots$

$[a^n b\$]$ $\qquad\qquad$ $[\#ba^n ba^n] \ldots [\#ba^1 ba^1]$

L-полнота линейных детерминированных грамматик

- $\mathrm{L} = \mathrm{DSPACE}(\log n)$;
  L-полная задача: достижимость в графе со степенью исхода 1.

L-полнота линейных детерминированных грамматик

- $\mathrm{L} = \mathrm{DSPACE}(\log n)$;
  L-полная задача: достижимость в графе со степенью исхода 1.
- *DetLinCF*: линейные LR(1) грамматики.

L-полнота линейных детерминированных грамматик

- $\mathrm{L} = \mathrm{DSPACE}(\log n)$;
  L-полная задача: достижимость в графе со степенью исхода 1.
- *DetLinCF*: линейные LR(1) грамматики.

Теорема (Хольцер, Ланге, 1993)

1. *DetLinCF* $\subseteq \mathrm{L}$,
2. *DetLinCF содержит L-полный язык.*

L-полнота линейных детерминированных грамматик

- $\mathrm{L} = \mathrm{DSPACE}(\log n)$;
  L-полная задача: достижимость в графе со степенью исхода 1.
- *DetLinCF*: линейные LR(1) грамматики.

Теорема (Хольцер, Ланге, 1993)

1. *DetLinCF* $\subseteq \mathrm{L}$,
2. *DetLinCF содержит L-полный язык.*

- Уточнение предыдущего доказательства.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.
  3. $y_{A,i,j;E,s,t} \& y_{E,s,t;D,k,\ell} \to y_{A,i,j;D,k,\ell}$.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.
  3. $y_{A,i,j;E,s,t} \& y_{E,s,t;D,k,\ell} \to y_{A,i,j;D,k,\ell}$.
- ✓ Удвоение длины строки.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ $(a_{i+1} \dots a_j \in L_G(A))$,
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.
  3. $y_{A,i,j;E,s,t} \& y_{E,s,t;D,k,\ell} \to y_{A,i,j;D,k,\ell}$.
- ✓ Удвоение длины строки.
- Высота $O(\log^2 n)$, $O(n^6)$ элементов.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.
  3. $y_{A,i,j;E,s,t} \& y_{E,s,t;D,k,\ell} \to y_{A,i,j;D,k,\ell}$.
- ✓ Удвоение длины строки.
- Высота $O(\log^2 n)$, $O(n^6)$ элементов.
  - Можно улучшить до $O(\mathrm{BMM}(n^2))$ элементов.

Разбор обыкновенных бесконтекстных грамматик в NC^2

- NC^2: схемы высоты $O(\log^2 n)$ с $n^{O(1)}$ элементов.

Теорема (Руззо, 1981; Брент и Гольдшлягер, 1984; Риттер, 1985)

$CF \subseteq \mathrm{NC}^2$.

- Основные элементы: $x_{A,i,j}$ ($a_{i+1} \dots a_j \in L_G(A)$),
  $y_{A,i,j;D,k,\ell}$ (если $a_{k+1} \dots a_\ell \in L_G(D)$, то $a_{i+1} \dots a_j \in L_G(A)$).
  1. $x_{B,i,k} \& x_{C,k,j} \to x_{A,i,j}$ (для правила $A \to BC$)
  2. $y_{A,i,j;D,k,\ell} \& x_{D,k,\ell} \to x_{A,i,j}$.
  3. $y_{A,i,j;E,s,t} \& y_{E,s,t;D,k,\ell} \to y_{A,i,j;D,k,\ell}$.
- ✓ Удвоение длины строки.
- Высота $O(\log^2 n)$, $O(n^6)$ элементов.
  - Можно улучшить до $O(\mathrm{BMM}(n^2))$ элементов.
- $\mathrm{NC}^2 \subseteq \mathrm{DSPACE}(\log^2 n) = \mathrm{DTIMESPACE}(n^{\log n}, \log^2 n)$

Разбор детерминированных бесконтекстных грамматик

- $\mathrm{SC}^2 = \mathrm{DTIMESPACE}(\mathit{poly}, \log^2 n)$

Теорема (Кук, 1979)

$\mathit{DetCF} \subseteq \mathrm{SC}^2$.

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

Теорема (Ибарра и Ким, 1984; Охотин, 2003)

1. *Bool* $\subseteq \mathrm{P}$,
2. *LinConj содержит P-полный язык.* *(покажем для Bool)*

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

Теорема (Ибарра и Ким, 1984; Охотин, 2003)

1. *Bool* $\subseteq \mathrm{P}$,
2. *LinConj содержит P-полный язык.* *(покажем для Bool)*

- Принадлежность P: обычным алгоритмом разбора.

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

Теорема (Ибарра и Ким, 1984; Охотин, 2003)

1. *Bool* $\subseteq \mathrm{P}$,
2. *LinConj содержит P-полный язык.* *(покажем для Bool)*

- Принадлежность P: обычным алгоритмом разбора.
- Построение P-полного языка: CVP с элементом $x_i = \neg x_{i-1} \wedge \neg x_{j_i}$.

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

Теорема (Ибарра и Ким, 1984; Охотин, 2003)

1. *Bool* $\subseteq \mathrm{P}$,
2. *LinConj содержит P-полный язык.* *(покажем для Bool)*

- Принадлежность P: обычным алгоритмом разбора.
- Построение P-полного языка: CVP с элементом $x_i = \neg x_{i-1} \wedge \neg x_{j_i}$.
- Схема представляется строкой $a^{n-1-j_n} b\, a^{n-2-j_{n-1}} b \ldots a^{2-j_3} b\, a^{1-j_2} b$.

P-полнота булевых грамматик

- $\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{DTIME}(n^k)$
- P-полная задача: значение булевой схемы.

Теорема (Ибарра и Ким, 1984; Охотин, 2003)

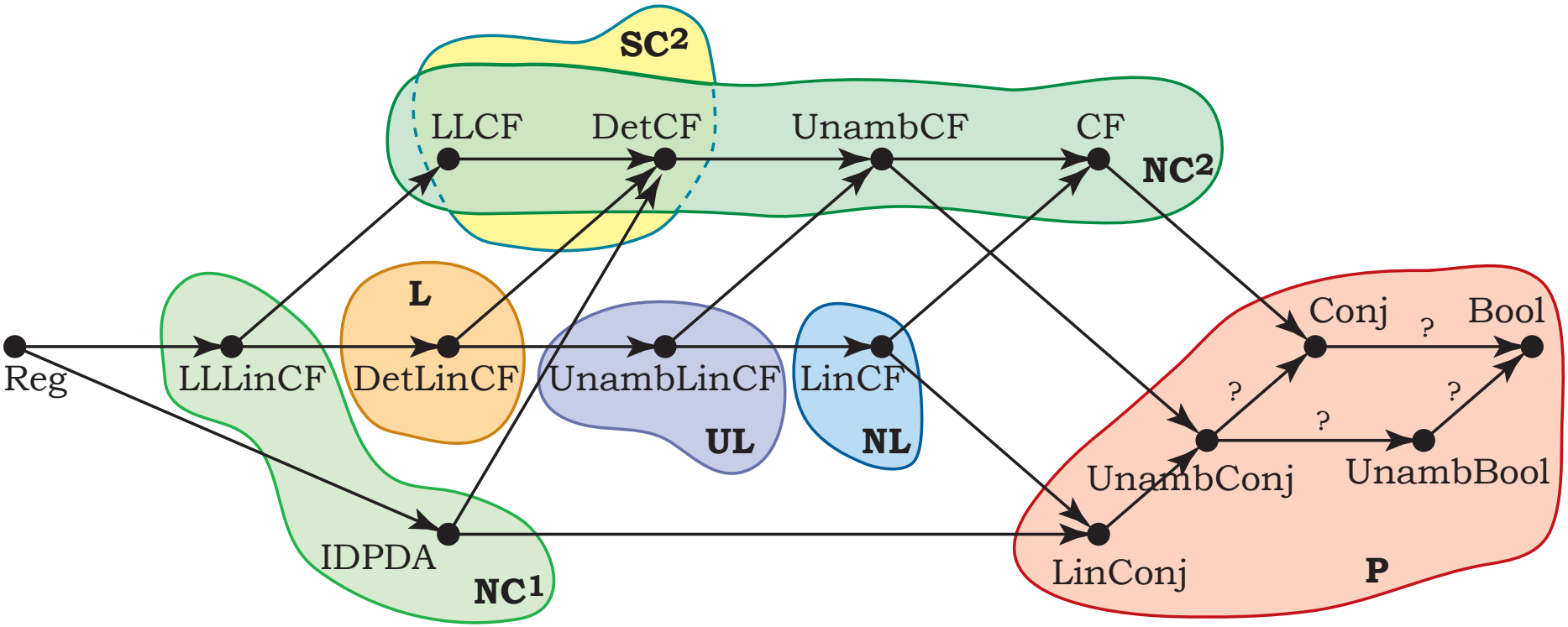
1. *Bool* $\subseteq \mathrm{P}$,
2. *LinConj содержит P-полный язык.* *(покажем для Bool)*

- Принадлежность P: обычным алгоритмом разбора.
- Построение P-полного языка: CVP с элементом $x_i = \neg x_{i-1} \wedge \neg x_{j_i}$.
- Схема представляется строкой $a^{n-1-j_n} b\, a^{n-2-j_{n-1}} b \ldots a^{2-j_3} b\, a^{1-j_2} b$.
- Грамматика:

$$\begin{array}{lcl} S & \to & \neg AbS\&\neg CS \\ A & \to & aA \mid \varepsilon \\ C & \to & aCAb \mid b \end{array}$$

Сравнение семейств языков

$$\mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{UL} \subseteq \mathrm{NL} \subseteq \mathrm{NC}^2 \subseteq \ldots \subseteq \mathrm{NC}^k \subseteq \ldots \subseteq P$$



Часть IV

Проверка свойств данной грамматики

Верхние оценки

Общая задача принадлежности для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данной строки w, определить, принадлежит ли w яаыку $L(G)$.

Верхние оценки

Общая задача принадлежности для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данной строки w, определить, принадлежит ли w яаыку $L(G)$.

Задача пустоты для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$, определить, пуст ли язык $L(G)$.

Верхние оценки

Общая задача принадлежности для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данной строки w, определить, принадлежит ли w яаыку $L(G)$.

Задача пустоты для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$, определить, пуст ли язык $L(G)$.

Задача равенства регулярному языку для $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данного (или фиксированного) $L_0 \in Reg$, определить, совпадает ли $L(G)$ с L_0.

Верхние оценки

Общая задача принадлежности для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данной строки w, определить, принадлежит ли w яаыку $L(G)$.

Задача пустоты для семейства грамматик $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$, определить, пуст ли язык $L(G)$.

Задача равенства регулярному языку для $\mathcal{G}$

Для данной грамматики $G \in \mathcal{G}$ и для данного (или фиксированного) $L_0 \in Reg$, определить, совпадает ли $L(G)$ с L_0.

Задача равенства (включения) для $\mathcal{G}$

Для данных грамматик $G_1, G_2 \in \mathcal{G}$, определить, верно ли, что $L(G_1) = L(G_2)$ ($L(G_1) \subseteq L(G_2)$, соответственно).

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.
  - Было: $x_i = 1$; стало: $A_i \to \varepsilon$.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.
  - Было: $x_i = 1$; стало: $A_i \to \varepsilon$.
  - Было: $x_i = x_j \vee x_k$; стало: $A_i \to A_j \mid A_k$.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.
  - Было: $x_i = 1$; стало: $A_i \to \varepsilon$.
  - Было: $x_i = x_j \vee x_k$; стало: $A_i \to A_j \mid A_k$.
  - Было: $x_i = x_j \wedge x_k$; стало: $A_i \to A_j A_k$.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.
  - Было: $x_i = 1$; стало: $A_i \to \varepsilon$.
  - Было: $x_i = x_j \vee x_k$; стало: $A_i \to A_j \mid A_k$.
  - Было: $x_i = x_j \wedge x_k$; стало: $A_i \to A_j A_k$.
  - Грамматика может оказаться неоднозначной.

Общая задача принадлежности

Теорема

1. *Общая задача принадлежности для Bool принадлежит P.*
2. *Она P-полна уже для LLCF и для строки ε.*

- Принадлежность P: обычным алгоритмом разбора.
- P-полнота для *CF*: сведением MCVP (значение монотонной схемы).
  - Было: элемент x_i; стало: $A_i \in N$.
  - Было: $x_i = 1$; стало: $A_i \to \varepsilon$.
  - Было: $x_i = x_j \vee x_k$; стало: $A_i \to A_j \mid A_k$.
  - Было: $x_i = x_j \wedge x_k$; стало: $A_i \to A_j A_k$.
  - Грамматика может оказаться неоднозначной.
- Построение LL(1) грамматики: прямым кодированием машины Тьюринга.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$
для данной G *из* $UnambCF$ *и для данного регулярного языка* L_0.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) =$$

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) = f_B(x) f_C(x) +$$

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) = f_B(x) f_C(x) + x^2 f_D(x) +$$

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) = f_B(x) f_C(x) + x^2 f_D(x) + 1$$

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) = f_B(x) f_C(x) + x^2 f_D(x) + 1$$

- Решение — алгебраическая функция; раскладывается в ряд.

UnambCF: равенство регулярному языку

Теорема (Саломаа и Сойттола, 1978)

Существует алгоритм проверки равенства $L(G) = L_0$ для данной G из UnambCF и для данного регулярного языка L_0.

- Проверить пустоту $L(G) \cap \overline{L_0}$ (легко) и равенство $L(G) \cap L_0 = L_0$.
- a_n^L: количество строк длины n в языке L.
- $a_n^{K \cup L} \leqslant a_n^K + a_n^L$; если $K \cap L = \varnothing$, то $a_n^{K \cup L} = a_n^K + a_n^L$.
- Если сцепление $K \cdot L$ однозначно, то $a_n^{KL} = \sum_{i=0}^{n} a_i^K \cdot a_{n-i}^L$.
- Формальный степенной ряд: $f_L(x) = \sum_{n=0}^{\infty} a_n^L x^n$.
- Языковые уравнения $\to$ функциональные уравнения.

$$A = BC \cup aDb \cup \{\varepsilon\} \qquad f_A(x) = f_B(x) f_C(x) + x^2 f_D(x) + 1$$

- Решение — алгебраическая функция; раскладывается в ряд.
- Использовать разрешимость элементарного анализа.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2; определить, равны ли $L(G_1)$ и $L(G_2)$.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

Теорема (Розенкранц, Стирнс, 1970; Ольшанский, Пнуэли, 1977)

Задача равенства двух LLCF языков разрешима.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

Теорема (Розенкранц, Стирнс, 1970; Ольшанский, Пнуэли, 1977)

Задача равенства двух LLCF языков разрешима.

Теорема (Валиант, 1974)

Задача равенства двух DetLinCF языков разрешима.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

Теорема (Розенкранц, Стирнс, 1970; Ольшанский, Пнуэли, 1977)

Задача равенства двух LLCF языков разрешима.

Теорема (Валиант, 1974)

Задача равенства двух DetLinCF языков разрешима.

- По двум DPDA строится третий, распознающий их различия.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

Теорема (Розенкранц, Стирнс, 1970; Ольшанский, Пнуэли, 1977)

Задача равенства двух LLCF языков разрешима.

Теорема (Валиант, 1974)

Задача равенства двух DetLinCF языков разрешима.

- По двум DPDA строится третий, распознающий их различия.

Теорема (Сенизерг, 1997)

Задача равенства двух DetCF языков разрешима.

DetCF: проверка на равенство

- Даны две LR(k) грамматики G_1, G_2;
  определить, равны ли $L(G_1)$ и $L(G_2)$.
- Детерминированные магазинные автоматы (DPDA).

Теорема (Розенкранц, Стирнс, 1970; Ольшанский, Пнуэли, 1977)

Задача равенства двух LLCF языков разрешима.

Теорема (Валиант, 1974)

Задача равенства двух DetLinCF языков разрешима.

- По двум DPDA строится третий, распознающий их различия.

Теорема (Сенизерг, 1997)

Задача равенства двух DetCF языков разрешима.

- Верхняя оценка сложности: примитивно рекурсивна (Стирлинг, 2002).

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.
- $C_i = \alpha q a \beta$ — положение дел на i-м шаге обработки w.

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.
- $C_i = \alpha q a \beta$ — положение дел на i-м шаге обработки w.
- $C_M(w) = C_0 \# \ldots \# C_{n-1} \$ C_n \$ \left(C_n\right)^R \# \ldots \# \left(C_1\right)^R \# \left(C_0\right)^R$.

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.
- $C_i = \alpha q a \beta$ — положение дел на i-м шаге обработки w.
- $C_M(w) = C_0 \# \ldots \# C_{n-1} \$ C_n \$ \left(C_n\right)^R \# \ldots \# \left(C_1\right)^R \# \left(C_0\right)^R$.
- $\mathrm{VALC}(M) = \{w \natural C_M(w) \mid w \in L(M)\} \subseteq \Sigma^* \natural \Gamma^*$

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.
- $C_i = \alpha q a \beta$ — положение дел на i-м шаге обработки w.
- $C_M(w) = C_0 \# \ldots \# C_{n-1} \$ C_n \$ \left(C_n\right)^R \# \ldots \# \left(C_1\right)^R \# \left(C_0\right)^R$.
- $\mathrm{VALC}(M) = \{w \natural C_M(w) \mid w \in L(M)\} \subseteq \Sigma^* \natural \Gamma^*$

Лемма

$\mathrm{VALC}(M) = L_1 \cap L_2$ *для* $L_1, L_2, \overline{L_1}, \overline{L_2} \in LLLinCF$.

Неразрешимость задачи включения

- Машина Тьюринга M с входным алфавитом Σ кодирование её успешных вычислений: $C_M : L(M) \to \Gamma^*$.
- $C_i = \alpha q a \beta$ — положение дел на i-м шаге обработки w.
- $C_M(w) = C_0 \# \ldots \# C_{n-1} \$ C_n \$ \left(C_n\right)^R \# \ldots \# \left(C_1\right)^R \# \left(C_0\right)^R$.
- $\mathrm{VALC}(M) = \{w \natural C_M(w) \mid w \in L(M)\} \subseteq \Sigma^* \natural \Gamma^*$

Лемма

$\mathrm{VALC}(M) = L_1 \cap L_2$ *для* $L_1, L_2, \overline{L_1}, \overline{L_2} \in$ *LLinCF*.

Теорема

Задача включения неразрешима для LLinCF.

Равенство регулярному языку

Теорема (Бар-Хиллель, Перлес, Шамир, 1961)

1. *Для бесконтекстных грамматик разрешимо равенство* $\varnothing$,

Равенство регулярному языку

Теорема (Бар-Хиллель, Перлес, Шамир, 1961)

1. *Для бесконтекстных грамматик разрешимо равенство* $\varnothing$,
2. *. . . но неразрешимо равенство* Σ^*.

Равенство регулярному языку

Теорема (Бар-Хиллель, Перлес, Шамир, 1961)

1. *Для бесконтекстных грамматик разрешимо равенство* $\varnothing$,
2. *. . . но неразрешимо равенство* Σ^*.

- Распространяется на *LinCF*.

Равенство регулярному языку

Теорема (Бар-Хиллель, Перлес, Шамир, 1961)

1. *Для бесконтекстных грамматик разрешимо равенство* $\varnothing$,
2. *. . . но неразрешимо равенство* Σ^*.

- Распространяется на *LinCF*.
- Равносильность двух грамматик тоже неразрешима.

Равенство регулярному языку

Теорема (Бар-Хиллель, Перлес, Шамир, 1961)

1. *Для бесконтекстных грамматик разрешимо равенство* $\varnothing$,
2. *. . . но неразрешимо равенство* Σ^*.

- Распространяется на *LinCF*.
- Равносильность двух грамматик тоже неразрешима.

Теорема (Еж, Охотин, 2007)

Для конъюнктивных грамматик неразрешимо равенство всякому конъюнктивному языку.

Часть V

Дальнейшие исследования

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?
5. Найти языки, не принадлежащие *Bool*.

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?
5. Найти языки, не принадлежащие *Bool*.
   - Только из $\mathrm{DTIMESPACE}(n^2, n)$, чтобы без тривиальных ответов.

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?
5. Найти языки, не принадлежащие *Bool*.
   - Только из $\mathrm{DTIMESPACE}(n^2, n)$, чтобы без тривиальных ответов.
   - Хотя бы не принадлежащие *UnambConj*.

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?
5. Найти языки, не принадлежащие *Bool*.
   - Только из $\mathrm{DTIMESPACE}(n^2, n)$, чтобы без тривиальных ответов.
   - Хотя бы не принадлежащие *UnambConj*.
6. Замкнуты ли *UnambLinCF* относительно дополнения?

Над чем можно поработать?

1. Можно ли разобрать *Bool*, используя память $o(n)$?
2. Можно ли разобрать *UnambLinCF* за время $o(n^2)$?
3. Какова сложность проверки равенства *UnambCF* регулярному языку?
4. Разрешима ли равносильность однозначных бесконтекстных грамматик (*UnambCF*)?
   - ...хотя бы в линейном случае (*UnambLinCF*)?
5. Найти языки, не принадлежащие *Bool*.
   - Только из $\mathrm{DTIMESPACE}(n^2, n)$, чтобы без тривиальных ответов.
   - Хотя бы не принадлежащие *UnambConj*.
6. Замкнуты ли *UnambLinCF* относительно дополнения?
7. Замкнуты ли *Conj* или *UnambConj* относительно дополнения?