Теория формальных языков и трансляций

Языки, КС языки

В. С. Полозов

Кафедра системого программирования СПбГУ

Теория автоматов и формальных языков

Содержание

Язык

Что такое язык?

Алфавит

Определение

Алфавитом называется произвольное *конечное* множество слов.

Пример

- Цифры $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Множество $\mathbb{N}$

Обозначения:

- Символы: a, b, c...
- Цепочки: u, v, w... α, β, γ...
- Алфавит: Σ

Алфавит

Определение

Алфавитом называется произвольное *конечное* множество слов.

Пример

- Цифры $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Множество $\mathbb{N}$

Обозначения:

- Символы: a, b, c...
- Цепочки: u, v, w... α, β, γ...
- Алфавит: Σ

Алфавит

Определение

Алфавитом называется произвольное *конечное* множество слов.

Пример

- Цифры $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Множество $\mathbb{N}$

Обозначения:

- Символы: a, b, c...
- Цепочки: u, v, w... α, β, γ...
- Алфавит: Σ

Операции над алфавитом

- Σ^* – множество всех цепочек над алфавитом Σ
- Слово длины 0 – *пустая цепочка* ε
- Без пустой цепочки: $\Sigma^+ = \Sigma \setminus \{\varepsilon\}$
- Операция *конкатенации*

$$w_1 w_2 = (a_1, ..., a_n)(b_1, ..., b_m) = (a_1, ..., a_n, b_1, ..., b_m)$$

- Операция конкатенации индуцирует оператор степени:

$$w^i = \begin{cases} \varepsilon, & \text{если } i = 0 \\ w^{i-1} w, & \text{если } i > 0 \end{cases}$$

- Существует взаимно-однозначное отображение $\Sigma^* \to \mathbb{N}$
- Следовательно, множество Σ^* - счётно.

Операции над алфавитом

- Σ^* – множество всех цепочек над алфавитом Σ
- Слово длины 0 – *пустая цепочка* ε
- Без пустой цепочки: $\Sigma^+ = \Sigma \setminus \{\varepsilon\}$
- Операция *конкатенации*

$$w_1 w_2 = (a_1, ..., a_n)(b_1, ..., b_m) = (a_1, ..., a_n, b_1, ..., b_m)$$

- Операция конкатенации индуцирует оператор степени:

$$w^i = \begin{cases} \varepsilon, & \text{если } i = 0 \\ w^{i-1} w, & \text{если } i > 0 \end{cases}$$

- Существует взаимно-однозначное отображение $\Sigma^* \to \mathbb{N}$
- Следовательно, множество Σ^* - счётно.

Операции над алфавитом

- Σ^* – множество всех цепочек над алфавитом Σ
- Слово длины 0 – *пустая цепочка* ε
- Без пустой цепочки: $\Sigma^+ = \Sigma \setminus \{\varepsilon\}$
- Операция *конкатенации*

$$w_1 w_2 = (a_1, ..., a_n)(b_1, ..., b_m) = (a_1, ..., a_n, b_1, ..., b_m)$$

- Операция конкатенации индуцирует оператор степени:

$$w^i = \begin{cases} \varepsilon, & \text{если } i = 0 \\ w^{i-1} w, & \text{если } i > 0 \end{cases}$$

- Существует взаимно-однозначное отображение $\Sigma^* \to \mathbb{N}$
- Следовательно, множество Σ^* - счётно.

Язык

Определение

Языком называется произвольное подмножество Σ^*

Алгоритм

Определение

Алгоритм – конечный набор правил, задающий порядок выполнения операций при решении конкретной задачи. Для алгоритма характерны следующие 5 особенностей:

1. *конечность* – завершается после конечного числа шагов;
2. *определенность* – однозначно определен для любого входа;
3. *ввод* – конечные исходные данные;
4. *вывод* – конечный результат;
5. *эффективность* – выполнение любой операции за конечное время.

Процедура

Процедура – определяется аналогично алгоритму, за исключением свойства конечности: на некоторых входных данных процедура может не завершаться.

Связь языков и процедур

Определение

Связь между понятиями языка и процедуры задается с помощью *распознающей процедуры* – процедуры, в качестве входа принимающей элемент Σ^*, а на выходе – булевское значение.

Двойственное понятие – порождающая процедура.

Теорема

По данной порождающей процедуре f_g для языка $L \subset \Sigma^*$ можно построить распознающую процедуру f_r для этого же языка.

Теорема

По данной распознающей процедуре f_r для языка $L \subset \Sigma^*$ можно построить порождающую процедуру f_g для этого же языка.

Грамматика Хомского

Определение

Грамматикой Хомского называется формальная система $G = (V_N, V_T, P, S)$, где V_N – алфавит нетерминалов, V_T – алфавит терминалов,
$P = \{\alpha \to \beta : \alpha \in (V_N \cup V_T)^* \setminus V_T^*, \beta \in (V_N \cup V_T)^*\}$ – конечное множество правил, S – начальный нетерминал.

Определение

$V = V_N \cup V_T$ называется *объединенным алфавитом*.

Отношение выводимости

Определение

Множество правил на V^* индуцирует отношение *непосредственной выводимости*:

$$\delta_1 \Rightarrow \delta_2, \text{ если} \exists \gamma_1, \gamma_2 : \delta_1 = \gamma_1 \alpha \gamma_2 \wedge \delta_2 = \gamma_1 \beta \gamma_2 \wedge \alpha \to \beta \in P$$

Говорят также, что δ_2 *непосредственно выводится* из δ_1

Определение

Отношение *выводимости* ($\overset{*}{\Rightarrow}$) – транзитивное замыкание отношения непосредственной выводимости.

Грамматика и язык

Определение

Языком, *порождаемым грамматикой* G, называется

$$L(G) = \{w \in V_T : S \overset{*}{\Rightarrow} w\}$$

Классификация грамматик Хомского

Название	Ограничения
тип 0	-
тип 1 (КЗ)	$\alpha \to \beta \Rightarrow \lvert\alpha\rvert < \lvert\beta\rvert$
тип 2 (КС)	$P = \{A \to \beta : A \in V_N, \beta \in V^*\}$
тип 3 (регулярные)	$P = \{A \to aB : A, B \in V_N, a \in V_T\}$
(конечные)	$P = \{A \to u : A \in V_N, u \in V_T^*\}$

Замечание

Отечественное название КЗ – *неукорачивающая грамматика*

Замечание

Альтернативное определение КЗ – *грамматика непосредственных составляющих* – $P = \{\beta A \gamma \to \beta \alpha \gamma : A \in V_N, \alpha, \beta, \gamma \in V^*, \alpha \neq \varepsilon\}$

Контекстно-свободные грамматики

Определение

Контекстно-свободным языком называется язык, порождаемый контекстно-свободной грамматикой.

Дерево вывода

Определение

Деревом вывода в КС-грамматике называется дерево, каждый узел которого помечен символом грамматики (терминалом или нетерминалом), причем если какой-то узел в грамматике помечен символом A, а его дочерние узлы помечены $X_1, ..., X_n$, то в грамматике должно присутствовать правило $A \to X_1, ..., X_n$. Корень дерева должен быть помечен начальным символом грамматики.

Если все листья помечены терминальными символами, то расположив их в порядке следования слева направо, можно получить некоторую терминальную цепочку w. Говорят, что данное дерево вывода является деревом вывода этой терминальной цепочки.

Дерево вывода

Теорема

Пусть $G = (V_N, V_T, P, S)$ – КС-грамматика. Терминальная цепочка w выводима в G тогда и только тогда, когда существует ее дерево вывода.

Замечание

На практике – мы получили ещё один механизм определения принадлежности цепочки КС-языку: проверка возможности построения ее дерева вывода.
Более того, на практике именно дерево вывода и имеет ценность при грамматическом анализе цепочки.
Более того, любой вывод является обходом некоторого дерева вывода.

Дерево вывода

Теорема

Пусть $G = (V_N, V_T, P, S)$ – КС-грамматика. Терминальная цепочка w выводима в G тогда и только тогда, когда существует ее дерево вывода.

Замечание

На практике – мы получили ещё один механизм определения принадлежности цепочки КС-языку: проверка возможности построения ее дерева вывода.
Более того, на практике именно дерево вывода и имеет ценность при грамматическом анализе цепочки.
Более того, любой вывод является обходом некоторого дерева вывода.

Отношения порядка на узлах дерева вывода

Связь между выводом и деревом вывода ...

Определение

Для двух узлов A и B с общим родителем X будем говорить, что A *левее* B, если соответствующее правило имеет вид $X \to \alpha A \beta B \gamma$

Определение можно расширить для поддеревьев и для любых пар узлов в дереве. Аналогично определяется отношение *«правее»*.

Левосторонний вывод

Определение

Левосторонний вывод – вывод, соответствующий левостороннему обходу дерева: на каждом шаге «раскрывается» самый левый нетерминал в сентенциальной форме.

Замечание

Отношение левосторонней выводимости обозначается: $\alpha \underset{lm}{\Longrightarrow} \beta$

Аналогично определяется *правосторонний вывод* $(\alpha \underset{rm}{\Longrightarrow} \beta)$

Предмет синтаксического разбора

Вход: грамматика $G = (V_N, V_T, P, S)$, цепочка $\alpha \in V_T^*$
Выход:

- Порождающее дерево (деревья)
- $\{\texttt{да} \parallel \texttt{нет}\}$

Семантика и неоднозначность?

Предмет синтаксического разбора

Вход: грамматика $G = (V_N, V_T, P, S)$, цепочка $\alpha \in V_T^*$
Выход:

- Порождающее дерево (деревья)
- $\{\text{да} \parallel \texttt{нет}\}$

Семантика и неоднозначность?

Классификация способов синтаксического разбора

Все методы делятся по способу построения дерева разбора

- Нисходящие[1]
- Восходящие

Рассмотрим на примере грамматики для языка $a^n b^n c^n$:

$$S \to aSQ$$

$$S \to abc$$

$$bQc \to bbcc$$

$$cQ \to Qc$$

и входной цепочки $aabbcc$

[1] Деревья растут сверху-вниз.

Классификация способов синтаксического разбора

Все методы делятся по способу построения дерева разбора

- Нисходящие[1]
- Восходящие

Рассмотрим на примере грамматики для языка $a^n b^n c^n$:

$$S \to aSQ$$

$$S \to abc$$

$$bQc \to bbcc$$

$$cQ \to Qc$$

и входной цепочки $aabbcc$

[1] Деревья растут сверху-вниз.

Классификация способов синтаксического разбора

Все методы делятся по способу построения дерева разбора

- Нисходящие[1]
- Восходящие

Рассмотрим на примере грамматики для языка $a^n b^n c^n$:

$$S \to aSQ$$

$$S \to abc$$

$$bQc \to bbcc$$

$$cQ \to Qc$$

и входной цепочки $aabbcc$

[1] Деревья растут сверху-вниз.

Синтаксический разбор по типу грамматики

Сложность: $O(C^n)$, $O(n^p)$, $O(n)$

- Тип 0 - проблема распознавания не разрешима.
- Тип 1 - порождающий NDA. Эффективность?
- Тип 2 (KC) - Основные результаты по построению синтаксических анализаторов.
- Тип 3 - нет разницы между lm и rm. Обычно эффективны нисходящие методы.
- Тип 4 - конечные языки.

Синтаксический разбор по типу грамматики

Сложность: $O(C^n)$, $O(n^p)$, $O(n)$

- Тип 0 - проблема распознавания не разрешима.
- Тип 1 - порождающий NDA. Эффективность?
- Тип 2 (KC) - Основные результаты по построению синтаксических анализаторов.
- Тип 3 - нет разницы между lm и rm. Обычно эффективны нисходящие методы.
- Тип 4 - конечные языки.

Синтаксический разбор по типу грамматики

Сложность: $O(C^n)$, $O(n^p)$, $O(n)$

- Тип 0 - проблема распознавания не разрешима.
- Тип 1 - порождающий NDA. Эффективность?
- Тип 2 (KC) - Основные результаты по построению синтаксических анализаторов.
- Тип 3 - нет разницы между lm и rm. Обычно эффективны нисходящие методы.
- Тип 4 - конечные языки.

Синтаксический разбор по типу грамматики

Сложность: $O(C^n)$, $O(n^p)$, $O(n)$

- Тип 0 - проблема распознавания не разрешима.
- Тип 1 - порождающий NDA. Эффективность?
- Тип 2 (KC) - Основные результаты по построению синтаксических анализаторов.
- Тип 3 - нет разницы между lm и rm. Обычно эффективны нисходящие методы.
- Тип 4 - конечные языки.

Синтаксический разбор по типу грамматики

Сложность: $O(C^n)$, $O(n^p)$, $O(n)$

- Тип 0 - проблема распознавания не разрешима.
- Тип 1 - порождающий NDA. Эффективность?
- Тип 2 (KC) - Основные результаты по построению синтаксических анализаторов.
- Тип 3 - нет разницы между lm и rm. Обычно эффективны нисходящие методы.
- Тип 4 - конечные языки.

•	Сверху-вниз	Снизу-вверх
Ненаправленные	Метод Ангера	CYK
Направленные	Автомат predict/match DFS (с откатами) BFS (Грейбах) Рекурсивный спуск DC грамматики	Автомат shift/reduce DFS (с откатами) BFS Ограниченный BFS (Earley)
Линейные направленные: поиск в ширину (не более 1)	LL(k)	приоритеты bounded-context LR(k) LALR(1) SLR(1)
Эффективные общие направленные методы	???	Tomita

Вопросы