Unity. Precision. Perfection.

КОНСПЕКТ ЛЕКЦИЙ

по дисциплине «Теория автоматов и формальных языков»

Лектор:	Вербицкая
Страниц:	10
Последнее обновление:	6 сентября 2019 г.
Автор:	Корытов Павел, 6304

Санкт-Петербург

2019

Содержание

1. Введение

Виды языков:

- Естественные — Русский, английский
- Искуственные
  - Эсперанто, ложбан
  - Клингонский, эльфийский
  - C++, Python, Java, C#, Haskell, OCaml, Perl, Coq, Agda

1.1. Синтаксис и семантика

- *Синтаксис* — правила построения программ из символов
- *Семантика* — правила истолкования программ

Пример 1.1. Язык арифметических выражений

$$1 \bullet (2 + 3)/4 - 5$$

- Синтаксис
  - *Терм* — последовательность цифр или любое выражение в скобках
  - *Слагаемое* , последовательность *термов* , соединенных знаками умножения и деление
  - *Выражение* — последовательность *слагаемых* , соединенных знаками сложения и вычитания (перед первым слагаемым может стоят минус)
- Семантика
  - Значение арифметического выражения
    * -3.75
    * -4
    * $\frac{-15}{4}$

1.2. Что такое язык? Теория множеств

Язык — множество строк

Множества

Множество — набор уникальных элементов

- $x \in X$: x — элемент множества X (x принадлежит X)
- $x \notin X$ — x не являеся элементом X
- Уникальность, неупорядоченность: $\{13, 42\} = \{42, 13\} = \{13, 42, 13\}$
- Универсальное множество (универсум) — множество всех мыслимых объектов
  - $[N] = \{1, 2, 3, \ldots\}$
  - $[Z] = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
  - $[Q] = m/n|m, n \in [Z]; n \neq 0$

Подмножества

A является *подмножеством* B тогда и только тогда, когда все элементы A являеются элементами B.

$$A \subseteq B \iff \forall x : x \in A \Rightarrow x \in B$$

Пустое множество ($\emptyset$) — мноэество без элементов

- $\forall x : x \notin \emptyset$
- $\forall A : \emptyset \subseteq A$

Множества A и B *равны* тогда и только тогда, когда A является подмножеством B и B является подмножеством A

$$A = B \iff (A \subseteq B) \wedge (B \subseteq A)$$

A является *строгим подмножеством* B тогда и только тогда, когда A является подмножеством B, но они не равны друг другу

$$A \subset B \iff (A \subseteq B) \wedge A \neq B$$

- $\forall x : \emptyset \subset \{x\}$
- $\forall A : (A = A) \wedge A \not\subset B$

$2^A = \{B | B \subseteq A\}$ — *powerset*, множество всех подмножеств A

- $\forall A : \emptyset in 2^A$
- $\forall A : A \in 2^A$
- $A = \{0, 1\} \Rightarrow \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$

Операции над множествами

- *Объединение* : $A \cup B = \{x | (x \in A) \vee (x \in B)\}$
- *Пересечение* : $A \cap B = \{x | (x \in A) \wedge (x \in B)\}$
- *Разность* : A
  $B = x | (x \in A) \wedge (x \notin B)$
- *Дополнение* : $\bar{A} = x | (x \in u) \wedge (x \notin A) = u$
  A

Строки

Строка — последовательность символов

- *Алфавит* (Σ) — конечное множество (атомарных, неделимых символов)
- *Цепочка* — любая конечная последовательность символов алфавита — предложение, слово, строка, …
  ε — цепочка, не содержащая ни одного символа

Операции над строками

- Конкатенация строк α и β ($\alpha \bullet \beta = \alpha\beta$) — соединение строк
  - $\forall \alpha\beta\gamma : (\alpha \bullet \beta) \bullet \gamma = \alpha \bullet (\beta \bullet \gamma)$
  - $\forall \alpha : \alpha \bullet \varepsilon = \varepsilon \bullet \alpha = \alpha$
- *Обращение (реверс)* a^R — цепочка, символы которой записаны в обратном порядке
- n-я степень цепочки a^n — конкатенация n повторений цепочки
- $|a|$ — *длина строки* — количество составляющих её символов

Пример — арифметические выражения
Алфавит $\Sigma = \{0, 1, \ldots, 9, +, -, *, /, (,)\}$

1.3. Формальный язык

- Σ — алфавит
  - $\Sigma = \{0, 1\}$
- Σ^* — подмножетсов, содержаещее все цепочке в алфавите Σ, включая пустую цепочку
- $\Sigma^+ = \Sigma^* / \{\varepsilon\}$
- Формальный язык в алфавите Σ — подмножество множества всех цепочек в этом алфавите

  - Для любого языка L (в алфавите Σ) справедливо $L \subseteq \Sigma$
  - $L = \{0, 0, 000, \ldots\} \subset \{0, 1\}^*$
  - $L = \{0, 0101, \ldots\} \subset \{0, 1\}^*$

Метаязык — язык, на котором дано описание языка

- Естественный язык
- Язык металингвистических формул Бэкуса (БНФ)
- Синтаксические диаграммы
- Грамматики
- …

1.3.1. БНФ — Бэкуса-Наура форма

- *Символ* — элементарное понятие языка
  - $+$ — означает сложение в языке арифметических выражений
- *Метапеременная* — сложное понятие языка
  - Переменной <выражение> можно обозначить выражение
- *Формула*
  - <определяемый символ> := <посл 1> | … | <посл n>
  - В правой части формулы — альтернатива конкатенаций строк, составленных из символов и метапеременных
- Пример: число
  - <цифра> := | 0 | 1 | … | 9
  - <число> := <цифра>|<цифра><число>

1.3.2. Расширенная БНФ (EBNF)

- *Итерация*
  - $< x >:= \{< y >\}$ эквивалентно $< x >:= \varepsilon| < y >< x >$
- *Условное вхождение*
  - $< x >:= [< y >]$ эквивалентно $< x >:= \varepsilon| < y >$
- *Скобки для группировки*
  - $(< x > \mid < y >) < z >$ эквивалентно $< x >< z > \mid < y >< z >$

Пример 1.2. Арифметические выражения

$$< expr >:= [-] < factor > \{(+|-) < factor >\}$$

$$< factor >:=< term > \{(*|/) < term >\}$$

$$< term >:=< number > /'('< expr >')'$$

1.3.3. Синтаксические диаграммы Вирта

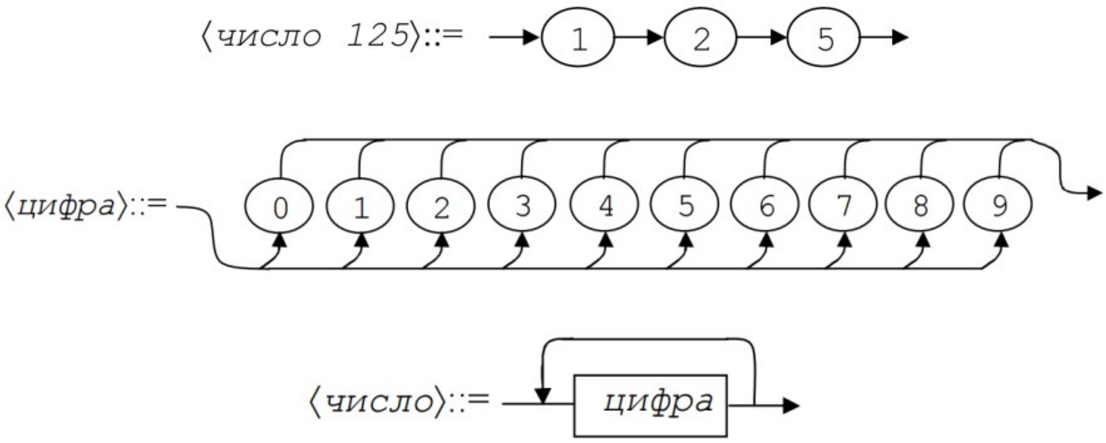

Рисунок 1. Синтаксические диаграммы Вирта

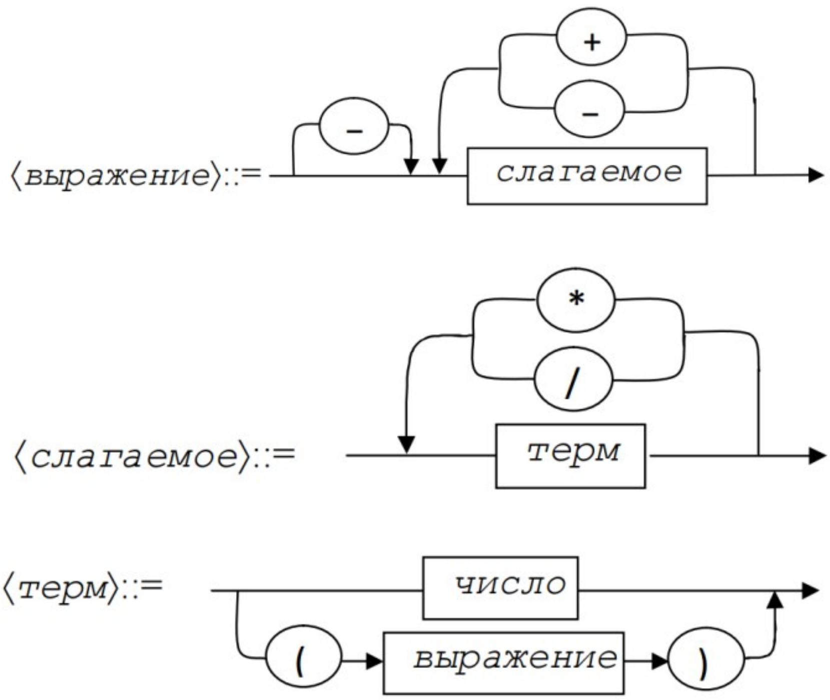

Рисунок 2. Синтаксические диаграммы Вирта

1.4. Формальная грамматика

- *Порождающая грамматика* G — это четверка $\{V_T, V_N, P, S\}$

- V_T — алфавит терминальных символов (*терминалов*)
- V_N — алфавит нетерминальных символов (*нетерминалов*)
  * $V_T \cap V_N = \emptyset$
  * $V : +V_T \cap V_N$
- P — конечное множество правил вида $\alpha \to \beta$
  * $\alpha \in V^* V_N V^*$
  * $\beta in V^*$
- S — начальный нетерминал грамматики
  * $S \in N$

Пример 1.3. Язык чисел в двоичной системе счисления

$$V_T = \{0, 1, -\}; V_n = \{S, N, A\}$$

- $S \to 0$
- $S \to N$
- $S \to -N$
- $N \to 1A$
- $A \to 0A$
- $A \to 1A$
- $A \to \varepsilon$

После преобразований:

- $S \to 0 \mid N \mid -N$
- $N \to 1A$
- $A \to 0A \mid 1A \mid \varepsilon$

Или так:

- $S \to 0 \mid [-]N$
- $N \to 1A$
- $A \to (0 \mid 1)A \mid \varepsilon$

Отношение непосредственной выводимости

- $\alpha \to \beta \in P$
- $\gamma, \delta \in V^*$
- $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$: $\gamma\beta\delta$ *непосредственно выводится* из $\gamma\alpha\beta$ при помощи правила $\alpha \to \beta$

Отношение выводимости — рефлексивно-транзитивное замыкание отношения непосредственной выводимости

- $\alpha_0, \alpha_1, \ldots, \alpha_n \in V^*$
- $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n$
- $\alpha_0 \overset{*}{\Rightarrow} \alpha_n : \alpha_n$ *выводится из* α_0

Пример 1.4. [
Отношение выводимости]

$$S \to 0|N| - N$$

$$N \to 1A$$

$$A \to 0A|1A|\varepsilon$$

$$S \Rightarrow -N \Rightarrow -1A \Rightarrow -11A \Rightarrow^* -1101A \Rightarrow -1101$$

Свойства отношения выводимости

- Транзитивность
  $\forall \alpha, \beta, \gamma in V^* : \alpha \overset{*}{\Rightarrow} \beta, \beta \overset{*}{\Rightarrow} \gamma$ следовательно $\alpha \overset{*}{\Rightarrow} \gamma$
- Рефлексивность: $\forall \alpha \in V^* : \alpha \overset{*}{\Rightarrow} \alpha$
- $\alpha_0 \overset{+}{\Rightarrow} \alpha_n$: вывод использует хотя бы одно правило грамматики
- $\alpha_0 \overset{k}{\Rightarrow} \alpha_n$: вывод происходит за k шагов

Левосторонний вывод

На каждом шагу заменяем самый левый нетерминал

$$S \to AA \mid s$$

$$A \to a$$

$$A \to AA \mid Bb \mid a$$

$$B \to c \mid d$$

$$S \Rightarrow AA \Rightarrow BbA \Rightarrow cbA \Rightarrow cbAA \Rightarrow cbaA \Rightarrow cbaa$$

Правосторонний вывод определяется аналогично

Язык, порождаемый грамматикой

— всевозможные цепочки из тервиналов, которые выводятся из стартового терминала

$$L(G) = \{\omega \in V_T^* | S \overset{*}{\Rightarrow} \omega\}$$

Грамматики G_1 и G_2 эквивалентны, если $L(G_1) = L(G_2)$

Контекстно-свободная грамматика

— грамматика, все правила которой имеют вид $A \to \alpha, A \in V_n, \alpha \in V^*$. В левой части находится только один терминал

Проблема такой грамматики в том, что запись естественных языков (и некоторых языков программирования) в таком виде невозможожна. Например, значение оператора $*$ в С++ зависит от контекста. В Java же синтаксис контекстно-свободный.

Тем не менее, с помощью контестно-свободной грамматики можно записать отдельные элементы языков.

Дерево вывода

Дерево является *деревом вывода* для $G = \{V_n, V_T, P, S\}$, если

- Каждый узел помечен символов из алфавита V
- Метка корня — S
- Листья помечены терминалами, остальные узлы — нетерминалами
- Если узлы $n_0, \ldots, n_k$ — прямые потомки узла n, перечисленные слева направо, с метками

Пример 1.5. Дерево вывода
$G =< \{S, A\}, \{a, b\}, \{S \to aAS \mid a, A \to SbA \mid ba \mid sSS\}, S >$

$S \Rightarrow aAS \Rightarrow aSBaS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

Теорема. Пусть $G = \{V_N, V_T, P, S\}$ — КС-грамматика. Вывод $S^* \Rightarrow \alpha$, где $\alpha \in V^*$, $\alpha \neq \varepsilon$ существует дерево вывода в грамматика G с результатом n

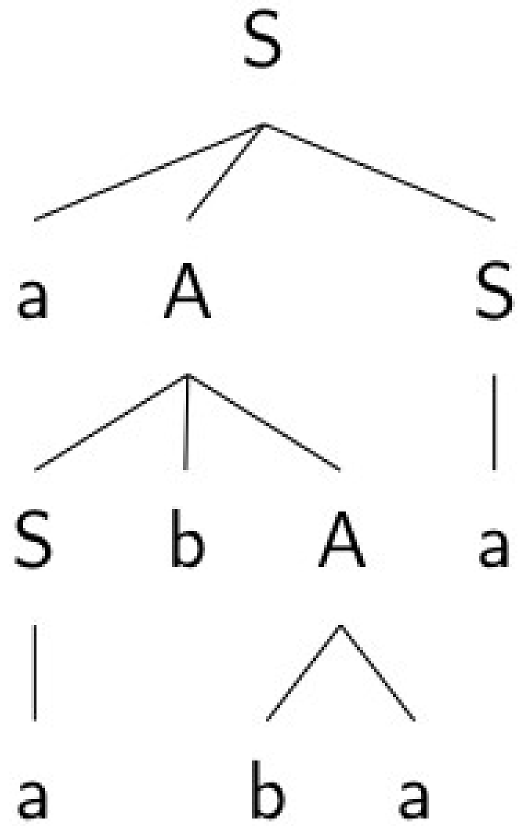

Рисунок 3. Пример дерева вывода