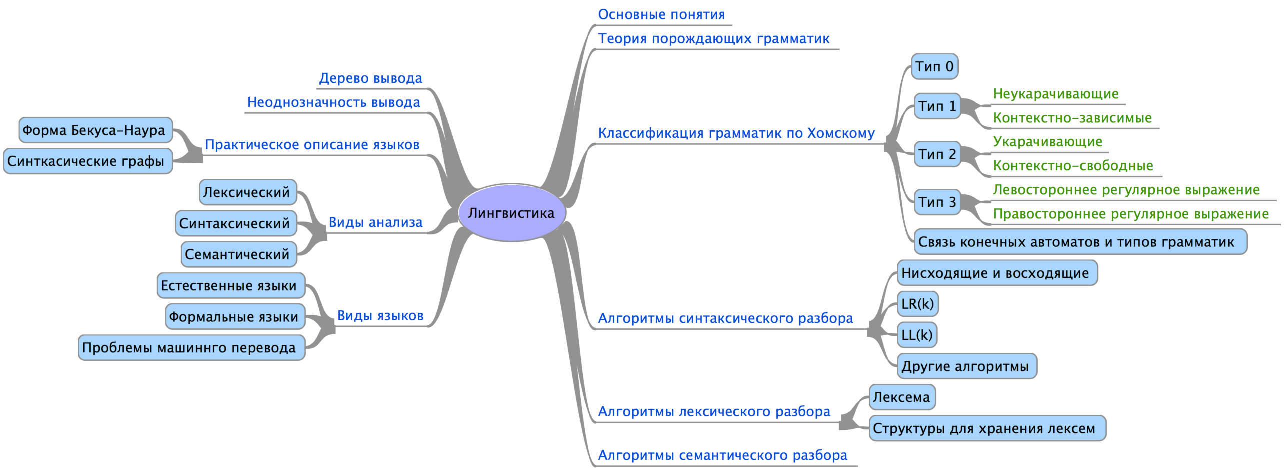
Лекции по ЛиПО



ЭЛЕМЕНТЫ ТЕОРИИ ФОРМАЛЬНЫХ ЯЗЫКОВ И ГРАММАТИК

Лекция. Основные понятия и определения

Определение: *алфавит* – это конечное множество символов.

Предполагается, что термин «символ» имеет достаточно ясный интуитивный смысл и не нуждается в дальнейшем уточнении.

Определение: *цепочкой символов в алфавите V* называется любая конечная последовательность символов этого алфавита.

Определение: цепочка, которая не содержит ни одного символа, называется *пустой цепочкой*. Для ее обозначения будем использовать символ ε.

Более формально цепочка символов в алфавите V определяется следующим образом:

(1) ε – цепочка в алфавите V;

(2) если α – цепочка в алфавите V и a – символ этого алфавита, то αa – цепочка в алфавите V;

(3) β – цепочка в алфавите V тогда и только тогда, когда она является таковой в силу (1) и (2).

Определение: если α и β – цепочки, то цепочка $\alpha\beta$ называется *конкатенацией* (или *сцеплением*) цепочек α и β.

Например, если $\alpha = ab$ и $\beta = cd$, то $\alpha\beta = abcd$.

Для любой цепочки α всегда $\alpha\varepsilon = \varepsilon\alpha = \alpha$.

Определение: *обращением* (или *реверсом*) цепочки α называется цепочка, символы которой записаны в обратном порядке.

Обращение цепочки α будем обозначать α^R.

Например, если $\alpha = abcdef$, то $\alpha^R = fedcba$.

Для пустой цепочки: $\varepsilon = \varepsilon^R$.

Определение: *n-ой степенью цепочки α* (будем обозначать α^n) называется конкатенация n цепочек α.

$\alpha^0 = \varepsilon; \alpha^n = \alpha\alpha^{n-1} = \alpha^{n-1}\alpha.$

Определение: *длина цепочки* – это число составляющих ее символов.

Например, если $\alpha = abcdefg$, то длина α равна 7.

Длину цепочки α будем обозначать $|\alpha|$. Длина ε равна 0.

Определение: *язык* в алфавите V – это подмножество цепочек конечной длины в этом алфавите.

Определение: обозначим через V^* множество, содержащее все цепочки в алфавите V, включая пустую цепочку ε.

Например, если $V = \{0,1\}$, то $V^* = \{\varepsilon,0,1,00,11,01,10,000,111,001,010,011,\ldots\}$.

Определение: обозначим через V^+ множество, содержащее все цепочки в алфавите V, исключая пустую цепочку ε.

Следовательно, $V^* = V^+ \cup \{\varepsilon\}$.

Ясно, что каждый язык в алфавите V является подмножеством множества V^*.

Определение: *декартовым произведением* $A \times B$ множеств A и B называется множество $\{(a,b) | a \subset A, b \subset B\}$.

Определение: порождающая грамматика G – это четверка (V_T, V_N, P, S), где

V_T – алфавит терминальных символов (терминалов),

V_N – алфавит нетерминальных символов (нетерминалов), не пересекающийся с V_T,

P – конечное подмножество множества $(V_T \cup V_N)^+ \times (V_T \cup V_N)^*$; элемент $(\alpha,\ \beta)$ множества P называется правилом вывода и записывается в виде $\alpha \to \beta$,

S – начальный символ (цель) грамматики, $S \subset V_N$.

Для записи правил вывода с одинаковыми левыми частями

$\alpha \to \beta_1, \alpha \to \beta_2, \ldots, \alpha \to \beta_n$

будем пользоваться сокращенной записью

$\alpha \to \beta_1 \,|\, \beta_2 \,|\, \ldots \,|\, \beta_n$.

Каждое $\beta_i, i = 1,2,\ldots,n$, будем называть альтернативой правила вывода из цепочки α.

Пример грамматики:

$G_1 = (\{0,1\}, \{A, S\}, P, S)$,

где P состоит из правил

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

Определение: цепочка $\beta \subset (V_T \cup V_N)^*$ *непосредственно выводима* из цепочки $\alpha \subset (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначим $\alpha \to \beta$), если $\alpha = \xi_1\gamma\xi_2$, $\beta = \xi_1\delta\xi_2$, где $\xi_1, \xi_2, \delta \subset (V_T \cup V_N)^*$, $\gamma \subset (V_T \cup V_N)^+$ и правило вывода $\gamma \to \delta$ содержится в P.

Например, цепочка $00A11$ непосредственно выводима из $0A1$ в грамматике G_1.

Определение: цепочка $\beta \subset (V_T \cup V_N)^*$ *выводима* из цепочки $\alpha \subset (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначим $\alpha \Rightarrow \beta$), если существуют цепочки $\gamma_0, \gamma_1, \ldots, \gamma_n, (n \geq 0)$, такие, что $\alpha = \gamma_0 \to \gamma_1 \to \ldots \to \gamma_n = \beta$.

Определение: последовательность $\gamma_0, \gamma_1, \ldots, \gamma_n$ называется выводом длины n.

Например, $S \Rightarrow 000A111$ в грамматике G_1 (см. пример выше), т.к. существует вывод $S \to 0A1 \to 00A11 \to 000A111$. Длина вывода равна 3.

Определение: *языком, порождаемым грамматикой* $G = (V_T, V_N, P, S)$, называется множество $L(G) = \{\alpha \subset V_T^* \,|\, S \Rightarrow \alpha\}$.

Другими словами, $L(G)$ – это все цепочки в алфавите V_T, которые выводимы из S с помощью P.

Например, $L(G_1) = \{0^n1^n \,|\, n > 0\}$.

Определение: цепочка $\alpha \subset (V_T \cup V_N)^*$, для которой $S \Rightarrow \alpha$, называется *сентенциальной формой* в грамматике $G = (V_T, V_N, P, S)$.

Таким образом, язык, порождаемый грамматикой, можно определить как множество терминальных сентенциальных форм.

Определение: грамматики G_1 и G_2 называются *эквивалентными*, если $L(G_1) = L(G_2)$.

Например, $G_1 = (\{0,1\}, \{A, S\}, P_1, S)$ и $G_2 = (\{0,1\}, \{S\}, P_2, S)$, где

P_1: $\qquad\qquad$ P_2:

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

$$S \to 0S1 \mid 01$$

эквивалентны, т.к. обе порождают язык $L(G_1) = L(G_2) = \{0^n1^n \mid n > 0\}$.

Определение: грамматики G_1 и G_2 *почти эквивалентны*, если $L(G_1) \cup \{\varepsilon\} = L(G_2) \cup \{\varepsilon\}$.

Другими словами, грамматики почти эквивалентны, если языки, ими порождаемые, отличаются не более, чем на ε.

Например, $G_1 = (\{0,1\}, \{A, S\}, P_1, S)$ и $G_2 = (\{0,1\}, \{S\}, P_2, S)$, где

P_1:

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P_2:

$$S \to 0S1 \mid \varepsilon$$

почти эквивалентны, т.к. $L(G_1) = \{0^n1^n \mid n > 0\}$, а $L(G_2) = \{0^n1^n \mid n \geqslant 0\}$, т.е. $L(G_2)$ состоит из всех цепочек языка $L(G_1)$ и пустой цепочки, которая в $L(G_1)$ не входит.

Лекция. Классификация грамматик и языков по Хомскому

ТИП 0:

Грамматика $G = (V_T, V_N, P, S)$ называется *грамматикой типа 0,* если на правила вывода не накладывается никаких ограничений (кроме тех, которые указаны в определении грамматики).

В общем случае, все языки типа 0 представимы недетерминированными машинами Тьюринга.

ТИП 1:

Грамматика $G = (V_T, V_N, P, S)$ называется *неукорачивающей грамматикой*, если каждое правило из Р имеет вид $\alpha \to \beta$, где $\alpha \subset (V_T \cup V_N)^+, \beta \subset (V_T \cup V_N)^+$ и $|\alpha| = |\beta|$.

Грамматика $G = (V_T, V_N, P, S)$ называется *контекстно-зависимой (КЗ)*, если каждое правило из Р имеет вид $\alpha \to \beta$, где $\alpha = \xi_1 A \xi_2$, $\beta = \xi_1 \gamma \xi_2$; $A \subset V_N$, $\gamma \subset (V_T \cup V_N)^+$; $\xi_1, \xi_2 \subset (V_T \cup V_N)^*$.

Грамматику типа 1 можно определить как неукорачивающую, либо как контекстно-зависимую.

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых неукорачивающими грамматиками, совпадает с множеством языков, порождаемых КЗ-грамматиками.

Любой язык типа 1 может быть представлен недетерминированным линейно-ограниченым автоматом.

ТИП 2:

Грамматика $G = (V_T, V_N, P, S)$ называется *контекстно-свободной (КС)*, если каждое правило из Р имеет вид $A \to \beta$, где $A \subset V_N, \beta \subset (V_T \cup V_N)^+$.

Грамматика $G = (V_T, V_N, P, S)$ называется *укорачивающей контекстно-свободной (УКС)*, если каждое правило из Р имеет вид A → β, где $A \subset V_N, \beta \subset (V_T \cup V_N)^*$.

Грамматику типа 2 можно определить как контекстно-свободную либо как укорачивающую контекстно-свободную.

Возможность выбора обусловлена тем, что для каждой УКС-грамматики существует почти эквивалентная КС-грамматика.

Любой язык типа 2 может быть представлен автоматом с магазинной памятью (МП-автоматом).

ТИП 3:

Грамматика $G = (V_T, V_N, P, S)$ называется *праволинейной*, если каждое правило из Р имеет вид $A \to tB$, либо $A \to t$, где $A \subset V_N, B \subset V_N, t \subset V_T$.

Грамматика $G = (V_T, V_N, P, S)$ называется *леволинейной*, если каждое правило из Р имеет вид $A \to Bt$, либо $A \to t$, где $A \subset V_N, B \subset V_N, t \subset V_T$.

Грамматику типа 3 (регулярную, Р-грамматику) можно определить как праволинейную либо как леволинейную.

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых праволинейными грамматиками, совпадает с множеством языков, порождаемых леволинейными грамматиками.

Для любой грамматики типа 3 существует детерминированный конечный автомат.

Соотношения между типами грамматик

(1) любая регулярная грамматика является КС-грамматикой;

(2) любая регулярная грамматика является УКС-грамматикой;

(3) любая КС-грамматика является КЗ-грамматикой;

(4) любая КС-грамматика является неукорачивающей грамматикой;

(5) любая КЗ-грамматика является грамматикой типа 0.

(6) любая неукорачивающая грамматика является грамматикой типа 0.

Замечание: УКС-грамматика, содержащая правила вида $A \to \varepsilon$, не является КЗ-грамматикой и не является неукорачивающей грамматикой.

Определение: язык L(G) является *языком типа k*, если его можно описать грамматикой типа k.

Соотношения между типами языков

(1) каждый регулярный язык является КС-языком, но существуют КС-языки, которые не являются регулярными (например, $L = \{a^n b^n \mid n > 0\}$).

(2) каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками (например, $L = \{a^n b^n c^n \mid n > 0\}$).

(3) каждый КЗ-язык является языком типа 0.

Замечание: УКС-язык, содержащий пустую цепочку, не является КЗ-языком.

Замечание: следует подчеркнуть, что если язык задан грамматикой типа k, то это не значит, что не существует грамматики типа k' (k'>k), описывающей тот же язык. Поэтому, когда говорят о языке типа k, обычно имеют в виду максимально возможный номер k.

Например, КЗ-грамматика $G_1 = (\{0,1\}, \{A, S\}, P_1, S)$ и КС-грамматика $G_2 = (\{0,1\}, \{S\}, P_2, S)$, где

P_1:

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P_2:

$S \to 0S1 \mid 01$

описывают один и тот же язык $L(G_1) = L(G_2) = \{0^n 1^n \mid n > 0\}$. Язык L называют КС-языком, т.к. существует КС-грамматика, его описывающая. Но он не является регулярным языком, т.к. не существует регулярной грамматики, описывающей этот язык.

Примеры грамматик и языков

Замечание: если при описании грамматики указаны только правила вывода P, то будем считать, что большие латинские буквы обозначают нетерминальные символы, S – цель грамматики, все остальные символы – терминальные.

1) Язык типа 0: $L = \{a^2 b^{n^2-1} | n \geq 1\}$

G(L):

$S \to aaCFD$

$F \to AFB | AB$

$AB \to bBA$

$Ab \to bA$

$AD \to D$

$Cb \to bC$

$CB \to C$

$bCD \to \varepsilon$

---

2) Язык типа 2: L={цепочки из 0 и 1 с одинаковым числом 0 и 1}

G(L):

$S \to ASB | AB$

$AB \to BA$

$A \to 0$

$B \to 1$

---

3) Язык типа 2: $L = \{(ac)^n (cb)^n | \omega \subset \{a, b\}^+$L = {(ac)n (cb)n | n > 0}

G(L):

$S \to aQb | accb$

$Q \to cSc$

---

4) Язык типа 3: $L = \{\omega \perp | \omega \subset \{a, b\}^+$, где нет двух рядом стоящих $a\}$

G(L):

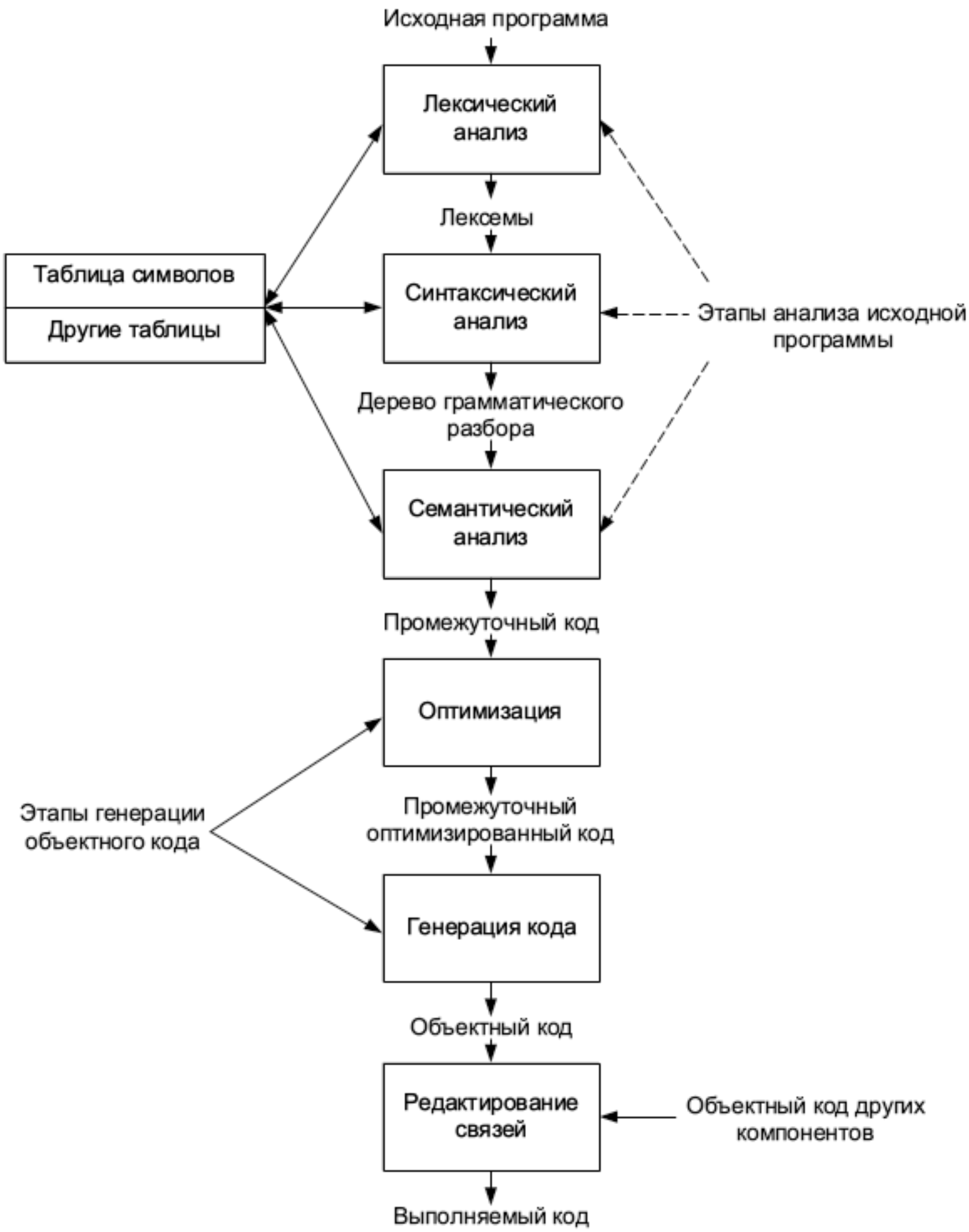
$S \to A \perp | B \perp$

$A \to a | Ba$

$B \to b | Bb | Ab$

Анализ исходной программы. Задание формальных языков.

Этапы анализа исходной программы



Лексический анализ, или сканирование.

Начальный этап любой трансляции состоит в выделении в исходной программе элементарных составляющих. Обычно лексический анализатор — это программа ввода для транслятора, последовательно читающая строки исходной программы, разбивая их на отдельные лексемы и передающая эти лексемы на дальнейшие стадии трансляции, для их анализа на более высоком уровне. Формальной моделью, используемой для создания лексического анализатора, являются *конечные автоматы.*

Синтаксический анализ, или разбор.

На этом этапе полученные лексемы используются для идентификации более крупных программных структур. Синтаксический анализ обычно чередуется с семантическим.

Семантический анализ.

На этом этапе обрабатываются структуры, которые были идентифицированы синтаксическим анализатором, и начинает формироваться структура выполняемого объектного кода. Таким образом, семантический анализ является мостом, соединяющим две части трансляции — анализ и синтез.

Семантический анализатор обычно разделен на ряд более мелких анализаторов, каждый из которых обрабатывает какую-либо определенную программную конструкцию. Некоторые наиболее общие функции можно описать следующим образом.

1. Поддержка таблицы символов.
2. Включение неявной информации.
3. Обнаружение ошибок.
4. Макрообработка и операции, выполняемые во время компиляции. Эти функции предусмотрены не во всех языках, но если они присутствуют, то обычно соответствующая обработка проводится семантическим анализатором.

Макрос в простейшей форме – это часть текста программы, которая определена отдельно и должна быть вставлена в программу во время трансляции, когда в программе встречается соответствующий вызов.

Операция, выполняемая во время компиляции – это операция, которая должна выполняться во время трансляции и осуществлять контроль над трансляцией исходной программы.

Синтез объектной программы

На заключительном этапе трансляции происходит создание выполняемой программы на основе того, что было сделано семантическим анализатором. Этот этап обязательно включает генерацию кода и может также включать оптимизацию получившейся программы.

НФБ-грамматика

Представление простого повествовательного предложения:

```
подлежащее / глагол / дополнение.

The girl/ ran/ home (Девочка побежала домой).
The boy/ cooks/ dinner (Мальчик готовит обед).

вспомогательный глагол / подлежащее / сказуемое

Did/ the girl/ run home? (Побежала ли девочка домой)?
Is/ the boy/ cooking dinner? (Готовит ли мальчик обед)?
```

```
<предложение> ::= <повествовательное> |
          <вопросительное>
<повествовательное> ::= <подлежащее> <глагол>
          <дополнение>.
<подлежащее> ::= <артикль> <существительное>.
<вопросительное> ::= <вспомогательный глагол>
          <подлежащее> <сказуемое>?
```

Где символ ::= обозначает «определено как», а символ | обозначает «или».

Такая специфическая запись называется НФБ (нормальной формой Бэкуса, или формой Бэкуса-Наура).

Синтаксис

НФБ-грамматика состоит из конечного набора правил НФБ-грамматики, которые определяют язык.

Полная НФБ-грамматика — это просто набор подробных правил, которые в совокупности определяют иерархию языков, завершающуюся синтаксической категорией самого верхнего уровня.

```
<оператор присваивания>::=<переменная> =
          <арифметическое выражение>
<арифметическое выражение>::=<терм> |
          <арифметическое выражение> + <терм> |
          <арифметическое выражение> -<терм>
<терм>::=<первичное выражение> |
          <терм> x <первичное выражение> |
          <терм> / <первичное выражение>
<первичное выражение> := <переменная> | <число> |
          (<арифметическое выражение>)
<переменная>::=<идентификатор> |
          <идентификатор> [<список индексов>]
<список индексов>::=<арифметическое выражение> |
     <список индексов>, <арифметическое выражение>
```

Деревья грамматического разбора

Имея некоторую грамматику, можно последовательно использовать правила подстановки для генерации цепочек языка.

Например, грамматика генерирует все последовательности (цепочки), состоящие из сбалансированных круглых скобок:

```
S → SS | (S) | ()
```

Цепочку (()()) можно получить из S следующим образом:

1) заменяем S по правилу S→(S) и получаем (S);
2) заменяем S в (S) по правилу S → SS и получаем (SS);
3) заменяем первое S в (SS) по правилу S → () и получаем (()S);

4) заменяем S в (()S) по правилу S → () и получаем (()()).

Если использовать символ ⇒ для указания вывода.

```
S ⇒ (S) ⇒ (SS) ⇒ (()S) ⇒ (()())
```

Каждый *член* этого вывода является *сентенциальной формой*, и формально определяется язык как множество сентенциальных форм, *которые состоят только* из терминальных символов и выводимы из исходного символа грамматики.

W = Y x (U + V)

Не могут быть заданы с помощью одной лишь НФБ-грамматики такие ограничения, как:

- ✦ один и тот же идентификатор не может быть описан дважды в одном блоке;
- ✦ каждый идентификатор должен быть описан в каком-либо блоке, определяющем область его использования;
- ✦ на массив, определенный как двухмерный, нельзя ссылаться с помощью трех индексов.

Ограничения такого рода должны быть определены как дополнение к формальной НФБ-грамматике.

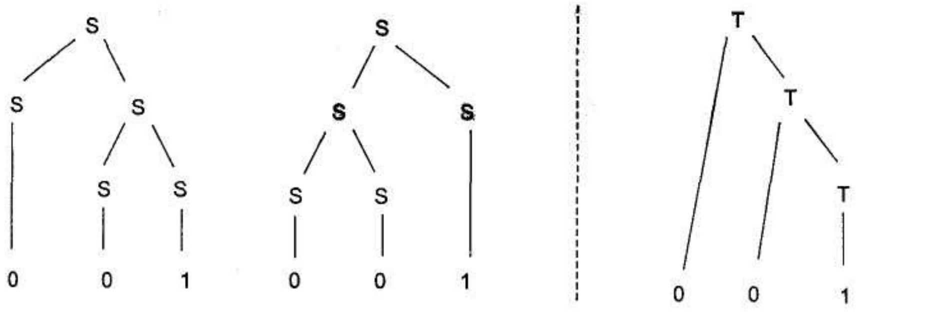
Неоднозначность

Неоднозначность — это проблема синтаксиса. Неоднозначность часто является свойством не языка, а грамматики. Например:

```
G1  :  S → SS  |  0  |  1
```

Если любая грамматика для какого-либо языка является неоднозначной, то говорят, что язык обладает *наследственной неоднозначностью.* Но в данном случае язык, состоящий из всех двоичных цепочек, не является наследственно неоднозначным, поскольку существует лишенная неоднозначности грамматика G_2:

$G_2 : T \rightarrow 0T \mid 1T \mid 0 \mid 1$



G₁: Неоднозначная грамматика G₂: Однозначная грамматика

Расширения НФБ-нотации

Для расширения НФБ-нотации применяются следующие дополнительные обозначения, которые не ограничивают возможности НФБ-грамматики, но упрощают описания языков:

- необязательный элемент может быть обозначен заключением его в квадратные скобки – [...];
- альтернативные варианты вводятся при помощи вертикальной черты | и в случае необходимости могут быть заключены в квадратные скобки ([,]);
- произвольная последовательность экземпляров одного и того же элемента может быть обозначена заключением его в фигурные скобки, за которыми следует символ «звездочка» – {...}*.

```
<оператор присваивания> ::= <переменная> =
          <арифметическое выражение>
<арифметическое выражение> ::= <терм> {[+ | -] <терм>}*
<терм> ::= <первичное выражение>
          {[x | /]<первичное выражение>}*
<первичное выражением := <переменная> | <число> |
          (<арифметическое выражение>)
<переменная> ::= <идентификатор> |
          <идентификатор> [<список индексов>]
<список индексов>:= <арифметическое выражение>
          {,<арифметическое выражение>}*
```

Синтаксические схемы.

Формы представления КС-языков. Нормальная форма Хомского

Проблемы КС языков

1. Проблема пустоты. Пусть задан КС-язык L(G), определить, является ли этот язык пустым или нет.
2. Оценка длины цепочки. Если дерево вывода имеет глубину n, то какова длина цепочки.
3. Существуют ли языки не являющиеся КС-языками?
4. Проблема конечности. Если задан язык L(G), то является ли этот язык конечным или нет?
5. Проблема принадлежности. Принадлежи ли некоторая цепочка $\boldsymbol{\alpha}$ КС-языку.
6. Проблема выводимости. Существует ли алгоритм построения дерева вывода для заданной цепочки z.

Теорема 1. Пусть G – КС-грамматика, $L(G) \neq \varnothing$, α – терминальная цепочка, длина которой ≥ 1. Тогда для αa можно поставить в соответствие дерево вывода в грамматике G.

Глубиной дерева будем называть длину самого длинного пути от корня до конечной вершины дерева. Длина дерева измеряется числом дуг, находящихся на этом пути.

Пример: G: S →AB, A→aAla, B→bBlb

$\alpha = a^3b^2$

глубина дерева = 4

Проблема пустоты

Теорема 2. Проблема непустоты разрешима для КС-грамматики, т.е. если G – является КС-грамматикой, то существует алгоритм, позволяющий определить, справедливо ли утверждение $L(G) = \varnothing$.

Доказательство:

1. Определим, имеет ли грамматика правило вывода, вида $S \to \varepsilon$. Если да, то $\varepsilon \in L(G)$ и $L(G) = \varnothing$.

2. Предположим, что в G нет правила $S \to \varepsilon$. Тогда построим все деревья вывода в грамматике, глубина которых $n \leq |N|$, где N – мощность множества нетерминалов. Число таких деревьев конечно. Если хотя бы одно из деревьев вывода является деревом вывода терминальной цепочки, то $L(G) = \varnothing$

Устранение бесполезных символов

Определение. Пусть дана порождающая грамматика $G = (V_T, V_N, P, S)$. Символ $A \subset V_N$ называется *полезным* (useful), если существуют такие слова $\alpha \subset (V_T \cup V_N)^*$, $\beta \subset (V_T \cup V_N)^*$ и $w \subset V_T^*$, что $S \Rightarrow \alpha A \beta$ и $\alpha A \beta \Rightarrow w$. Символ $A \subset V_N$ называется *бесполезным* (useless), если он не является полезным. Символ $A \subset V_N$ называется *порождающим* (generating), если существует такое слово $w \subset V_T^*$, что $A \Rightarrow w$. Символ $A \subset V_N$ называется *достижимым* (reachable), если существуют такие слова $\alpha \subset (V_T \cup V_N)^*$ и $\beta \subset (V_T \cup V_N)^*$, что $S \Rightarrow \alpha A \beta$.

Теорема 3. Пусть дана контекстно-свободная грамматика $G = (V_T, V_N, P, S)$ и $L(G) \neq \varnothing$. Тогда существуют такие множества $V_N{}' \subseteq V_N$ и $P' \subseteq P$, что в контекстно-свободной грамматике $G = (V_T, V_N{}', P', S)$ нет бесполезных символов и она эквивалентна исходной грамматике.

Доказательство. Сначала удалим все непорождающие символы (удалим также каждое правило, содержащее хотя бы один такой символ). Затем из полученной грамматики удалим все недостижимые символы (и правила, их содержащие).

Пример. Рассмотрим контекстно-свободную грамматику G с правилами

S → UX,

S → V Z,

T → aa,

T → bb,

U → aUa,

U →bUb,

V →aTb,

V →bTa,

W →YZY,

W →aab,

X →Xa,

X→Xb,

X→ε,

Y →YY,

Y →aU,

Y →ε,

Z→W,

Z→b.

Удалив четыре правила, содержащие непорождающий символ U, получим грамматику G_1. В ней символ X является недостижимым. Удалив три правила, содержащие X, получим грамматику G_2 с правилами S → VZ, T → aa, T → bb, V → aTb, V → bTa, W → YZY, W → aab, Y → YY, Y → ε, Z → W, Z → b. Очевидно, $L(G) = L(G_2)$ и грамматика G_2 не содержит бесполезных символов.

Устранение ε-правил

Теорема 4. Пусть язык *L* является контекстно-свободным. Тогда язык *L – {ε}* порождается некоторой контекстно-свободной грамматикой без *ε*-правил.

Доказательство. Пусть дана контекстно-свободная грамматика $G = (V_T, V_N, P, S)$, порождающая язык L. Проведём серию преобразований множества P.

Если для каких-то $A \subset V_N$, $B \subset V_N$, $\alpha \subset (V_N \cup V_T)^*$и $\beta \subset (V_N \cup V_T)^*$ множество P содержит правила B → αAβ и A → ε, но не содержит правила B → αβ, то добавим это правило в P. Повторяем эту процедуру, пока возможно.

Теперь исключим из множества P все правила вида A → ε. Полученная грамматика порождает язык L – {ε}.

Пример. Рассмотрим язык L, порождаемый грамматикой S → ε, S → aSbS. Язык L – {ε} порождается грамматикой S → aSbS, S → abS, S → aSb, S → ab.

Нормальная форма Хомского

Определение. *Грамматика в нормальной форме Хомского* (*грамматика в бинарной нормальной форме, квадратичная грамматика*) (grammar in Chomsky normal form) — контекстно-свободная грамматика (V_T, V_N, P, S), в которой каждое правило имеет один из следующих трёх видов: $S \to \varepsilon, A \to a, A \to BC$, где $A \subset V_N, B \subset V_N - \{S\}, C \subset V_N - \{S\}, a \subset V_T$.

Пример 7. Грамматика S → RR, S → AB, R → RR, R → AB, A → a, B → RB, B → b — грамматика в нормальной форме Хомского.

Теорема 8. Каждая контекстно-свободная грамматика эквивалентна некоторой грамматике в нормальной форме Хомского.

Доказательство. Пусть дана контекстно-свободная грамматика $G = (V_T, V_N, P, S)$. Проведём ряд преобразований этой грамматики так, что порождаемый ею язык остаётся неизменным.

Если правая часть какого-нибудь правила содержит символ S, то заменим грамматику (V_T, V_N, P, S) на грамматику $(V_T, V_N \cup S_0, P \cup \{S_0 \to S\}, S_0)$, где S_0 — новый символ, не принадлежащий множеству $V_T \cup V_N$.

Заменим во всех правилах каждый терминальный символ a на новый нетерминальный символ T_a и добавим к множеству P правила $T_a \to a$ для всех $a \subset V_T$.

Устраним правила вида $A \to \alpha$, где $|\alpha| > 2$, заменив каждое из них на ряд более коротких правил (при этом добавляются новые нетерминальные символы).

Теперь устраним все правила вида A → ε, где A не является начальным символом. Это можно сделать, как в доказательстве теоремы 4.

Если для каких-то $A \to V_N$, $B \to V_N$ и $\alpha \subset (V_N \cup V_T)^*$ множество P содержит правила $A \to B$ и $B \to \alpha$, но не содержит правила $A \to \alpha$, то добавим это правило в P. Повторяем эту процедуру, пока возможно. После этого исключим из множества P все правила вида $A \to B$.

Пример 9. Грамматика S → ε, S → aUbU, U → S, U → ba эквивалентна следующей грамматике в нормальной форме Хомского:S_0 →ε, S_0 →AD, D→UC, D→BU, D→b, C → BU, C → b, U → BA, U → AD, A → a, B → b.

Теорема 10. Если контекстно-свободный язык не содержит пустого слова, то он порождается некоторой грамматикой, в которой каждое правило имеет один из следующих двух видов: A → a, A → BC, где $A \subset V_N, B \subset V_N - \{S\}, C \subset V_N - \{S\}, a \subset V_T$.