Теоретический минимум по предмету Конструирование Компиляторов (2009 г.)

Определение грамматик типа 0 по Хомскому

Если на грамматику G = (N, T, P, S) не накладываются никакие ограничения, то её называют грамматикой типа 0, или грамматикой без ограничений.

Определение грамматик типа 1 (неукорачивающих) по Хомскому

Если

1. Каждое правило грамматики, кроме $S \to \varepsilon$, имеет вид $\alpha \to \beta$, $|\alpha| \leq |\beta|$
2. В том случае, когда $S \to \varepsilon \in P$, символ S не встречается в правых частях правил

то грамматику называют грамматикой типа 1, или неукорачивающей.

Определение детерминированной машины Тьюринга

Детерминированная машина Тьюринга — $T_m = (Q, \Gamma, \Sigma, D, q_0, F)$

1. Q — конечное множество состояний
2. Γ — конечное множество символов (конечный алфавит)
3. Σ — входной алфавит
4. D — правила перехода
5. $D: (Q \backslash F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$
6. $q_0 \in Q$ — начальное состояние
7. $F \subseteq Q$ — множество конечных состояний

Определение недетерминированной машины Тьюринга

Недетерминированная машина Тьюринга — $T_m = (Q, \Gamma, \Sigma, D, q_0, F)$

1. Q — конечное множество состояний
2. Γ — конечное множество символов (конечный алфавит)
3. Σ — входной алфавит
4. D — правила перехода
5. $D: (Q \backslash F) \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$
6. $q_0 \in Q$ — начальное состояние
7. $F \subseteq Q$ — множество конечных состояний

Определение конфигурации машины Тьюринга

Конфигурацией машины Тьюринга называется тройка (q, w, i), где

1. $q \in Q$ — состояние машины Тьюринга
2. $w \in \Gamma^*$ — вход, помещаемый на ленту машины Тьюринга, $w = a_1 \ldots a_n$
3. $i \in Z$ — положение головки машины Тьюринга

Определение языка, допускаемого машиной Тьюринга

Язык, допускаемый машиной Тьюринга — множество таких слов w, что, машина Тьюринга, находясь в состоянии $(q_0, w, 1)$ может достигнуть через конечное число переходов состояния $q \in F$.

Соотношение между языками, порождаемыми грамматиками типа 0 и языками, допускаемыми машинами Тьюринга

Класс языков, допускаемых машиной Тьюринга, эквивалентен классу языков, порождаемых грамматиками типа 0.

Объяснить разницу между недетерминированной и детерминированной машиной Тьюринга

Детерминированная машина Тьюринга из данного состояния по данному символу может сделать не более одного перехода, недетерминированная же таким свойством не обладает.

Определение регулярного множества

Регулярное множество в алфавите T определяется следующим образом:

1. {} (пустое множество) — регулярное множество в алфавите T
2. {a} — регулярное множество в алфавите T для каждого a ∈ T
3. {ε} — регулярное множество в алфавите T
4. Если P и Q — регулярные множества в алфавите T, то таковы же и множества
5. P ∪ Q (объединение)
6. PQ (конкатенация, то есть множество таких pq, что p ∈ P, q ∈ Q)
7. P* (итерация: P* = {ε} ∪ P ∪ PP ∪ PPP ∪ …)
8. Ничто другое не является регулярным множеством в алфавите T

Определение регулярного выражения

Регулярное выражение — форма записи регулярного множества.

Регулярное выражение и обозначаемое им регулярное множество определяются следующим образом:

1. ∅ — обозначает множество {}
2. ε — обозначает множество {ε}
3. a — обозначает множество {a}
4. Если РВ p и q обозначают множества P и Q соответственно, то:
5. (p|q) обозначает P ∪ Q
6. pq обозначает PQ
7. (p*) обозначает P*
8. Ничто другое не является регулярным выражением в данном алфавите

Определение праволинейной грамматики

Праволинейная грамматика или грамматика типа 3 по Хомскому — грамматика вида

1. A → w
2. A → wB
3. w ∈ T*

Определение недетерминированного конечного автомата

Недетерминированный конечный автомат - M = (Q, Σ, D, q_0, F)

1. Q — конечное непустое множество состояний
2. Σ — входной алфавит
3. D — правила перехода
4. Q × (Σ ∪ {ε}) → 2^Q
5. q_0 ∈ Q — начальное состояние
6. F ⊆ Q — множество конечных состояний

Определение детерминированного конечного автомата

Детерминированный конечный автомат - M = (Q, Σ, D, q_0, F)

1. Q — конечное непустое множество состояний
2. Σ — конечный входной алфавит
3. D — правила перехода
   a. $Q \times \Sigma \to Q$
4. $q_0 \in Q$ — начальное состояние
5. $F \subseteq Q$ — множество конечных состояний

Объяснить разницу между недетерминированным и детерминированным конечным автоматом

Детерминированный конечный автомат является частным случаем недетерминированного конечного автомата. ДКА на каждом такте работы имеет возможность перейти не более чем в одно состояние и не может делать переходы по *e.*

Определение конфигурации конечного автомата

Пусть M = (Q, T, D, q_0, F) — НКА. Конфигурацией автомата M называется пара $(q, \omega) \in Q \times T^*$, где q — текущее состояние управляющего устройства, а ω — цепочка символов на входной ленте, состоящая из символов под головкой и всех символов справа от неё.

Определение языка, допускаемого конечным автоматом

Автомат M допускает цепочку ω, если $(q_0, \omega) \vdash^* (q, \varepsilon)$ для некоторого $q \in F$. Языком, допускаемым автоматом M, называется множество входных цепочек, допускаемых автоматом M.
То есть:

- $L(M) = \{\omega \mid \omega \in T^*$ и $(q_0, \omega) \vdash^* (q, \varepsilon)$ для некоторого $q \in F\}$

Определение ε-замыкания для подмножества состояний НКА

ε-замыкание множества состояний R, $R \subseteq Q$ — множество состояний НКА, достижимых из состояний, входящих в R, посредством только переходов по ε, то есть множество

- $S = \bigcup_{q \in R} \{p \mid (q, \varepsilon) \vdash^* (p, \varepsilon)\}$

Определение расширенной функции переходов для ДКА

Расширенная функция переходов множества состояний R, $R \subseteq Q$ по a — множество состояний ДКА, в которые есть переход на входе a для состояний из R, то есть множество

- $S = \bigcup q \in R \{p \mid p \in D(q, a)\}$

Определение расширенной функции переходов для НКА

Расширенная функция переходов множества состояний R, $R \subseteq Q$ по a — множество состояний НКА, в которые есть переход на входе a для состояний из R, то есть множество

- $S = \bigcup q \in R \{p \mid p \in D(q, a)\}$

Определение функции firstpos для поддерева в дереве регулярного выражения

Функция firstpos(n) для каждого узла n узла синтаксического дерева регулярных выражений даёт множество позиций, которые соответствуют первым символам в цепочках, генерируемых подвыражением с вершиной n. Построение:

узел n	**firstpos(n)**
ε	∅
i ≠ ε	{i}
u | v	firstpos(u) ∪ firstpos(v)
u . v	if nullable(u) then firstpos(u) ∪ firstpos(v) else firstpos(u)
v*	firstpos(v)

Определение функции lastpos для поддерева в дереве регулярного выражения

Функция lastpos(n) для каждого узла n узла синтаксического дерева регулярных выражений даёт множество позиций, которым соответствуют последние символы в цепочках, генерируемых подвыражениями с вершиной n. Построение lastpos(n):

узел n	**lastpos(n)**
ε	∅
i ≠ ε	{i}
u | v	lastpos(u) ∪ lastpos(v)
u . v	if nullable(v) then lastpos(u) ∪ lastpos(v) else lastpos(v)
v*	lastpos(v)

Определение функции followpos для позиций в дереве регулярного выражения

Функция followpos(i) для позиции i есть множество позиций j таких, что существует некоторая строка …cd…, входящая в язык, описываемый регулярным выражением, такая, что позиция i соответствует вхождению c, а позиция j — вхождению d.

Сформулировать соотношение между регулярными множествами и языками, допускаемыми КА

Для всякого регулярного множества имеется конечный автомат, допускающий в точности это регулярное множество, и наоборот – язык, допускаемый конечным автоматом, есть регулярное множество.

Определение регулярной грамматики

правая регулярная грамматика – все правила могут быть в одной из следующих форм:

1. A → a
2. A → aB
3. A → ε

левая регулярная грамматика – все правила могут быть в одной из следующих форм:

4. A → a
5. A → Ba
6. A → ε

Сформулировать соотношение между языками, порождаемыми праволинейными грамматиками и языками, допускаемыми КА

Для любой праволинейной грамматики существует конечный автомат, проверяющий порождаемый грамматикой язык. Для любого конечного автомата существует праволинейная грамматика, порождающая проверяемый конечным автоматом язык.

Определение эквивалентных состояний ДКА

Состояния p и q полного детерминированного конечного автомата $M = (Q, \Sigma, D, q_0, F)$ называются эквивалентными, если не существует слова w, которое их различает.

Определение различимых состояний ДКА

Состояния p и q полного детерминированного конечного автомата $M = (Q, \Sigma, D, q_0, F)$ называются различимыми, если существует слово w, которое их различает.

Говорят, что слово w различает состояния p и q полного детерминированного конечного автомата, если одно из состояний Δw (p), Δw (q) заключительное, а другое не является заключительным.

Определение контекстно-свободной грамматики без e-правил

1. $A \to \alpha$, $\alpha \in (N \cup T)^+$
2. допускается $S \to \varepsilon$, если S не входит ни в какую правую часть

Определение контекстно-свободной грамматики

- $A \to \alpha$, $A \in N$, $\alpha \in (N \cup T)^*$

Определение вывода в КС-грамматике

Определим на множестве $(N \cup T)^*$ грамматики $G = (N, T, P, S)$ бинарное отношение выводимости «$\Rightarrow$» следующим образом: если $\delta \to \gamma \in P$, то $\alpha\delta\beta \Rightarrow \alpha\gamma\beta$ для всех $\alpha, \beta \in (N \cup T)^*$. Если $\alpha1 \Rightarrow \alpha2$, то $\alpha2$ непосредственно выводима из $\alpha1$.
Если $\alpha \Rightarrow k\ \beta$ $(k \geq 0)$, то существует последовательность шагов $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_{k-1} \Rightarrow \gamma_k$,
где $\alpha = \gamma_0$ и $\beta = \gamma_k$. Последовательность цепочек $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_{k-1}, \gamma_k$ в этом случае называется выводом β из α.

Определение языка, порождаемого КС-грамматикой

Языком, порождаемым грамматикой $G = (N, T, P, S)$ называется множество всех цепочек терминалов, выводимых из аксиомы, то есть:

- $L(G) = \{w \mid w \in T^*, S \Rightarrow^+ w\}$

Определение сентенциальной формы

Сентенциальная форма — последовательность символов (терминалов и нетерминалов), выводимых из аксиомы.

Определение однозначной КС-грамматики

КС грамматика называется однозначной или детерминированной, если всякая выводимая терминальная цепочка имеет только одно дерево вывода (соответственно только один левый и только один правый вывод).

Определение неоднозначной КС-грамматики

КС-грамматика G называется неоднозначной, если существует хотя бы одна цепочка $\alpha \subset L(G)$, для которой может быть построено два или более различных деревьев вывода.

Определение недетерминированного МП автомата

Недетерминированный автомат с магазинной памятью (МП-автомат) — семёрка $M = (Q, T, \Gamma, D, q_0, Z_0, F)$, где

1. Q — конечное множество состояний, представляющее всевозможные состояния управляющего устройства
2. T — конечный входной алфавит
3. Γ — конечный алфавит магазинных символов
4. D — отображение множества $Q \times (T \cup \{\varepsilon\}) \times \Gamma$ в множество всех конечных подмножеств $Q \times \Gamma^*$, называемое функцией переходов
5. $q_0 \in Q$ — начальное состояние управляющего устройства
6. $Z_0 \in \Gamma$ — символ, находящийся в магазине в начальный момент (начальный символ магазина)
7. $F \subseteq Q$ — множество заключительных состояний

Определение детерминированного МП автомата

Детерминированный автомат с магазинной памятью (МП-автомат) — семёрка $M = (Q, T, \Gamma, D, q_0, Z_0, F)$, где

1. Q — конечное множество состояний, представляющее всевозможные состояния управляющего устройства
2. T — конечный входной алфавит
3. Γ — конечный алфавит магазинных символов
4. D — отображение множества $Q \times (T \cup \{\varepsilon\}) \times \Gamma$ в множество всех конечных подмножеств $Q \times \Gamma^*$, называемое функцией переходов
5. $q_0 \in Q$ — начальное состояние управляющего устройства
6. $Z_0 \in \Gamma$ — символ, находящийся в магазине в начальный момент (начальный символ магазина)
7. $F \subseteq Q$ — множество заключительных состояний

Кроме того, должны выполняться следующие условия:
1. Множество $D(q, a, Z)$ содержит не более одного элемента для любых $q \in Q$, $a \in T \cup \{\varepsilon\}$, $Z_0 \in \Gamma$
2. Если $D(q, \varepsilon, Z) \neq \emptyset$, то $D(q, a, Z) = \emptyset$ для всех $a \in T$

Определение конфигурации МП автомата

Конфигурацией автомата с магазинной памятью (МП автомата) называется тройка (q, w, u), где

1. $q \in Q$ — текущее состояние магазинного устройства
2. $w \in T^*$ — непрочитанная часть входной цепочки; первый символ цепочки w находится под входной головкой; если $w = \varepsilon$, то считается, что входная лента прочитана
3. $u \in \Gamma^*$ — содержимое магазина; самый левый символ цепочки u считается вершиной магазина; если $u = \varepsilon$, то магазин считается пустым

Определение языка, допускаемого МП автоматом

Цепочка w допускается МП автоматом, если $(q_0, w, Z_0) \vdash^* (q, \varepsilon, u)$ для некоторых $q \in F$ и $u \in \Gamma^*$. Язык, допускаемый МП-автоматом M — множество всех цепочек, допускаемых автоматом M.

Что означает то, что недетерминированный МП автомат допускает опустошение магазина

Цепочка w допускается МП автоматом, если $(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$ для некоторого $q \in Q$. В таком случае говорят, что автомат допускает цепочку опустошением магазина.

Соотношение, между языками, порождаемыми КС-грамматиками, и языками, допускаемыми недетерминированными МП автоматами

Они совпадают.

Формулировка леммы о разрастании для КС-языков

Пусть L – КС язык. Тогда существует такая константа k, что если $|z| \geq k$ и $z \in L$, то z можно представить в виде $z = uvwxy$, где $vx \neq \varepsilon$, $|vwx| \leq k$ и $uv^iwx^iy \in L$ для всех $i \geq 0$.

Определение нормальной формы Хомского для КС-грамматики

Грамматика находится в нормальной форме Хомского, если правила вывода имеют вид:

1. A → BC; B, C ∈ N
2. A → a
3. S → ε (если ε ∈ L; S не входит ни в одну правую часть)

Определение правостороннего вывода в КС-грамматике

Вывод, в котором в любой сентенциальной форме на каждом шаге делается подстановка самого правого нетерминала, называется правосторонним.

Определение левостороннего вывода в КС-грамматике

Вывод, в котором в любой сентенциальной форме на каждом шаге делается подстановка самого левого нетерминала, называется левосторонним.

Какая грамматика называется леворекурсивной?

Грамматика называется леворекурсивной, если в ней имеется нетерминал A такой, что существует вывод A ⇒ Au для некоторой строки u.

Определение множества $FIRST_1$

$FIRST_1$ — множество всех терминальных символов, с которых может начинаться цепочка терминальных символов, выводимых из цели грамматики или ε, если u ⇒* ε.

Определение множества FOLLOW1 (?)

Множество FOLLOW(A) — множество терминальных символов, которые следуют за цепочками, выводимыми из A в грамматике G = (VT, VN, P, S), то есть, FOLLOW(A) = {a ∈ VT | S ⇒ αAβ, β → aγ, A ∈ VN, α, β, γ ∈ (VT ∪ VN)*}

Определение LL(1) Грамматики

Контекстно-свободная грамматика называется LL(1) грамматикой тогда и только тогда, когда выполняются следующие два условия:

1. Для каждого нетерминала, являющегося левой частью нескольких правил:
   < A > → α1 | α2 | … | αn необходимо, чтобы пересечение множеств FIRST(αi) и FIRST(αj) было пусто для всех i ≠ j

2. Для каждого аннулирующего нетерминала < A > ⇒ *$ необходимо, чтобы пересечение FIRST(A) и FOLLOW(A) было пустым

Грамматика, для которой таблицы анализа не имеют неоднозначно определённых входов, называются LL(1).

Определение LR(1) ситуации

LR(1)-ситуацией называется пара [A → α . β, a], где A → α β — правило грамматики, a — терминал или правый концевой маркер $. Вторая компонента ситуации называется аванцепочкой.

Определение LR(1) грамматики

Грамматика называется LR(1), если из условий

1. $S' \Rightarrow_r^* uAw \Rightarrow_r uvw$
2. $S' \Rightarrow_r^* zBx \Rightarrow_r uvy$
3. FIRST(w) = FIRST(y)

следует, что uAy = zBx (т.е. u = z, A = B и x = y).

Какого типа конфликты могут появиться в канонической системе множеств LR(1) ситуаций?

1. Сдвиг/свертка
2. Свертка/свертка

Определение конфигурации LR-анализатора

Конфигурация LR анализатора — пара, первая компонента которой —содержимое стека, вторая — непросмотренный вход:

$(S_0\ X_1\ S_1\ X_2\ S_2 \ldots X_m\ S_m,\ a_i\ a_{i+1} \ldots a_n\ \$)$.

Эта конфигурация соответствует правой сентенциальной форме $X_1\ X_2 \ldots X_m\ A_i\ A_{i+1} \ldots A_n$.

Как меняется конфигурация LR-анализатора при действии reduce?

Пусть LR – находится в конфигурации $(S_0\ X_1\ S_1\ X_2\ S_2 \ldots X_m\ S_m,\ a_i\ a_{i+1} \ldots a_n\ \$)$.

Если Action[S_m, a_i] =*reduce* A → γ, то анализатор выполняет свертку, переходя в конфигурацию $(S_0\ X_1\ S_1\ X_2\ S_2 \ldots X_{m-r}\ S_{m-r}AS,\ a_i\ a_{i+1} \ldots a_n\ \$)$. Где S = *Goto*[$S_{m-r}$,A] и r – длина γ, правой части правила вывода.

Какие типы действий выполняет LR-анализатор?

1. shift
2. reduce
3. accept
4. error

Как меняется конфигурация LR-анализатора при действии shift?

Пусть LR – находится в конфигурации $(S_0\ X_1\ S_1\ X_2\ S_2 \ldots X_m\ S_m,\ a_i\ a_{i+1} \ldots a_n\ \$)$.

Если Action[S_m, a_i] =*shift S,* то анализатор выполняет сдвиг, переходя в конфигурацию
$(S_0\ X_1\ S_1\ X_2\ S_2 \ldots X_m\ S_m a_i S,\ a_{i+1} \ldots a_n\ \$)$. Таким образом, в магазин помещается входной символ ai и символ состояния S, определяемый Action[S_m, a_i]. Текущим входным символом становится a_{i+1}.

Что такое основа правой сентенциальной формы

Подцепочка сентенциальной формы, которая может быть сопоставлена правой части некоторого правила вывода, свертка по которому к левой части правила соответствует одному шагу в обращении правостороннего вывода, называется основой цепочки.