Лекции по теории формальных языков

Лекция 7.
Преобразования контекстно-свободных грамматик (окончание).
Лексический анализ.
Разделённые грамматики

Александр Сергеевич Герасимов
http://gas-teach.narod.ru

Кафедра математических и информационных технологий
Санкт-Петербургского академического университета
Российской академии наук.
Весенний семестр 2010/11 учебного года

25 марта 2011 г.

План

План

Левая рекурсия: определения

- Нетерминал B называется *леворекурсивным*, если $B \Rightarrow^+ B\gamma$, и *непосредственно леворекурсивным*, если $B \to B\gamma$ — правило вывода (такое правило вывода называется *леворекурсивным*).
- КС-грамматика называется *леворекурсивной*, если леворекурсивен хотя бы один её нетерминал. В противном случае КС-грамматика называется *нелеворекурсивной* (или грамматикой *без левой рекурсии*).
- *A-правилом* КС-грамматики называется правило вида $A \to \alpha$.

Лемма об устранении непосредственной левой рекурсии

Лемма

Пусть A — непосредственно леворекурсивный нетерминал грамматики G,

$$A \to A\alpha_1 | \ldots | A\alpha_k | \beta_1 | \ldots | \beta_m, \tag{1}$$

— все A-правила грамматики G, причём ни одна из цепочек β_i не начинается с A. Тогда грамматика G', которая получается из G добавлением нового нетерминала A' и заменой правил (1) на правила

$$A \to \beta_1 A' | \ldots | \beta_m A', \quad A' \to \alpha_1 A' | \ldots | \alpha_k A' | \varepsilon, \tag{2}$$

эквивалентна грамматике G.

Множество правил, полученное заменой правил (1) на правила (2) в множестве правил P, обозначим через $eilr(P, A)$.

Лемма об устранении непосредственной левой рекурсии: доказательство

Доказательство.

- Применением правил (1) грамматики G из нетерминала A можно вывести в точности все цепочки вида $\beta_i\alpha_{j_1}\ldots\alpha_{j_l}$, где $i \in \{1,\ldots,m\}$, $l \geqslant 0$, $j_1,\ldots,j_l \in \{1,\ldots,k\}$.
- Применением правил (2) грамматики G' из нетерминала A можно вывести в точности такие же цепочки.
- Остальные правила грамматик G и G' и их аксиомы совпадают.

□

Пример устранения леворекурсивных правил из КС-грамматики

- Грамматика GA_2 (порождающая язык арифметических выражений LA) с множеством правил P:
  - $\underbrace{E}_{A} \to \underbrace{T}_{\beta_1} \mid \underbrace{E}_{A}\underbrace{+T}_{\alpha_1}$,
  - $T \to F \mid T * F$,
  - $F \to (E) \mid \mathtt{x}$.
- $P' = eilr(P, E)$:
  - $\underbrace{E}_{A} \to \underbrace{T}_{\beta_1}\underbrace{E'}_{A'}$, $\underbrace{E'}_{A'} \to \underbrace{+T}_{\alpha_1}\underbrace{E'}_{A'} \mid \varepsilon$,
  - $T \to F \mid T * F$,
  - $F \to (E) \mid \mathtt{x}$.
- Грамматика $GA_3 = eilr(P', T)$:
  - $E \to TE'$, $E' \to +TE' \mid \varepsilon$,
  - $T \to FT'$, $T' \to *FT' \mid \varepsilon$,
  - $F \to (E) \mid \mathtt{x}$.

Устранение правила из КС-грамматики

Лемма

Пусть КС-грамматика G содержит правило $A \to B\alpha$; $B \to \beta_1 | \ldots | \beta_k$ — все B-правила этой грамматики. Тогда грамматика G', которая получается из G удалением правила $A \to B\alpha$ и добавлением правил $A \to \beta_1\alpha | \ldots | \beta_k\alpha$, эквивалентна G.

Множество правил, полученное описанным в этой лемме преобразованием из множества правил P грамматики G, обозначим через $er(P,\ A \to B\alpha)$.

Алгоритм устранения левой рекурсии

Вход. Неукорачивающая КС-грамматика $G = (\Sigma, \Gamma, P, A_1)$ без правил вида $A \to A$, где $\Gamma = \{A_1, \ldots, A_n\}$.
Выход. Нелеворекурсивная КС-грамматика G', эквивалентная G.

(Для каждого правила вида $A_i \to A_j\alpha$ грамматики G' будет $i < j$.)

1. $P' := P$; $\Gamma' := \Gamma$; $i := 1$;
2. **пока** $i \leqslant n$ **повторять**
3. $\quad j := 1$;
4. $\quad$ **пока** $(j < i)$ **повторять**
5. $\quad\quad$ **для каждого** $(A_i \to A_j\alpha) \in P'$ **выполнить**
6. $\quad\quad\quad P' := er(P',\ A_i \to A_j\alpha)$;
7. $\quad\quad j := j + 1$;
8. $\quad$ **если** $(\exists\beta\ ((A_i \to A_i\beta) \in P'))$
9. $\quad\quad \Gamma' := \Gamma' \cup \{A'_i\}$;
10. $\quad\quad P' := eilr(P', A_i)$;
11. $\quad i := i + 1$;
12. $G' := (\Sigma, \Gamma', P', A_1)$

Пример устранения левой рекурсии: начало

- Исходная грамматика:
  - $A_1 \to A_2 A_3 | \mathrm{a}$,
  - $A_2 \to A_3 A_1 | A_1 \mathrm{b}$,
  - $A_3 \to A_1 A_2 | A_3 A_3 | \mathrm{a}$.
- $i = 1$: без изменений.
- $i = 2, j = 1$:
  - $A_1 \to A_2 A_3 | \mathrm{a}$,
  - $A_2 \to A_3 A_1 | \sout{A_1 \mathrm{b}} | \underline{A_2 A_3 \mathrm{b} | \mathrm{ab}}$,
  - $A_3 \to A_1 A_2 | A_3 A_3 | \mathrm{a}$.
- $i = 2$:
  - $A_1 \to A_2 A_3 | \mathrm{a}$,
  - $A_2 \to \sout{A_3 A_1 | A_2 A_3 \mathrm{b} | \mathrm{ab}}$ $\underline{A_3 A_1 A_2' | \mathrm{ab} A_2'}$, $\underline{A_2' \to A_3 \mathrm{b} A_2' | \varepsilon}$,
  - $A_3 \to A_1 A_2 | A_3 A_3 | \mathrm{a}$.

Пример устранения левой рекурсии: окончание

- $i = 3, j = 1$:
  - $A_1 \to A_2A_3|\mathrm{a}$,
  - $A_2 \to A_3A_1A_2'|\mathrm{ab}A_2', \quad A_2' \to A_3\mathrm{b}A_2'|\varepsilon$,
  - $A_3 \to \cancel{A_1A_2}|A_3A_3|\mathrm{a}|\underline{A_2A_3A_2|\mathrm{a}A_2}$.
- $i = 3, j = 2$:
  - $A_1 \to A_2A_3|\mathrm{a}$,
  - $A_2 \to A_3A_1A_2'|\mathrm{ab}A_2', \quad A_2' \to A_3\mathrm{b}A_2'|\varepsilon$,
  - $A_3 \to A_3A_3|\mathrm{a}|\cancel{A_2A_3A_2}|\mathrm{a}A_2|\underline{A_3A_1A_2'A_3A_2|\mathrm{ab}A_2'A_3A_2}$.
- $i = 3$:
  - $A_1 \to A_2A_3|\mathrm{a}$,
  - $A_2 \to A_3A_1A_2'|\mathrm{ab}A_2', \quad A_2' \to A_3\mathrm{b}A_2'|\varepsilon$,
  - $A_3 \to \cancel{A_3A_3|\mathrm{a}|\mathrm{a}A_2|A_3A_1A_2'A_3A_2|\mathrm{ab}A_2'A_3A_2}\ \underline{\mathrm{a}A_3'|\mathrm{a}A_2A_3'|\mathrm{ab}A_2'A_3A_2A_3'}$,
    $\underline{A_3' \to A_3A_3'|A_1A_2'A_3A_2A_3'|\varepsilon}$.

План

Теорема о нелеворекурсивной КС-грамматике

Теорема

Для любой неукорачивающей КС-грамматики G без правил вида $A \to A$ алгоритм на слайде 9 строит эквивалентную ей нелеворекурсивную КС-грамматику G'.

Доказательство.

- По леммам на слайдах 5 и 8 грамматики G и G' эквивалентны.
- Никакой нетерминал A'_i грамматики G' не является леворекурсивным, поскольку A'_i не может быть первым символом правой части никакого правила (см. преобразования *eilr* и *er* на слайдах 5 и 8).
- Индукцией по i докажем, что после i-го исполнения тела цикла по i (рассматриваемого алгоритма) справедливо утверждение: для каждого $k \leqslant i$ $A_k \to A_j\alpha \in P'$ влечёт $k < j$.
- База индукции ($i = 1$) верна, поскольку все правила вида $A_1 \to A_1\beta$ устранены в строках 8–10 алгоритма.

Теорема о нелеворекурсивной КС-грамматике: окончание доказательства

- Индукционный переход. Рассмотрим i-е исполнение тела цикла по i, предположив, что устанавливаемое утверждение верно для всех предыдущих исполнений тела этого цикла.
  - При j-м ($j < i$) исполнении тела цикла по j преобразование *er* (см. слайд 8) заменяет каждое правило вида $A_i \to A_j\alpha$, на правила вида $A_i \to \beta\alpha$, причём если $\beta = A_m\beta'$, то $A_j \to A_m\beta' \in P'$ и по индукционному предположению $j < m$. Следовательно, после исполнения цикла по j устранены все правила вида $A_i \to A_j\alpha$, где $j < i$.
  - Затем при исполнении строк 8–10 устраняются все правила вида $A_i \to A_i\alpha$, при этом не появляются правила вида $A_k \to A_l\alpha$, где $l \leqslant k$ (см. преобразование *eilr* на слайде 5).
  - Таким образом, после рассматриваемого исполнения тела цикла для каждого $k \leqslant i$ $A_k \to A_j\alpha \in P'$ влечёт $k < j$.
- Теперь видно, что никакой из нетерминалов $A_1, \ldots, A_n$ грамматики G' не является леворекурсивным, так как если существует левосторонний вывод $A_i \Rightarrow^+_{G'} A_m\alpha$, то $i < m$.

План

Левофакторизованные КС-грамматики: мотивировка и определение

- Рассмотрим два правила из грамматики языка программирования Паскаль:
  $\langle \texttt{оператор} \rangle \to \texttt{if}\ \langle \texttt{условие} \rangle\ \texttt{then}\ \langle \texttt{оператор} \rangle$
  $\langle \texttt{оператор} \rangle \to \texttt{if}\ \langle \texttt{условие} \rangle\ \texttt{then}\ \langle \texttt{оператор} \rangle\ \texttt{else}\ \langle \texttt{оператор} \rangle$
  Когда синтаксический анализатор доходит до `if`, ему непонятно, какое из этих правил применить.
- Заменим эти правила на
  $\langle \texttt{оператор} \rangle \to \texttt{if}\ \langle \texttt{условие} \rangle\ \texttt{then}\ \langle \texttt{оператор} \rangle\ \langle \texttt{иначе} \rangle$
  $\langle \texttt{иначе} \rangle \to \texttt{else}\ \langle \texttt{оператор} \rangle \mid \varepsilon$
- Пусть $A \to \alpha_1 | \ldots | \alpha_n$ — все A-правила КС-грамматики. Тогда каждая цепочка α_i называется *альтернативой нетерминала* A.
- КС-грамматика называется *левофакторизованной*, если никакой её нетерминал не имеет альтернатив с непустым общим префиксом.
- Под *левой факторизацией* КС-грамматики понимается преобразование этой КС-грамматики к эквивалентной левофакторизованной.

Левая факторизация некоторых A-правил

Лемма

Пусть $G = (\Sigma, \Gamma, P, S)$ *— КС-грамматика,* $A \in \Gamma$, $\alpha \in (\Sigma \cup \Gamma)^+$ *— максимальная по длине цепочка, с которой начинаются хотя бы две альтернативы нетерминала* A,

$$A \to \alpha\beta_1 | \ldots | \alpha\beta_n \quad (n \geqslant 2) \tag{1}$$

— все A*-правила грамматики* G*, правые части которых начинаются с* α*. Тогда грамматика* G'*, которая получается из* G *добавлением нового нетерминала* A' *и заменой правил* (1) *на правила*

$$A \to \alpha A', \quad A' \to \beta_1 | \ldots | \beta_n, \tag{2}$$

эквивалентна грамматике G*, и нетерминал* A' *не имеет альтернатив с непустым общим префиксом.*

Множество правил, полученное заменой правил (1) на правила (2) в множестве правил P, обозначим через $lf(P, A, \alpha, A')$.

Алгоритм левой факторизации КС-грамматики

Вход. КС-грамматика $G = (\Sigma, \Gamma, P, S)$.
Выход. Левофакторизованная КС-грамматика G', эквивалентная G.

1. $P' := P$; $\Gamma' := \Gamma$;
2. **для каждого** $A \in \Gamma$
3. $\quad M := \{\alpha \in (\Gamma' \cup \Sigma)^+ \mid \exists \beta_1, \beta_2 \in (\Gamma' \cup \Sigma)^* : \beta_1 \neq \beta_2 \wedge (A \to \alpha\beta_1 | \alpha\beta_2) \in P'\}$;
4. $\quad$ **если** $(M \neq \emptyset)$
5. $\quad\quad$ **выбрать из** M **цепочку** α' **максимальной длины**;
6. $\quad\quad$ **выбрать новый нетерминал** $A' \notin \Gamma'$; $\Gamma' := \Gamma' \cup \{A'\}$;
7. $\quad\quad$ $P' := lf(P', A, \alpha', A')$;
8. $\quad\quad$ **перейти на** 3;
9. $G' := (\Sigma, \Gamma', P', S)$

Пример левой факторизации КС-грамматики

- Исходная грамматика:
  - $A \to \mathtt{ab}C|\mathtt{abd}|\mathtt{a}B|\mathtt{a}C|\mathtt{b}$.
- $M = \{\mathtt{ab}, \mathtt{a}\}$, $\alpha' = \mathtt{ab}$:
  - $A \to \sout{\mathtt{ab}C|\mathtt{abd}}|\mathtt{a}B|\mathtt{a}C|\mathtt{b}|\underline{\mathtt{ab}A'}$,
  - $\underline{A' \to C|\mathtt{d}}$.
- $M = \{\mathtt{a}\}$, $\alpha' = \mathtt{a}$:
  - $A \to \sout{\mathtt{a}B|\mathtt{a}C|\mathtt{b}|\mathtt{ab}A'}|\underline{\mathtt{a}A''}$,
  - $A' \to C|\mathtt{d}$,
  - $\underline{A'' \to B|C|\mathtt{b}A'}$.

Теорема о левофакторизованной КС-грамматике

Теорема

Для любой КС-грамматики $G = (\Sigma, \Gamma, P, S)$ алгоритм на слайде 18 строит эквивалентную ей левофакторизованную КС-грамматику $G' = (\Sigma, \Gamma', P', S)$.

Доказательство.

- По лемме на слайде 17 грамматики G и G' эквивалентны, и каждый нетерминал из $\Gamma' \setminus \Gamma$ не имеет в G' альтернатив с непустым общим префиксом.
- Для каждого нетерминала $A \in \Gamma$ грамматики G' в результате преобразования *If* множество M уменьшается, поэтому за конечное число шагов M станет пустым и у нетерминала A не останется альтернатив с непустым общим префиксом.

□

ВТОРАЯ ЧАСТЬ КУРСА.

СИНТАКСИЧЕСКИЙ АНАЛИЗ

Задача синтаксического анализа

По данной КС-грамматике G и цепочке терминалов w определить, $w \in L(G)$ или нет; а также

- если $w \in L(G)$, то выдать информацию о выводе этой цепочки, например, дерево вывода;
- если $w \notin L(G)$, то желательно указать предполагаемую ошибку (или ошибки), приводящую к факту $w \notin L(G)$.

Упрощённая схема компилятора

План

Задача лексического анализа. Лексемы и токены

Лексический анализатор

- читает входную цепочку (программу),
- выделяет из неё подцепочки, имеющие самостоятельное значение и называемые *лексемами*, и
- каждую лексему относит к некоторому классу лексем — *токену*.

На вход синтаксического анализатора подаётся цепочка в алфавите токенов.

Примеры лексем

```
if a13>2300 then bc2:=84
```

Лексемы	Токены
`if`	ключевое слово `if`
`a13`	идентификатор `id`
`>`	операция отношения `rel`
2300	число `num`
`then`	ключевое слово `then`
`bc2`	идентификатор `id`
`:=`	операция присваивания `assign`
84	число `num`

Шаблоны токенов

Токены описываются *шаблонами*, в качестве которых обычно выступают (расширенные) регулярные выражения.

- Для записи альтернатив, являющихся символами общеизвестного алфавита, используют дефис, например, вместо $\mathtt{a}|\mathtt{b}|\ldots|\mathtt{z}|0|1|\ldots|9$ пишут $\mathtt{a}-\mathtt{z}|0-9$.
- Вместо rr^* пишут r^+.
- Вместо $r|\varepsilon$ пишут $r?$.

Шаблон токена `id`:

$$\langle \texttt{имя} \rangle = \mathtt{a}-\mathtt{z}(\mathtt{a}-\mathtt{z}|0-9)^*$$

Шаблон токена `num`:

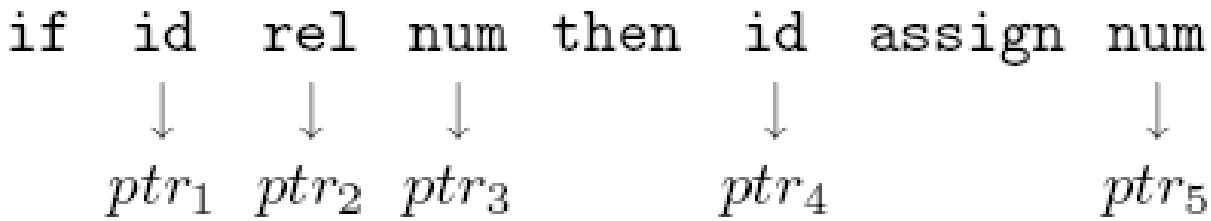
$$\langle \texttt{число} \rangle = (+|-)?(0-9)^+(.(0-9)^+)?(\mathtt{E}(+|-)?(0-9)^+)?$$

Шаблоны токенов с примерами лексем

Токен	Примеры лексем	Шаблон
`if`	`if`	`if`
`id`	`a, bc2, sum`	⟨имя⟩
`rel`	`=, <, >=`	=|<>|<|<=|>|>=
`aop`	+, −	+|−
`mop`	∗, /	∗|/
`num`	84, +9.72, −43.6E50	⟨число⟩
`then`	`then`	`then`
`assign`	`:=`	`:=`
`comma`	,	,

Атрибуты токена. Регистрация токена

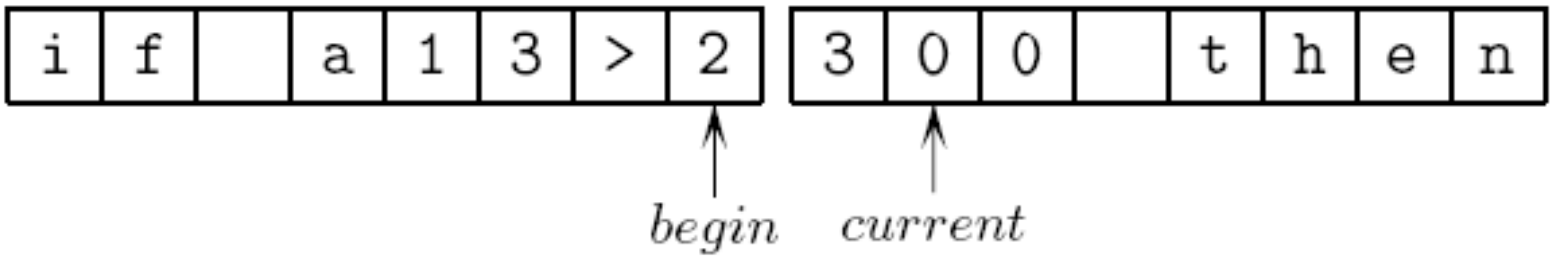
- Лексемы одного класса (например, идентификаторы) можно не различать на фазе синтаксического анализа, но их потребуется различать на следующих фазах компиляции.
- Если токен — это класс, состоящий из более чем одной лексемы, то информация о конкретной лексеме сохраняется в виде *атрибутов* токена.
- Обычно у токена нет атрибутов или один атрибут — указатель на запись в таблице символов.



- Процесс занесения информации о лексеме в таблицу символов с последующим выводом пары (токен, атрибут) на выход лексического анализатора называют *регистрацией токена*.

Организация ввода

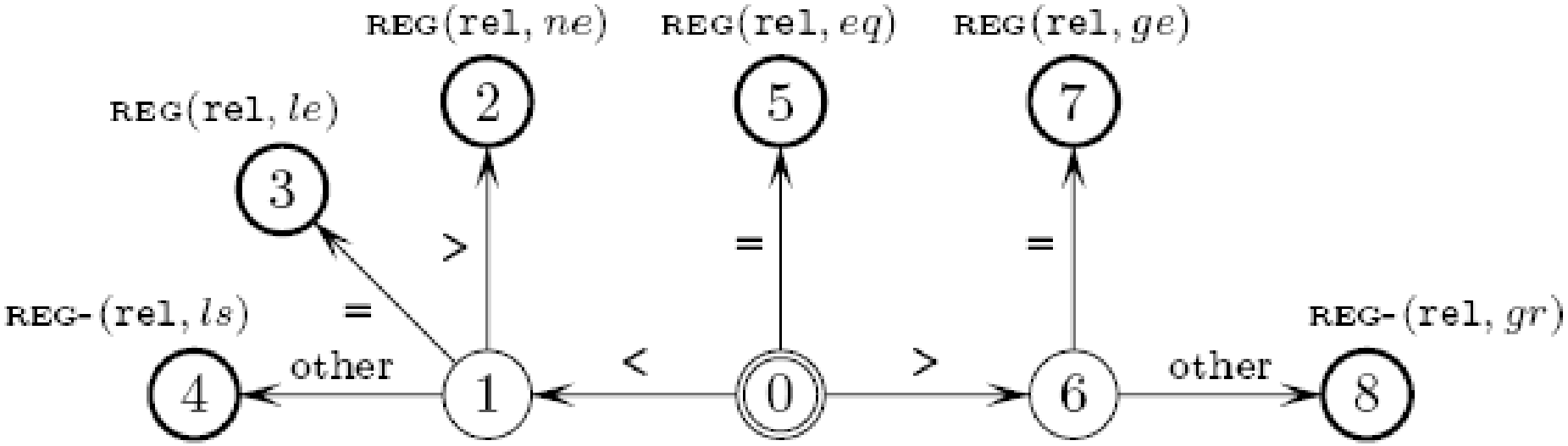
- Естественно требовать однопроходности от лексического анализатора.
- Вместо посимвольного чтения программы эффективнее зачитывать блок программы в буфер, причём удобно использовать два буфера.
- Один указатель адресует первый символ текущей лексемы, а другой — обрабатываемый символ. Распознанная лексема будет находиться между этими указателями.



Распознавание лексем

- Неоднозначности при распознавании лексем обычно разрешаются с помощью *принципа наидлиннейшей лексемы*: среди всех лексем, начинающихся с данной позиции, выбирается самая длинная.
- Можно использовать конечный автомат, распознающий шаблоны всех токенов. Каждому заключительному состоянию такого автомата однозначно соответствует токен, и с этим состоянием связана функция регистрации токена.
- Можно построить конечные автоматы для каждого шаблона в отдельности, затем соединить их в один ε-НКА и найти для этого ε-НКА эквивалентный ДКА с минимальным числом состояний (см. лекции 1 и 2).

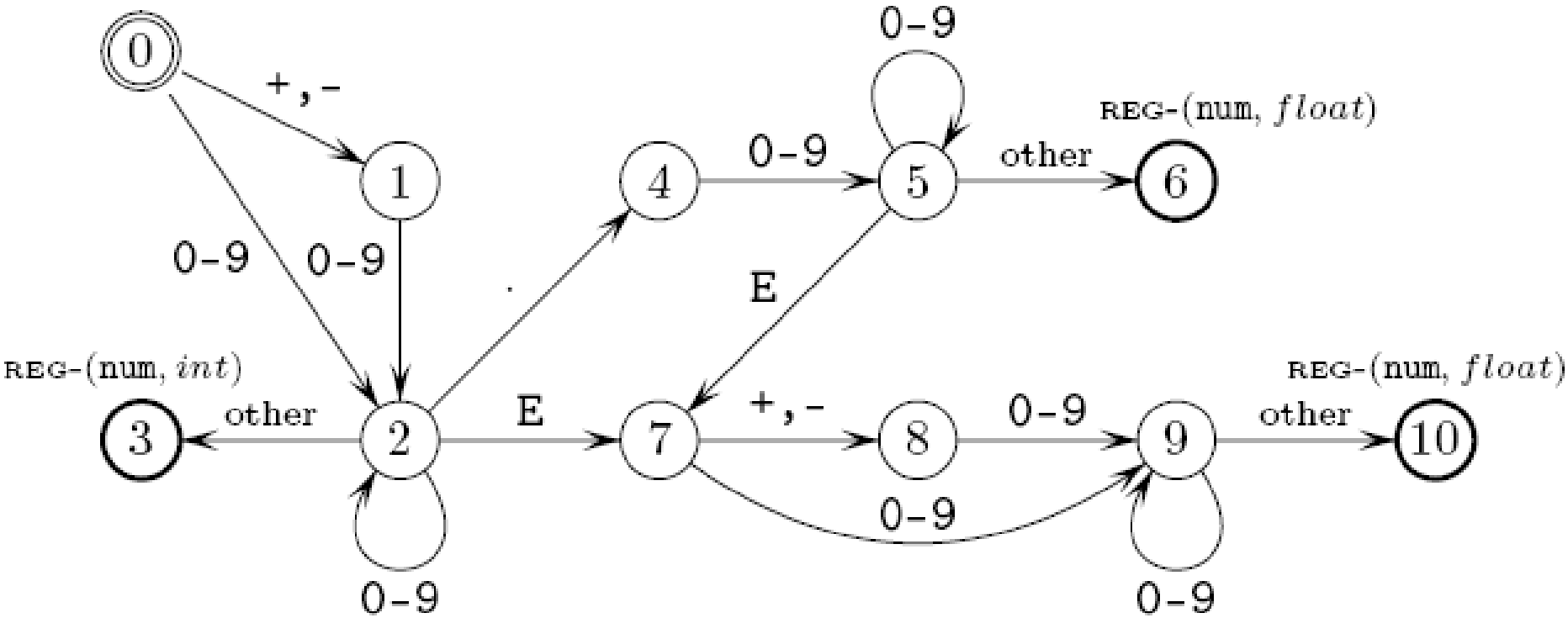
Автомат для шаблона rel



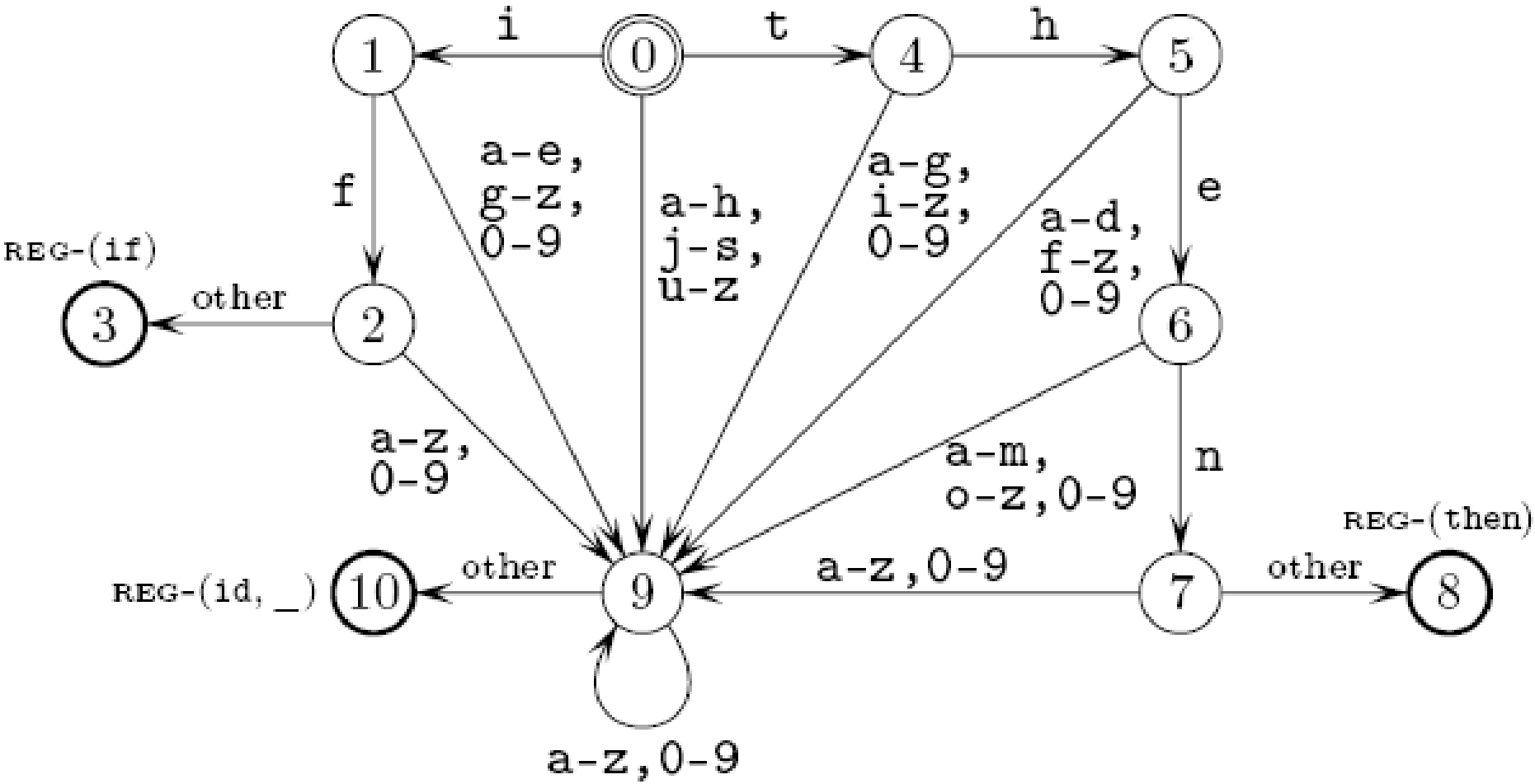
Функция регистрации токена REG- применяется, если последний прочитанный символ не принадлежит распознанной лексеме.

Функция регистрации токена REG применяется, если последний прочитанный символ принадлежит распознанной лексеме.

Автомат для шаблона num



Автомат для шаблонов if, then, id



Не изображены дуги из вершин 1, 4, 5, 6 в вершину 10 с меткой other.

План

Нисходящий анализ. Пример 1

- Синтаксический анализ называется *нисходящим анализом* (*анализом сверху вниз*), если дерево вывода анализируемой цепочки строится сверху (от корня) вниз.
- Далее все рассматриваемые выводы являются левосторонними, если иное не будет оговорено.
- Рассмотрим грамматику G_1:
  - $S \to \mathtt{a}DS \mid \mathtt{b}D\mathtt{c}$,
  - $D \to \mathtt{a}D \mid \mathtt{cc}$.
- Проверим, $\mathtt{aaccbccc} \in L(G_1)$ или нет. Предположим, что $S \Rightarrow^* \mathtt{aaccbccc}$. Тогда

$$S \Rightarrow \mathtt{a}\underbrace{DS}_{\mathtt{accbccc}} \Rightarrow \mathtt{aa}\underbrace{DS}_{\mathtt{ccbccc}} \Rightarrow \mathtt{aacc}\underbrace{S}_{\mathtt{bccc}} \Rightarrow \mathtt{aaccb}\underbrace{D\mathtt{c}}_{\mathtt{ccc}} \Rightarrow \mathtt{aaccbccc}.$$

Итак, $\mathtt{aaccbccc} \in L(G_1)$.

Нисходящий анализ. Пример 2. Определение разделённой грамматики

- Рассматриваем грамматику G_1:
  - $S \to \mathtt{a}DS \mid \mathtt{b}D\mathtt{c}$,
  - $D \to \mathtt{a}D \mid \mathtt{cc}$.
- Проверим, $\mathtt{baabcc} \in L(G_1)$ или нет. Предположим, что $S \Rightarrow^* \mathtt{baabcc}$. Тогда

$$S \Rightarrow \mathtt{b}\underbrace{D\mathtt{c}}_{\mathtt{aabcc}} \Rightarrow \mathtt{ba}\underbrace{D\mathtt{c}}_{\mathtt{abcc}} \Rightarrow \mathtt{baa}\underbrace{D\mathtt{c}}_{\mathtt{bcc}} \Rightarrow ?!$$

  Таким образом, $\mathtt{baabcc} \notin L(G_1)$.
- КС-грамматика называется *разделённой*, если
  - правая часть каждого правила вывода начинается с терминала, и
  - для каждого нетерминала все его альтернативы начинаются с разных терминалов.

Алгоритм построения распознающего МПА для разделённой грамматики

По умолчанию под МПА будем понимать ДМПА, который имеет единственное состояние и команду допуска. Такой МПА задаётся четвёркой $(\widehat{\Sigma}, \widehat{\Gamma}, \delta, \gamma_0)$, состоящей из входного алфавита $\widehat{\Sigma}$, стекового алфавита $\widehat{\Gamma}$, множества команд δ и начального содержимого стека γ_0.

Вход. Разделённая грамматика $G = (\Sigma, \Gamma, P, S)$.
Выход. МПА $\mathcal{M} = (\widehat{\Sigma}, \widehat{\Gamma}, \delta, \gamma_0)$ такой, что $L(G) = L(\mathcal{M})$.

1. $\widehat{\Sigma} := \Sigma \cup \{\dashv\}$; $\widehat{\Gamma} := \Sigma \cup \Gamma \cup \{\nabla\}$; $\gamma_0 := S$;
2. $\delta := \{(\dashv, \nabla) \to \checkmark\}$;
3. **для каждого** $(B \to \mathrm{a}\gamma) \in P$
4. $\qquad \delta := \delta \cup \{(\mathrm{a}, B) \to (\mathrm{a}\gamma, \_)\}$;
5. **для каждого** $\mathrm{a} \in \Sigma$
6. $\qquad \delta := \delta \cup \{(\mathrm{a}, \mathrm{a}) \to (\varepsilon, \vec{\ })\}$

Пример построения распознающего МПА для разделённой грамматики

Грамматика G_1:

- $S \to \mathtt{a}DS \mid \mathtt{b}D\mathtt{c}$,
- $D \to \mathtt{a}D \mid \mathtt{cc}$.

Управляющая таблица распознающего МПА для G_1 и её оптимизированный вариант:

	a	b	c	$\dashv$
S	$\mathtt{a}DS$	$\mathtt{b}D\mathtt{c}$		
D	$\mathtt{a}D$		$\mathtt{cc}$	
a	$\varepsilon, \vec{}$			
b		$\varepsilon, \vec{}$		
c			$\varepsilon, \vec{}$	
∇				✓

	a	b	c	$\dashv$
S	$DS, \vec{}$	$D\mathtt{c}, \vec{}$		
D	$D, \vec{}$		$\mathtt{c}, \vec{}$	
c			$\varepsilon, \vec{}$	
∇				✓

Протокол обработки цепочки оптимизированным МПА для разделённой грамматики

Такт	Позиция указателя	Содержимое стека
1	$\diamond$ aaccbccc $\dashv$	$S\nabla$
2	a $\diamond$ accbccc $\dashv$	$DS\nabla$
3	aa $\diamond$ ccbccc $\dashv$	$DS\nabla$
4	aac $\diamond$ cbccc $\dashv$	c$S\nabla$
5	aacc $\diamond$ bccc $\dashv$	$S\nabla$
6	aaccb $\diamond$ ccc $\dashv$	Dc∇
7	aaccbc $\diamond$ cc $\dashv$	cc∇
8	aaccbcc $\diamond$ c $\dashv$	c∇
9	aaccbccc $\diamond$ $\dashv$	∇

Последовательность (с удалёнными подряд идущими дубликатами) конкатенаций прочитанного префикса входной цепочки и содержимого стека (без ∇) есть левый вывод входной цепочки в грамматике G_1:

S, aDS, aaDS, aacc$S$, ~~aacc$S$~~, aaccb$D$c, aaccbccc, ~~aaccbccc~~, ~~aaccbccc~~.

Литература

Основная литература

- Замятин А. П., Шур А. М. Языки, грамматики, распознаватели: Учебное пособие. Екатеринбург : Изд-во Урал. ун-та, 2007 (электронный вариант книги — на `http://elar.usu.ru`, поиск).

Дополнительная литература

- Ахо А., Лам М., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструментарий. М.: ООО "И.Д. Вильямс", 2008.
- Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. М.: Мир, 1978.
- Мартыненко Б. К. Языки и трансляции: Учеб. пособие. СПб.: Издательство С.-Петербургского университета, 2004 (электронный вариант книги — на `http://www.math.spbu.ru/user/mbk`).