Синтаксический анализ

Гладков Артемий Николаевич

09.10.2024

Задача синтаксического анализа

- Требуется установить, имеет ли строка лексем **структуру**, заданную синтаксисом языка, и зафиксировать эту структуру.
- Снова надо решать задачу разбора: дана строка лексем, и надо определить, выводима ли она в грамматике, определяющей синтаксис языка.
- Для задания синтаксиса языка используют КС-грамматики.

Как вести разбор?

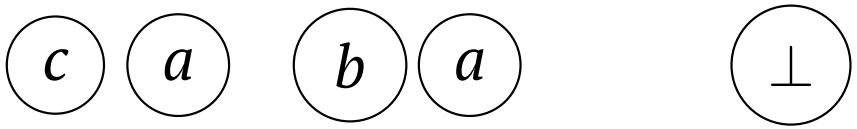
- Существуют методы анализа, применимые ко всему классу КС-грамматик и имеющие сложность $O(n^3)$ для разбора строк длины n (алгоритм Кока-Янгера-Касами) либо сложность $O(n^2)$ (алгоритм Эрли).
- Использование таких алгоритмов оправдано только в том случае, если для языка не существует более простой КС-грамматики.
- Существуют методы, применимые к разным подклассам КС-грамматик, работающие за $O(n)$. Один из таких методов мы и рассмотрим.

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \perp\}, S, P)$

$$P: \begin{array}{l} S \to AB\perp \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\perp \in L(G)?$



Цепочка выводов:

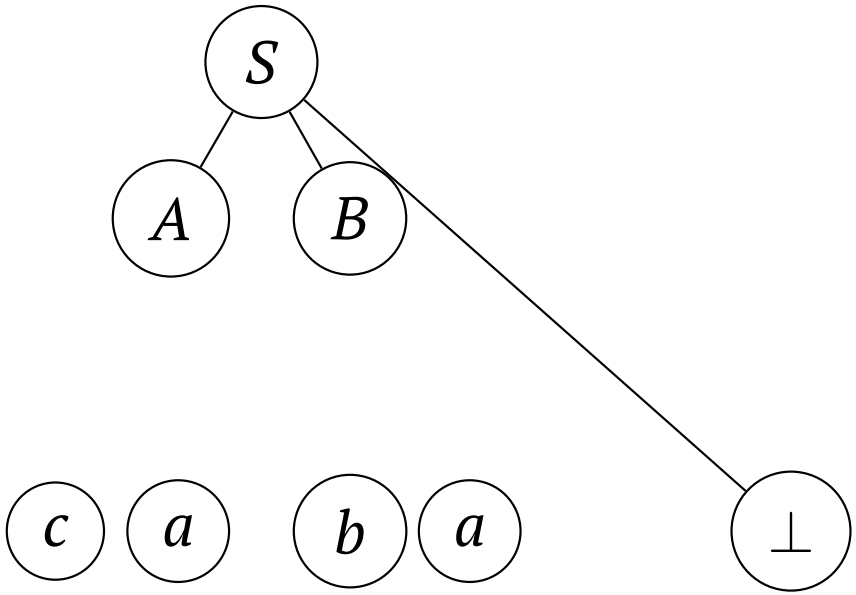
S

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \bot\}, S, P)$

$$P: \begin{array}{l} S \to AB\bot \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\bot \in L(G)?$



Цепочка выводов:

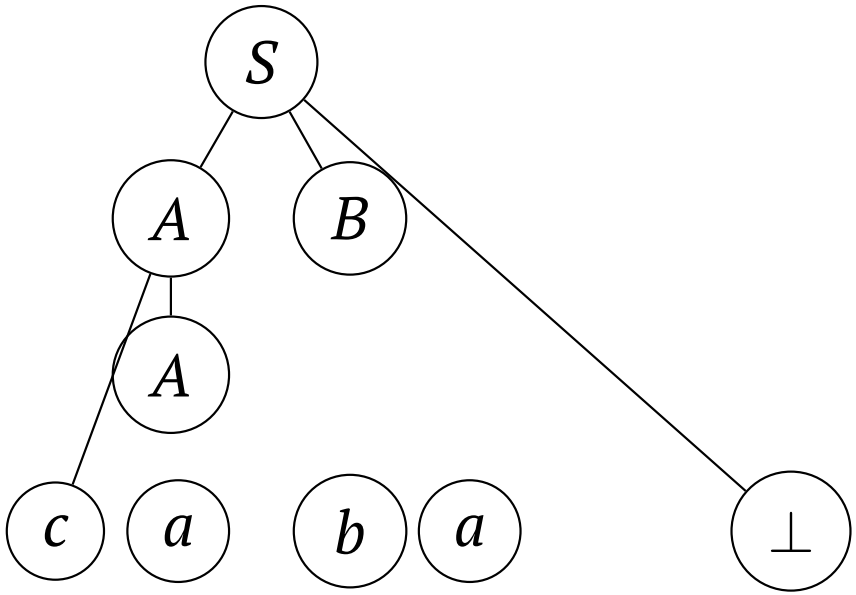
$$S \Rightarrow AB\bot$$

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \bot\}, S, P)$

$$P: \begin{array}{l} S \to AB\bot \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\bot \in L(G)?$



Цепочка выводов:

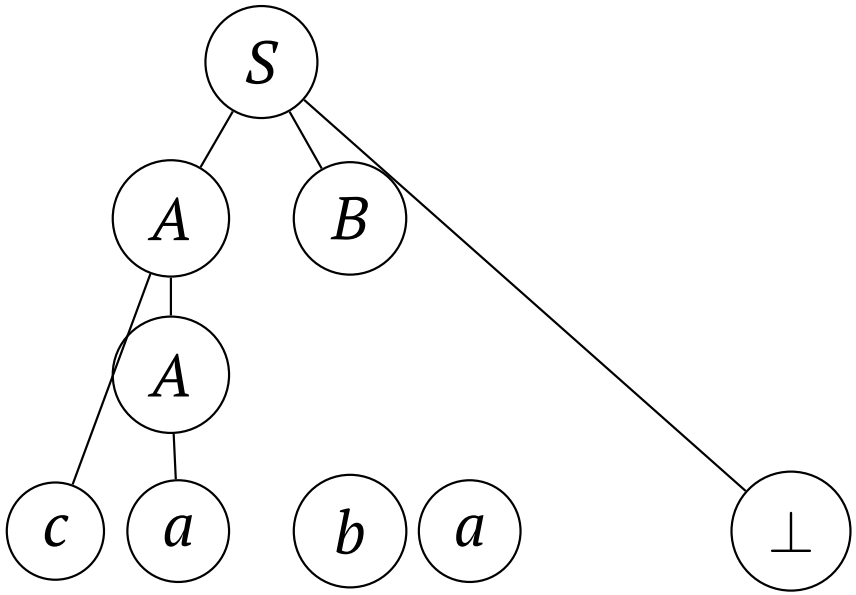
$$S \Rightarrow AB\bot \Rightarrow cAB\bot$$

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \perp\}, S, P)$

$$P: \begin{array}{l} S \to AB\perp \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\perp \in L(G)?$



Цепочка выводов:

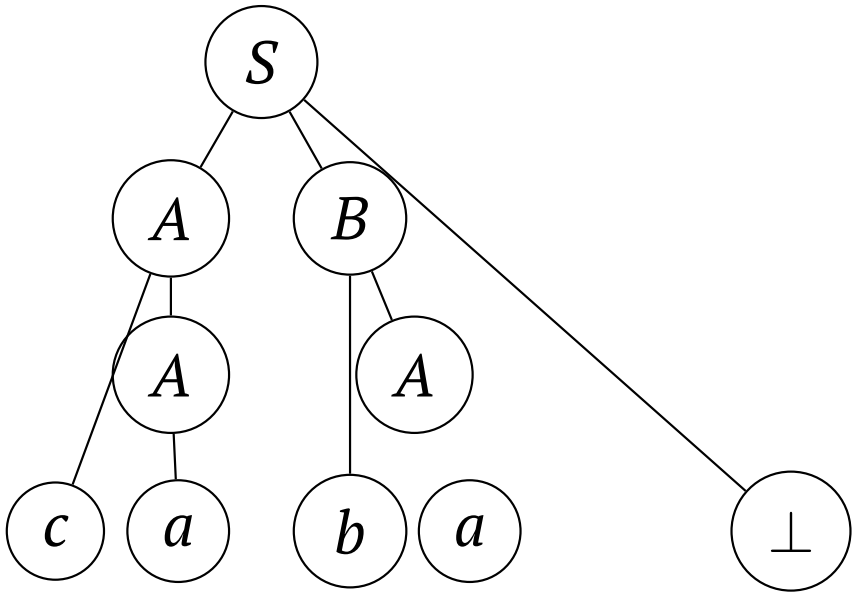
$$S \Rightarrow AB\perp \Rightarrow cAB\perp \Rightarrow caB\perp$$

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \bot\}, S, P)$

$$P: \begin{array}{l} S \to AB\bot \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\bot \in L(G)?$



Цепочка выводов:

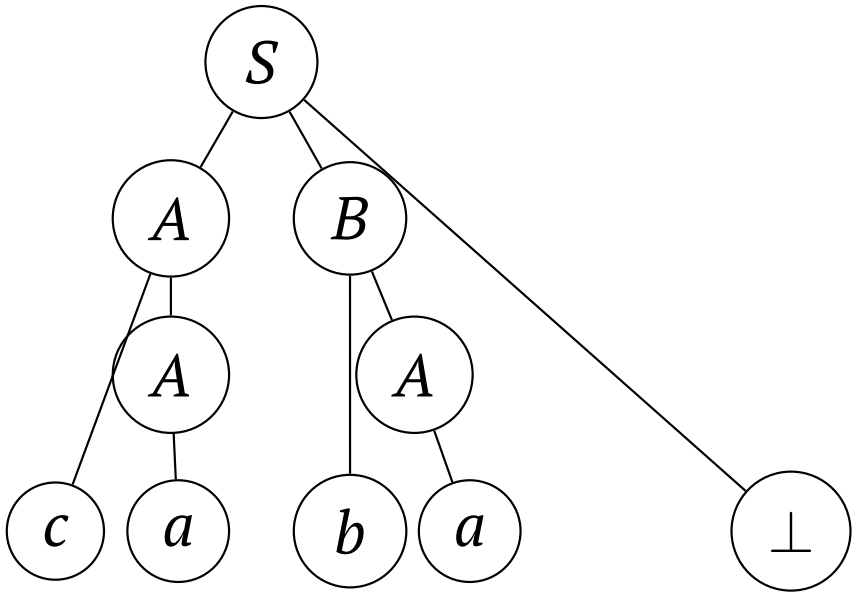
$$S \Rightarrow AB\bot \Rightarrow cAB\bot \Rightarrow caB\bot \Rightarrow cabA\bot$$

Метод рекурсивного спуска

$G = (\{S, A, B\}, \{a, b, c, \bot\}, S, P)$

$$P: \begin{array}{l} S \to AB\bot \\ A \to a \mid cA \\ B \to bA \end{array}$$

$caba\bot \in L(G)?$



Цепочка выводов:

$$S \Rightarrow AB\bot \Rightarrow cAB\bot \Rightarrow caB\bot \Rightarrow cabA\bot \Rightarrow caba\bot$$

Реализация алгоритма рекурсивного спуска

```
#include <stdio.h>
// Символ конца строки
#define EOF '&'
// Текщщий исмвол
int c;
// Ситаем строку из файла
FILE* fp;
void A();
void B();
//функция обработки ошибок
void ERROR();
void S() {
   A(); B();
   if (c != EOF) ERROR();
}
void A() {
   if (c == 'a')
      c = fgetc(fp);
   else if (c == 'c') {
      c = fgetc(fp);
      A();
   }
   else ERROR();
}
```

```
void B() {
   if (c == 'b'){
      c = fgetc(fp);
      A();
   }
   else ERROR();
}
void ERROR() {
   printf("ERROR!!!");
   throw "error";
}
int main() {
   fp = fopen("data", "r");
   c = fgetc(fp);
   S();
   printf("SUCCESS!!!");
   return 0;
}
```

Синтаксис M-языка

$P \to \textbf{program}\ D_1\ B\bot$
$D_1 \to \textbf{var}\ D\ \{;D\}$
$D \to I\{;I\} : [\textbf{int}|\textbf{bool}]$
$B \to \textbf{begin}\ S\{;S\}\ \textbf{end}$
$S \to I := E \mid \textbf{if}\ E\ \textbf{then}\ S\ \textbf{else}\ S \mid \textbf{while}\ E\ \textbf{do}\ S \mid B \mid \textbf{read}(I) \mid \textbf{write}(E)$
$E \to E_1 \mid E_1[= \mid < \mid > \mid ! =]E_1$
$E_1 \to T\{[+|-|\textbf{or}]T\}$
$T \to F\{[*|/|\textbf{and}]F\}$
$F \to I \mid N \mid L \mid \textbf{not}\ F \mid (E)$
$L \to \textbf{true} \mid \textbf{false}$
$I \to C \mid IC \mid IR$
$N \to R \mid NR$
$C \to a \mid b \mid \ldots \mid z \mid A \mid B \mid \ldots \mid Z$
$R \to 0 \mid 1 \mid \ldots \mid 9$

Синтаксис M-языка

$P \to$ **program** $D_1\ B\perp$
$D_1 \to$ **var** $D\ \{;D\}$
$D \to I\{;I\} : [$**int**$|$**bool**$]$
$B \to$ **begin** $S\{;S\}$ **end**
$S \to I := E \mid$ **if** E **then** S **else** $S \mid$ **while** E **do** $S \mid B \mid$ **read**$(I) \mid$ **write**(E)
$E \to E_1 \mid E_1[= \mid < \mid > \mid ! =]E_1$
$E_1 \to T\{[+|-|$**or**$]T\}$
$T \to F\{[*|/|$**and**$]F\}$
$F \to I \mid N \mid L \mid$ **not** $F \mid (E)$
$L \to$ **true** $\mid$ **false**

В данном случае I (Идентификаторы) и N (числовые константы) рассматриваются как целые лексемы. То же самое касается и ключевых слов и разделителей вроде $:=$ и $!=$.

Применимость метода рекурсивного спуска

Метод применим в том случае, если каждое правило грамматики имеет вид:

- либо $A \to \alpha$, где $\alpha \in (N \cup T)^*$ и это единственное правило вывода для этого нетерминала.
- либо $A \to a_1\alpha_1 \mid \ldots \mid a_n\alpha_n$, где $a_i \in T, \forall i = \overline{1,n}$ и $a_i \neq a_j$, при $i \neq j$.

Описанные условия являются достаточными, но не необходимыми.

Замечание

То есть, аналогично идее лексического разбора, мы будем прочитывать по **одномой** лексеме и, в зависимости от её значения, выбирать следующий шаг.

Применимость метода рекурсивного спуска

Метод применим в том случае, если каждое правило грамматики имеет вид:

- либо $A \to \alpha$, где $\alpha \in (N \cup T)^*$ и это единственное правило вывода для этого нетерминала.
- либо $A \to a_1\alpha_1 \mid \ldots \mid a_n\alpha_n$, где $a_i \in T, \forall i = \overline{1,n}$ и $a_i \neq a_j$, при $i \neq j$.

Описанные условия являются достаточными, но не необходимыми.

Замечание

То есть, аналогично идее лексического разбора, мы будем прочитывать по **одномой** лексеме и, в зависимости от её значения, выбирать следующий шаг.

Применимость метода рекурсивного спуска

- Часто в языках программирования встречаются конструкции вида $L \to a \mid a, L$ или сокращённо $L \to a\{,\ a\}$.
- Пример таких конструкций: параметры функций, арифметические выражения и так далее.
- При попытке вывести строку a, a, a, a можно остановиться на вариантах a либо a, a либо a, a, a.
- В таких случаях будем выбирать самую длинную подстроку, чтобы сделать разбор детерминированным.

Реализация L:

```
void L() {
   if (c != 'a') ERROR();
   while ((c = fgetc(fp)) == ',')
      if ((c = fgetc(fp)) != 'a')
         ERROR();
}
```

Применимость метода рекурсивного спуска

Также могут встречаться эпсилон-продукции. Например: $A \to aA \mid \epsilon$.
В синтаксисе языка это будет означать опциональность конструкции.
Реализация будет выглядеть так:

```
void A(void) {
   if (c == 'a') {
      c = fgetc(fp);
      A();
   }
}
```

Замечание

Для корректной работы необходимо, чтобы в грамматике не имелось продукций вида $B \to \alpha A a \beta, \forall \alpha . \beta \in (N \cup T)*, B \in N$. То есть, чтобы после нетерминала A не мог стоять символ a, иначе возникает неоднозначность.

Применимость метода рекурсивного спуска

Также могут встречаться эпсилон-продукции. Например: $A \to aA \mid \epsilon$.
В синтаксисе языка это будет означать опциональность конструкции.
Реализация будет выглядеть так:

```
void A(void) {
   if (c == 'a') {
      c = fgetc(fp);
      A();
   }
}
```

Замечание

Для корректной работы необходимо, чтобы в грамматике не имелось продукций вида $B \to \alpha A a \beta, \forall \alpha . \beta \in (N \cup T)*, B \in N$. То есть, чтобы после нетерминала A не мог стоять символ a, иначе возникает неоднозначность.

Пример реализации распознавателя М-языка

$D_1 \to \textbf{var}\, D\{;\, D\}$

```
void D1()
{
   if (eq("var"))
      curr_lex = getlex();
   else
      ERROR();
   D();
   while (eq(";")) {
      curr_lex = getlex();
      D();
   }
}
```

$D \to I\{,\, I\} : [\textbf{int} \mid \textbf{bool}]$

```
void D (){
  if (!id()) ERROR();
  else {
    curr_lex = getlex();
    while (eq(",")){
      curr_lex = getlex();
      if (!id()) ERROR();
      else curr_lex = getlex();
    }
    if (!eq(":")) ERROR();
    else {
      curr_lex = getlex();
      if (eq ("int")
        || eq ("bool"))
        curr_lex = getlex();
      else
        ERROR();
    }
  }
}
```

Что дальше?

- Посмотреть видео с моей реализацией синтаксического анализатора: <ссылка>.

Что дальше?

- Посмотреть видео с моей реализацией синтаксического анализатора: <ссылка>.
- Сделать лабораторную работу №2 по синтаксическому анализу.

Лабораторная работа №2

- Разбивка по вариантам и сами варианты будут выложены в курсе в moodle. И туда же нужно прикрепить решения.
- Что требуется? — Реализовать синтаксический анализатор.
- Дедлайн: 4 недели.
- Когда сдавать? — На следующих парах или online(назначим созвон на один из вечеров).

Будущие лекции

- Занятие 3: первая половина пары – проверка контекстных условий.
- Занятие 4: первая половина пары – генерация внутренних представлений.
- Занятие 5: первая половина пары – исполнение кода.
- Занятие 6: приём Лабораторных. Или конец материала, если что-то не успеем.