LR-анализаторы и LR-алгоритм разбора.
Алексей Сорокин

1 Алгоритм перенос-свёртка

Простейшим восходящим алгоритмом синтаксического анализа является алгоритм “перенос-свёртка". Он осуществляется с помощью МП-автомата и может быть применён к произвольной контекстно-свободной грамматике $G = \langle \Sigma, N, P, S \rangle$. В этом случае требуемый автомат имеет вид $M = \langle \{q_0, q_1\}, \Sigma, N \cup \Sigma, \Delta, q_0, \{q_1\} \rangle$, где множество переходов задаётся равенством $\Delta = \{\langle q_0, S, \varepsilon \rangle \to \langle q_1, \varepsilon \rangle\} \cup \{\langle q_0, a, \varepsilon \rangle \to \langle q_0, a \rangle \mid a \in \Sigma\} \cup \{\langle q_0, \varepsilon, \alpha \rangle \to \langle q_0, A \rangle \mid (A \to \alpha) \in P\}$. Тогда нетрудно проверить равносильность $\langle q_0, u, \varepsilon \rangle \vdash \langle q_0, \varepsilon, \alpha \rangle \Leftrightarrow \alpha \vdash_G u$, откуда вытекает корректность разбора, осуществляемого построенным МП-автоматом. Однако недостатком данного автомата является его недетерминированность, так что для реализации детерминированного алгоритма разбора необходимо хранить все ветки вычислений, что приводит к экспоненциальным временным и пространственным затратам. Возможным способом устранения неоднозначности является сохранение вспомогательной информации с помощью состояний анализатора и предпросмотр следующего входного символа, что и реализовано в LR-алгоритме.

2 Множества LR-ситуаций и операции над ними

Мы будем придерживаться следующих обозначений: буквы алфавита обозначаются малыми латинскими буквами a, b, c, нетерминалы обозначаются большими буквами $A, B, C, \ldots$, слова над алфавитом Σ обозначаются буквами $u, v, w, \ldots$, а слова из терминалов и нетерминалов — греческими буквами $\alpha, \beta, \gamma, \ldots$, возможно, с нижними индексами.

Определение 1. *$\alpha = \alpha_1\beta_1 \in (\Sigma \cup N)^*$ называется активным префиксом для грамматики G, если выполняется условие $S \vdash_{G,r} \alpha_1 B u \vdash^1_G \alpha_1\beta_1\beta_2 u$.*

Можно заметить, что при успешном разборе по методу “перенос-свёртка"в стеке в любой момент времени содержится некоторый активный префикс. Также нетрудно доказать, что всякое начало активного префикса также является активным префиксом.

Пусть $\$ \notin \Sigma$, обозначим $\Sigma_\$ = \Sigma \cup \{\$\}$, $\$ будет служить обозначением завершения слова.

Определение 2. *Пусть $\alpha \in (\Sigma \cup N)^*$, обозначим*

$$E_\alpha = \begin{cases} \{\$\}, & \alpha \vdash \varepsilon; \\ \varnothing, & \text{иначе.} \end{cases}$$

Введём функцию $\mathrm{First}(\cdot)\colon (\Sigma \cup N)^* \to \mathcal{B}(\Sigma_\$)$, *положив* $\mathrm{First}(\alpha) = E_\alpha \cup \{a \mid \exists \beta \in (\Sigma \cup N)^* (\alpha \vdash_G a\beta)$.

Определение 3. *Назовём* LR-*ситуацией объект вида* $\langle A \to \alpha_1 \cdot \alpha_2, a\rangle$, *где* $(A \to \alpha_1\alpha_2) \in P, \cdot \notin (N \cup \Sigma), a \in \Sigma_\$$.

Определение 4. *Пусть* I — *произвольное множество* LR-*ситуацией, его замыканием будем называть наименьшее множество* J, *такое что:*

1. $I \subset J$,
2. *Если* $(B \to \gamma) \in P$ *и* $\langle A \to \alpha_1 \cdot B\alpha_2, a\rangle \in J$, *то* $\langle B \to \cdot\gamma, c\rangle \in J$ *для всех* $c \in \mathrm{First}(\alpha_2 a)$.

Замыкание множества I *обозначается через* $\mathrm{CLOSURE}(I)$. *Если* $B \in N$, *то через* $\mathrm{GOTO}(I, B)$ *будем обозначать множество* $\mathrm{CLOSURE}(\{\langle A \to \alpha_1 B \cdot \alpha_2, a\rangle \mid \langle A \to \alpha_1 \cdot B\alpha_2, u\rangle \in I\})$

В дальнейшем будем считать, что грамматика G содержит дополнительное правило $S' \to S$, причём нетерминал S' является стартовым и не входит в другие правила грамматики. Также в дальнейшем везде рассматриваются только правосторонние выводы в грамматике G, так что мы будем опускать нижние индексы при обозначении выводимости.

Определение 5. LR-*ситуацию* $\langle A \to \beta_1 \cdot \beta_2, a\rangle$ *будем называть допустимой для активного префикса* $\alpha\beta_1$, *если выполняется условие* $S' \vdash \alpha A u \vdash^1 \alpha\beta_1\beta_2 u$, $a \in \mathrm{First}(u)$. *Это множество будем обозначать через* $Adm(\alpha\beta_1)$.

Лемма 1. *Если* $I \subseteq Adm(\alpha)$, *то также и* $\mathrm{CLOSURE}(I) \subseteq Adm(\alpha)$.

Доказательство. По определению операции замыкания и допустимой ситуации. □

Теорема 1. *Пусть* $X_1 \dots X_k$ — *активный префикс,* $k > 0$, *тогда выполняется условие* $Adm(X_1 \dots X_k) = \mathrm{GOTO}(Adm(X_1 \dots X_{k-1}), X_k)$.

Доказательство. $\supseteq$ В силу леммы 1 достаточно доказать, что если $\langle A \to \alpha_1 \cdot X_k\alpha_2, a\rangle$ — допустимая ситуация для $X_1 \dots X_{k-1}$, то $\langle A \to \alpha_1 X_k \cdot \alpha_2, a\rangle$ будет допустимой ситуацией для $X_1 \dots X_{k-1}X_k$. Но это очевидным образом следует из определения.

$\subseteq$ Пусть $\langle A \to \beta_1 \cdot \beta_2, a\rangle \in Adm(X_1 \dots X_k)$, докажем что $\langle A \to \beta_1 \cdot \beta_2, a\rangle \in \mathrm{GOTO}(Adm(X_1 \dots X_{k-1}), X_k)$. Условие означает, что существует вывод $S' \vdash X_1 \dots X_j A\gamma \vdash^1 X_1 \dots X_j(X_{j+1} \dots X_k)\beta_2\gamma$, причём $\beta_1 = X_{j+1} \dots X_k$ и $a \in \mathrm{First}(\gamma)$. Обозначим через d длину соответствующего вывода, также будем до конца доказательства обозначать $X_{m,n} = X_m \dots X_n$.

Пусть вначале $j < k$, тогда по определению получаем, что $\langle B \to X_{j+1,k-1} \cdot X_k\beta_2, a\rangle \in Adm(X_1 \dots X_{k-1})$, что влечёт $\langle B \to X_{j+1,k} \cdot \beta_2, a\rangle \in \mathrm{GOTO}(Adm(X_{1,k-1}), X_k)$, что и требовалось. Теперь пусть $j = k$, в этом случае применим индукцию по d. Вывод представим в виде: $S' \vdash X_1, iB\gamma_2 \vdash^1 X_1, iX_{i+1,k}A\beta_1'\gamma_2 \vdash X_1, iX_{i+1,k}A\gamma_1\gamma_2$, причём $a \in \mathrm{First}(\gamma_1\gamma_2)$. Заметим, что для всякого $a \in \mathrm{First}(\gamma_1\gamma_2) \subseteq \mathrm{First}(\beta_1\gamma_2)$ найдётся $b \in \mathrm{First}(\gamma_2)$, такой что $a \in \mathrm{First}(\beta_1 b)$. Зафиксируем данный символ b.

По определению имеем, что $\langle B \to X_{i+1,k} \cdot A\beta_1', b\rangle \in Adm(X_{1,k})$, по предположению индукции получаем, что $\langle B \to X_{i+1,k} \cdot A\beta_1', b\rangle \in \mathrm{GOTO}(Adm(X_{1,k-1}), X_k)$. По определению операции замыкания получаем $\langle A \to \cdot\beta_2, a\rangle \in \mathrm{CLOSURE}(\{\langle B \to X_{i+1,k} \cdot A\beta_1', b\rangle\})$, что в силу замкнутости результата операции GOTO даёт требуемое утверждение. Лемма доказана. □

3 Алгоритм анализа по LR-таблице

LR-алгоритм представляет собой модификацию наивного алгоритма “перенос-свёртка", позволяющую учитывать информацию об уже прочитанном префиксе и следующей букве во входном потоке для принятия решения. Как и ранее, в стеке хранится некоторый активный префикс, из которого выводится прочитанное начало слова, однако теперь входящие в префикс буквы чередуются с состояниями, в которых находился анализатор после прочтения данного префикса. В общем виде стек имеет вид $q_0 A_0 q_1 A_1 \dots q_r$, причём q_0 — стартовое состояние, в самом начале помещаемое в стек и всегда находящееся на его дне. На каждом шаге алгоритма в зависимости от текущего состояния на вершине стека и следующего символа входного потока принимается решение о переносе, свёртке, а также принятии слова или отказе, означающем, что никакой непрочитанный суффикс не приведёт к слову, принимаемому анализатором.

Решение принимается на основе LR-таблицы, состоящей из 2 частей — *Action* и *Goto*. Строки LR-таблицы помечены состояниями анализатора, столбцы подтаблицы *Action* помечены элементами множества $\Sigma_\$$, а подтаблицы *Goto* — элементами множества N.

В каждой ячейке $Action(k, l)$ таблицы содержится ровно один из следующих элементов:

- $shift_j$, где j — номер состояния.
- $reduce_i$, где i — номер правила.
- *accept*, при этом соответствующий столбец помечен символом \$.
- *reject*.

В каждой ячейке $Goto(k, l)$ таблицы содержится ровно один из следующих элементов:

- $shift_j$, где j — номер состояния.
- *reject*.

Пусть функции $Left(i)$ и $Right(i)$ возвращают левую и правую части правила с номером i, тогда LR-алгоритм можно описать следующим псевдокодом:

Алгоритм 1 LR-алгоритм синтаксического разбора.

Вход: LR-таблица T, соответствующая контекстно-свободной грамматике G, слово $w\$$, $w \in \Sigma^*$.

Выход: **True**, если $w \in L(G)$, **False**, иначе.

```
▷ Инициализация:
LRStack ← Stack()                    ▷ Создаём пустой стек.
LRStack.push(q_0)
pos ← 0
```

```
▷ Шаг алгоритма
while pos < |w| + 1 do
    a ← w[pos] ▷ предпросмотр следующего символа без сдвига текущей позиции
    q ← LRStack.top()
    switch Action(q, a) do
        case shift_j
            LRStack.push(a)
            pos+= 1
            LRStack.push(j)              ▷ мы отождествляем состояния и их номера
        case reduce_i
            for i = 0, . . . , 2|Right(i)| − 1 do▷ удаляем 2|Right(i)| верхних символов
                LRStack.pop()
            end for
            q_new ← LRStack.top()
            A ← Left(i)
            if Goto(q_new, A) == shift_j then
                LRStack.push(A)
                LRStack.push(j)
            else
                returnFalse
            end if
        case accept
            return True
        case reject
            return False
end while
```

4 Построение LR-таблицы

В этом разделе мы построим определённую в предыдущем разделе LR-таблицу для достаточно широкого класса грамматик (для которых такое построение осуществимо). Состояниями LR-анализатора будут замкнутые множества LR-ситуаций. Заметим, что таких множеств конечное число (хотя уже для небольших грамматик оно может быть довольно велико).

Обозначим через q_0 стартовую ситуацию, равную CLOSURE($\langle S' \to \cdot S, \$\rangle$). Через $Clos(G)$ обозначим множество всех замкнутых множеств ситуаций, достижимых из стартовой с помощью некоторого количества применений операции GOTO.

LR-таблица строится по следующему алгоритму (через Q обозначены состояния LR-анализатора, будем считать, что в ячейках таблицы хранятся множества возможных операций, при этом при корректном завершении алгоритма каждое множество будем одноэлементным):

Алгоритм 2 Алгоритм построения LR-таблицы.

Вход: контекстно-свободная грамматика $G = \langle \Sigma, N, P, S' \rangle$.

Выход: LR-таблица, соответствующая грамматике G, если её построение возможно, $False$ иначе.

▷ Инициализация:
$Q \leftarrow Clos(G)$
for $q \in Q$ **do**
  for $A \in N$ **do**
    $Goto(q, A) = \varnothing$
  end for
  for $a \in \Sigma_\$$ **do**
    $Action(q, a) = \varnothing$
  end for
end forЗаполнение таблицы:

for $q \in Q$ **do**
  for $a \in \Sigma$ **do**
    if $\langle A \to \beta_1 \cdot a\beta_2, b \rangle \in q$ **then**
      if GOTO$(q, a) = q_j$ **then**
        $Action(q, a).add(shift_j)$
      end if
    end if
    if $\langle A \to \beta\cdot, a \rangle \in q$ **then**
      if $(A \to \beta)$ - i-ое правило грамматики **then**
        $Action(q, a).add(reduce_i)$
      end if
    end if
    if $\langle S' \to S\cdot, \$ \rangle \in q$ **then**
      $Action(q, a).add(accept)$
    end if
    if $|Action(q, a)| == 0$ **then** ▷ ничего не смогли добавить
      $Action(q, a).add(reject)$
    end if
  end for
  for $A \in N$ **do**
    if GOTO$(q, A) == q_j$ **then**
      $Goto(q, A) = j$
    else ▷ GOTO(q, A) не определено
      $Goto(q, A) = reject$
    end if
  end for
end for
▷ Проверка корректности
for $q \in Q$ **do**
  for $a \in \Sigma_\$$ **do**
    if $|Action(q, a)| > 1$ **then**
      return False;
    end if
  end for
end for

Определение 6. LR-*грамматикой будет называть такую контекстно-свободную грамматику, для которой алгоритм 2 успешно завершает работу.*

Напомним, что q_0 обозначает стартовое состояние LR-анализатора. Разрешим второму аргументу функции GOTO быть последовательностью символов, в этом случае необходимо последовательно брать в качестве второго аргумента очередной символ последовательности, а в качестве первого — результат предыдущего шага (в самом начале он равен первому аргументу функции).

Лемма 2. $q_0 = Adm(\varepsilon)$.

Доказательство. Включение слева направо следует из леммы 1, докажем обратное включение. Пусть $\langle A \to \cdot\alpha, a\rangle \in Adm(\varepsilon)$, это означает, что $S' \vdash A\gamma \vdash^1 \alpha\gamma$, причём $a \in \mathrm{First}(\alpha)$. Проведём доказательство индукцией по длине этого вывода. База индукции: вывод имеет длину 1, то есть $A = S'$, $\gamma = \varepsilon$, $\alpha = S$, $a = \$$, тогда по определению требуемая ситуация $\langle S' \to \cdot S, \$\rangle$ принадлежит q_0. Докажем шаг индукции, для чего выделим в выводе шаг, на котором появилась A: $S' \vdash B\gamma_2 \vdash^1 A\beta_2\gamma_2 \vdash A\gamma_1\gamma_2 \vdash \alpha\gamma_1\gamma_2$. По предположению индукции имеем $\langle B \to \cdot A\beta_2, b\rangle \in q_0$ для всех $b \in \mathrm{First}(\gamma_2)$. Но тогда $\langle A \to \cdot\alpha, a\rangle \in \mathrm{First}(\beta_2\gamma_2)$, то есть и для $a \in \mathrm{First}(\gamma_1\gamma_2)$, что и требовалось. Лемма доказана. □

Лемма 3. *Если во время работы* LR-*анализатора, построенного по алгоритму 2 в стеке находится некоторая последовательность $q_0X_1q_1 \dots X_kq_k$, то для всех $i \leqslant k$ верно, что $q_i = Adm(X_1 \dots X_i)$.*

Доказательство. Докажем индукцией по числу шагов LR-анализатора. В самом начале $k = 0$ и по лемме 2 имеем $q_0 = Adm(\varepsilon)$. Шаг индукции следует из задания алгоритмов 2 и 1. □

Лемма 4. *Если после прочтения слова u в стеке находится последовательность $q_0X_1q_1 \dots X_kq_k$, то $X_1 \dots X_k \vdash u$.*

Доказательство. Индукция по числу шагов алгоритма 1, база тривиальна. Докажем шаг индукции, рассмотрим последний шаг алгоритма. Если это был перенос алфавитного символа a, то получаем $X_k = a, u = va$, причём по предположению индукции $X_1 \dots X_{k-1} \vdash v$. Но тогда $X_1 \dots X_k \vdash vX_k = u$, что и требовалось. Пусть последним шагом была свёртка последовательности $Y_1 \dots Y_r$ в X_k, тогда по предположению индукции получаем $X_1 \dots X_{k-1}Y_1 \dots Y_r \vdash u$. Поскольку соответствующая свёртка возможна, то $X_k \to Y_1 \dots Y_r$ — правило грамматики, тогда $X_1 \dots X_k \vdash X_1 \dots X_{k-1}Y_1 \dots Y_r \vdash u$. Лемма доказана. □

По сути мы доказали, что LR-алгоритм не выполняет никаких шагов, кроме допустимых операций алгоритма “перенос-свёртка". То есть мы доказали, что всякое слово, принимаемое данным алгоритмом, принадлежит $L(G)$. При этом мы не пользовались спецификой LR-грамматик, на неё мы будем опираться при доказательстве обратного утверждения.

Лемма 5. *Если $S' \vdash X_1 \dots X_kv \vdash uv$, причём $X_1 \dots X_k$ — активный префикс, то после прочтения слова u в стеке в какой-то момент будет находиться последовательность $q_0X_1q_1 \dots X_kq_k$.*

Доказательство. Индукция по длине вывода u из $X_1 \dots X_k$. Базу индукции составляет случай $u = \varepsilon, k = 0$, который тривиальным образом верен. Разберём шаг индукции, возможны 2 случая: $X_k = a \in \Sigma$ и $X_k \in N$. В первом случае обозначим $u = u'a$, тогда получим, что $X_1 \dots X_{k-1} \vdash u'$, по предположению индукции имеем, что после прочтения слова v' в стеке находится последовательность $q_0 X_1 q_1 \dots X_{k-1} q_{k-1}$. Поскольку $X_1 \dots X_{k-1} a$ является активным префиксом, множество q_{k-1} обязано содержать некоторую ситуацию вида $\langle A \to \beta_1 \cdot a \beta_2, b \rangle$. Но отсюда следует, что в состоянии q_{k-1} применима инструкция $shift_j$ для некоторого номера j, после применения которой в стек будет дополнительно перенесена буква a, чего нам и хотелось. Первый случай разобран.

Во втором случае вывод имел вид $X_1 \dots X_k \vdash X_1 \dots X_{k-1} Y_1 \dots Y_r \vdash u$. По предположению индукции после прочтения слова u в стеке однажды появится последовательность $q_0 X_1 q_1 \dots X_{k-1} q_{k-1} Y_1 q'_1 \dots Y_r q'_r$. При этом из наличия правостороннего вывода $S' \vdash X_1 \dots X_k v \vdash^1 X_1 \dots X_{k-1} Y_1 \dots Y_r v$ следует, что ситуация $\langle X_k \to Y_1 \dots Y_r \cdot, b \rangle$ будет допустимой для данного активного префикса. Здесь b — первый символ слова v. Но это означает, что данная ситуация будет принадлежать q'_r и следовательно, в данном состоянии допустима свёртка по правилу $X_k \to Y_1 \dots Y_r$, если b является первым непрочитанным символом. Таким образом, со стека будут удалены $2r$ верхних символов и помещён символ X_k, что и требовалось. Лемма доказана. □

Из лемм 4 и 5 следует следующая теорема.

Теорема 2. LR-*анализатор, построенный по грамматике G, принимает в точности слова из языка $L(G)$.*

Теорема 3. *Всякая грамматика, допускающая построение* LR-*анализатора, является однозначной.*

Доказательство. Упражнение. □

Теорема 4. *Алгоритм 1 имеет линейную сложность разбора.*

Доказательство. Упражнение. □