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

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*

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

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

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* $c \in \text{First}(\alpha_2 a)$.

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

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* $S' \vdash \alpha A u \vdash \alpha\beta_1\beta_2 u$, $a \in \text{First}(a)$. *Это множество будем обозначать через* $Adm(\alpha\beta_1)$.

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

Доказательство. По определению операции замыкания и допустимого префикса. □

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

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 $\langle A \to \beta_1 \cdot \beta_2, a\rangle \in$ GOTO$(Adm(X_1 \dots X_{k-1}), X_k)$. Доказательство проведём индукцией по длине вывода длине слова β_1. Докажем базу индукции.

По определению имеем $S' \vdash X_1 \dots X_k A u \vdash X_1 \dots X_k\beta_2 u$. По определению правостороннего вывода получаем $S' \vdash X_1 \dots X_j B u_2 \vdash^1 X_1 \dots X_j X_{j+1} \dots X_k\gamma u_2 \vdash X_1 \dots X_k A u_1 u_2 \vdash X_1 \dots X_k\beta_2 u$ для некоторых $B \in N, \gamma \in (\Sigma \cup N)^*, u_1, u_2 \in \Sigma^*$, причём $u_1u_2 = u$. По определению получаем $\langle B \to X_{j+1} \dots X_{k-1} \cdot X_k\gamma, c\rangle \in Adm(X_1 \dots X_{k-1})$ для некоторого $c \in \text{First}(u_2)$, что очевидным образом влечёт $\langle B \to X_{j+1} \dots X_k \cdot \gamma, c\rangle \in$ GOTO$(Adm(X_1 \dots X_{k-1}), X_k)$. Поскольку A — самый левый нетерминал, выводимый из γ в некотором правостороннем выводе, то по индукции можно доказать, что при применении операции замыкания он рано или поздно возникнет после точки в некоторой ситуации в множестве GOTO$(Adm(X_1 \dots X_{k-1}), X_k)$, что приведёт к тому, что данное множество содержит и ситуацию $\langle A \to \cdot\beta_2, a\rangle$ для нужного a (детали доказательства опущены).

Теперь докажем шаг индукции. Пусть $\langle A \to \beta_1' X_k \cdot \beta_2, a\rangle \in Adm(X_1 \dots X_k)$, тогда по определению допустимой ситуации легко доказать, что $\langle A \to \beta_1' \cdot X_k \beta_2, a\rangle \in Adm(X_1 \dots X_{k-1})$, откуда имеем $\langle A \to \beta_1' X_k \cdot \beta_2, a\rangle \in \mathrm{GOTO}(Adm(X_1 \dots X_{k-1}), X_k)$. Теорема 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, при этом соответствующий столбец помечен элементом множества $\Sigma_\$$.
- $reduce_i$, 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 длину левой и правой 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, . . . , |Right(i)| do
                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):

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 второму аргументу функции).

Лемма 2.

1. $q_0 = Adm(\varepsilon)$.
2. $\mathrm{GOTO}(q_0, X_1 \dots X_k) = Adm(X_1 \dots X_k)$.

Доказательство. 1) По определению допустимой ситуации. 2) Следует из леммы 1. □

Лемма 3. *Во время работы* LR-*анализатора, построенного по алгоритму 2, после прочтения начала и входного слова в стеке находится некоторая последовательность* $q_0 X_1 q_1 \dots X_k q_k$, *такая что*

1. $q_i = Adm(X_1 \dots X_i)$,
2. $X_1 \dots X_k \vdash u$.

Доказательство. 1) Следует из предыдущей леммы. 2) Достаточно доказать, что все действия LR-анализатора являются допустимы действиями алгоритма “перенос-свёртка". Для $shift$ это очевидно, докажем для $reduce$, для этого достаточно показать, что в момент применения команды $reduce_i$ на вершине стека действительно находится правая часть правила с номером i, пусть оно имеет вид $A \to \alpha$. Из алгоритма построения следует, что ситуация $(A \to \alpha\cdot, b)$, где b — следующий символ входного слова, является допустимой для $X_1 \dots X_k$, но это автоматически влечёт, что α является суффиксом данной последовательности, что и требовалось. □

Следствие 1. *Всякое слово, принимаемое* LR-*анализатором, построенным по грамматике* G, *принадлежит языку* $L(G)$.

Теперь докажем обратное утверждение, что всякое слово, порождаемое грамматикой G, распознаётся построенным LR-анализатором. Для этого достаточно доказать, что всякая операция, выполняемая алгоритмом “перенос-свёртка"в процессе разбора, будет выполняться и LR-анализатором. Пусть в данный момент прочитано слово u, которое “свёрнуто"в активный префикс α, и следующей операцией является перенос символа a. В этом случае найдётся вывод $S' \vdash \alpha_1 A u_2 \vdash^1 \alpha_1\alpha_2 a\beta_1 u_2 \vdash \alpha_1\alpha_2 a u_1 u_2$, причём выполняются равенства $\alpha = \alpha_1\alpha_2$ и $u = u_1 u_2$. Тогда ситуация $\langle A \to \alpha_2 \cdot a\beta, c\rangle$, где $c \in \mathrm{First}(u_2)$, является допустимой для α. Следовательно, в текущем состоянии на вершине стека допустим перенос a.

Пусть следующей операцией является свёртка по правилу $A \to \alpha_2$, тогда существует вывод $S' \vdash \alpha_1 A v \vdash^1 \alpha_1\alpha_2 v \vdash uv$. Аналогично,ситуация $\langle A \to \alpha_2\cdot, c\rangle$, где $c \in \mathrm{First}(v)$, будет допустимой для α. Заметим, что следующим входным символом является как раз c, поэтому допустима свёртка по данному правилу.

Таким образом, доказана теорема.

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**
        $Goto(q, A) = \varepsilon$
    end for
    for $a \in \Sigma_\$$ **do**
        $Action(q, a) = \varepsilon$;
    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
        else if $\langle A \to \beta\cdot, a \rangle \in q$ **then**
            if $(A \to \beta)$ - i-ое правило грамматики **then**
                $Action(q, a).add(shift_j)$
            end if
        else if $\langle S' \to S\cdot, \$ \rangle \in q$ **then**
            $Action(q, a).add(accept)$
        else$Action(q, a).add(reject)$
        end if
    end for
    for $A \in N$ **do**
        if GOTO$(q, A) == shift_j$ **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

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.*