Алгоритмы преобразования бесконтекстных грамматик в L-графы

Ли Цзямянь

***Аннотация*—Теория формальных языков занимает важное место в компьютерной науке, где бесконтекстные (контекстно-свободные) грамматики благодаря своей мощной выразительности широко применяются в таких областях, как проектирование языков программирования, обработка естественного языка и построение компиляторов. Традиционно основным инструментом распознавания и обработки бесконтекстных языков были магазинные автоматы, однако разрозненность описаний переходов между состояниями и операций с магазином в виде отдельных команд усложняет анализ вычислений, затрудняя и практическое применение. L-графы задуманы как более совершенный и компактный инструмент представления и анализа формальных языков, благодаря наглядной графической структуре и чёткой семантике путей на графе, отражающих вычисления.**

В данной статье предлагается систематизированный алгоритм преобразования, способный трансформировать произвольную бесконтекстную грамматику в эквивалентный бесконтекстный L-граф. Данный метод выполняет наглядное взаимно-однозначное преобразование грамматических структур в графические. В отличие от магазинных автоматов, L-графы не содержат явных операций с магазином (стеком), используя скобочные пометки для реализации рекурсивной структуры, что делает их использование более интуитивно понятным. Кроме того, в работе показана возможность эффективного синтаксического анализа с помощью L-графов, что имеет практическую ценность. L-графы, сохраняя выразительную мощность бесконтекстных грамматик, легче поддаются оптимизации (устранение избыточных вершин, дуг, уменьшение циклической сложности и др.), что повышает их эффективность как для теоретических исследований, так и для практических приложений.

***Ключевые слова*—формальные языки, контекстно-свободная грамматика, L-граф, синтаксический анализ.**

I. ВВЕДЕНИЕ

Теория формальных языков, с момента своего возникновения в 1950-х годах, когда Ноам Хомский предложил систему формальных грамматик, стала фундаментальной основой для таких областей, как информатика, лингвистика и математика. В иерархии классов языков, предложенной Хомским, важное место занимает класс бесконтекстных (контекстно-свободных) языков. Бесконтекстные языки задаются бесконтекстными (контекстно-свободными) грамматиками, которые благодаря своей мощной выразительной мощности стали ключевым инструментом для описания синтаксиса языков программирования и структуры естественных языков. Традиционно магазинные автоматы, как эквивалентная вычислительная модель бесконтекстной грамматики, широко использовались для распознавания и анализа бесконтекстных языков. Однако механизм работы со магазином (стеком) в магазинном автомате усложняет переходы между состояниями и затрудняет интуитивное понимание, особенно при практическом применении в синтаксическом анализе и проектировании компиляторов, где явное управление стеком увеличивает сложность разработки и оптимизации алгоритмов.

В [1] был предложен новый инструмент анализа формальных языков — L-графы. Представляя формальные языки посредством графовой структуры, L-графы лучше отражают вычисления благодаря понятной семантике путей на графе, способны выразить рекурсию с помощью парных циклов, синхронизируемых вложенными скобочными гнездами, становясь полезным инструментом для исследований в области формальных языков. L-графы используют вершины вместо состояний, дуги вместо переходов между состояниями, скобочные пометки на дугах вместо явных операций с магазином (в магазинных автоматах), что позволяет избежать сложностей, связанных с явным управлением магазином. Такой подход не только более наглядно отражает структуру синтаксических правил, но и более естественно описывает иерархичность и рекурсивность языков. Например, при анализе вложенных арифметических выражений или управляющих структур программ механизм сопоставления путей в L-графах оказывается более понятным и удобным для реализации, чем операции с магазином в магазинном автомате.

В данной статье предлагается алгоритм преобразования $A_{G\to L}$, который позволяет преобразовать произвольную бесконтекстную грамматику $G = \langle T, N, R, S\rangle$ в эквивалентный бесконтекстный L-граф $L = \langle V, \Sigma, P, E, I, F\rangle$. Алгоритм реализуется в два этапа:

- Построение вершин. Для каждого синтаксического правила создается набор вершин, представляющих начальное, промежуточное и конечное состояния правила.
- Построение дуг. В зависимости от терминалов и нетерминалов в правилах генерируются соответствующие дуги, причем рекурсивные вызовы нетерминалов реализуются через скобочные пометки.

В работе с помощью метода математической индукции доказывается эквивалентность языков, задаваемых произведенным алгоритмом L-графом и исходной грамматикой. Доказательство основано на следующих двух леммах.

Статья получена 30 сентября 2025 г.

Ли Цзямянь, Университет МГУ – ППИ в Шэньчжэне (lijiamian0804@live.com).

- (Лемма 1) Любая цепочка, порождаемая исходной бесконтекстной грамматикой, является пометкой успешного пути в произведенном алгоритмом L-графе.
- (Лемма 2) Пометка любого успешного пути в произведенном алгоритмом L-графе соответствует цепочке, порождаемой исходной грамматикой.

По сравнению с традиционными магазинными автоматами, L-графы демонстрируют значительные преимущества в синтаксическом анализе:

- Более наглядная рекурсивная структура: Скобочные пометки непосредственно отражают вложенность нетерминалов, устраняя сложности, связанные с операциями со стеком в магазинном автомате.
- Более прозрачный процесс анализа: графовая структура L-графов естественным образом подходит для алгоритмов синтаксического анализа на основе путей, таких как $LL(1)$-анализ, при этом процесс анализа позволяет четко видеть текущую позицию в исходной грамматике.

Далее для демонстрации практической применимости L-графов на примере $LL(1)$-грамматики подробно описывается, как использовать L-граф для нисходящего предсказывающего анализа. Построенные предложенным алгоритмом L-графы, полностью сохраняя структурные свойства исходной грамматики (линейность, отсутствие самовставлений и др.), делают процесс синтаксического анализа более наглядным.

Настоящее исследование предлагает новые возможности в использовании L-графов как для теоретического анализа формальных языков, так и для их практического применения. Бесконтекстные L-графы имеют потенциал для более широкого использования в проектировании компиляторов, обработке естественных языков и других областях, особенно в задачах, требующих эффективной протоколируемой обработки рекурсивных структур. Кроме того, L-графы более общего вида, чем бесконтекстные, могут быть полезны для распознавания более сложных языковых свойств (например, для анализа по контекстно-зависимым грамматикам), возможно открывая новые направления для развития теории формальных языков.

Структура статьи следующая. В разделе II представлены определения, связанные с L-графами, а также некоторые базовые концепции. В разделе III описан алгоритм преобразования бесконтекстных грамматик в эквивалентные бесконтекстные L-графы и доказана его корректность. В разделе IV предложен метод анализа грамматик с использованием построенных L-графов, приведены примеры и проведено сравнение с магазинными автоматами.

II. L-ГРАФ

Для удобства описания алгоритма здесь приводятся определение L-графа и связанные понятия [1].

Для определения L-графа, сначала необходимо ввести несколько вспомогательных понятий.

A. Вспомогательные понятия

Пусть P_{left} и P_{right} — два непересекающихся алфавита и пусть существует биекция $\phi : P_{left} \to P_{right}$.

Определение 1: Назовем тройку $\langle P_{left}, P_{right}, \phi \rangle$ *скобочным множеством*.

Определение 2: Назовем элементы множества P_{left} *открывающими* скобками, а элементы множества P_{right} — *закрывающими* скобками.

Пусть P — скобочное множество, $P = \langle P_{left}, P_{right}, \phi \rangle$. Введем следующие понятия и обозначения, связанные с P:
$P_\langle$ — это множество открывающих скобок, $P_\langle = P_{left}$;
$P_\rangle$ — это множество закрывающих скобок, $P_\rangle = P_{right}$;
$P_{\langle\rangle}$ — это множество открывающих и закрывающих скобок, $P_{\langle\rangle} = P_\langle \cup P_\rangle$;
P_{pair} — это множество пар скобок, $P_{pair} = \{\, (a, b) \mid a \in P_{left}, b \in P_{right}, b = \phi(a) \,\}$.

Язык, порождаемый грамматикой с начальным символом S и правилами вида

$$S \to a_i S b_i S | \varepsilon, \tag{1}$$

где $(a_i, b_i) \in P_{pair}, i = 1, ..., |P_\langle|$, назовем *языком Дика над скобочным множеством* P и обозначим его $D(P)$.

Цепочки в алфавите $P_{\langle\rangle}$ будем называть *скобочными последовательностями*. Скобочные последовательности, принадлежащие языку Дика, будем называть *правильными*. Правильные скобочные последовательности будем называть *скобочными системами* над алфавитом $P_{\langle\rangle}$.

B. Определение L-графа

Определение 3: *L-граф* задается шестеркой

$$G = \langle V, \Sigma, P, E, I, F \rangle, \tag{2}$$

где:
V — конечное множество вершин;
Σ — алфавит языка;
P — скобочное множество;
$I \subseteq V$ — множество начальных вершин;
$F \subseteq V$ — множество заключительных вершин;
$E \subseteq V \times (\Sigma \cup \{\varepsilon\}) \times (P \cup \{\varepsilon\}) \times (P \cup \{\varepsilon\}) \times V$ — множество дуг.
L-граф называется *бесконтекстным*, если
$E \subseteq V \times (\Sigma \cup \{\varepsilon\}) \times (P \cup \{\varepsilon\}) \times \{\varepsilon\} \times V$

Пример 1: Бесконтекстный L-граф $L(D) = \{a^n b^n | n \geq 1\}$:

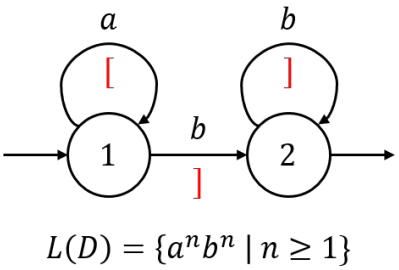

Рис. 1. Бесконтекстный L-граф

Пусть $\pi \in E, \pi = \langle q, a, d_1, d_2, p \rangle$. Введем вспомогательные понятия и обозначения, связанные с дугой π:
$beg(\pi)$ — начальная вершина дуги, $beg(\pi) = q$;
$end(\pi)$ — заключительная вершина дуги, $end(\pi) = p$;
$\omega(\pi)$ — пометка дуги, $\omega(\pi) = a$;

$\iota_1(\pi)$ — первый(скобочный) след, $\iota_1(\pi) = d_1$;
$\iota_2(\pi)$ — второй(скобочный) след, $\iota_2(\pi) = d_2$.

Пусть $T = p_0\pi_1p_1...\pi_kp_k, \pi_i \in E, i = 1, ..., k, p_i \in V, i = 0, ..., k, beg(\pi_i) = p_{i-1}, end(\pi_i) = p_i$.

Для $i = 0$ путь T назовем пустым. Он представляется одной вершиной. Непустой путь T однозначно определяется своими дугами и его можно обозначить $T = \pi_1\pi_2...\pi_k$. В дальнейших рассуждениях путь будем определять либо вершиной $T_0 = p$, либо последовательностью дуг $T = \pi_1...\pi_k$.

Длина пути — это количество дуг в пути, обозначается $|T|$.

Начальная вершина пути (обозначение $beg(T)$) — вершина, из которой исходит первая дуга пути, $beg(T) = p_0$.

Конечная вершина пути (обозначение $end(T)$) — вершина, в которую входит последняя дуга пути, $end(T) = p_k$.

L-графы являются развитием понятия D-графа, предложенного Л.И. Станевичене [2], [3].

III. АЛГОРИТМЫ ПРЕОБРАЗОВАНИЯ БЕСКОНТЕКСТНЫХ ГРАММАТИК В L-ГРАФЫ

В данном разделе представлен алгоритм преобразования бесконтекстной грамматики $G = \langle T, N, R, S\rangle$ в эквивалентный бесконтекстный L-граф $L = \langle V,\ \Sigma,\ P,\ E,\ I,\ F\rangle$. Алгоритм построен с учетом идей, изложенных в [4]. Основная идея алгоритма заключается в построении соответствующего подграфа для каждого нетерминального символа в грамматике, тогда как терминальные символы обозначаются пометками на дугах. Используется механизм согласования скобок для обеспечения корректного входа в подграф и выхода из него. В результате получается ориентированный граф с множеством пометок, где все успешные пути должны соответствовать предложениям, порождаемым исходной грамматикой, и все предложения, порождаемые исходной грамматикой, должны иметь соответствующий успешный путь.

A. Вспомогательные понятия

Для построения алгоритма сначала необходимо определить два вспомогательных понятия.

Определение 4: Определим упорядоченную пару $(n, label)$ как вершину, обозначаемую v^n_{label}, где:
n — нетерминальный символ ($n \in N$);
$label$ — метка $label \in \{beg, end\} \cup \{(a, b)|a, b \in \mathbb{Z}^+\}$.

Если $label = beg$ или end, то $v^n_{beg}(v^n_{end})$ обозначают начальную(заключительную) вершину участка L-графа, соответствующего нетерминалу n.

Если $label = (a, b)$, то $v^n_{a,b}$ обозначает $(b + 1)$-ю вершину участка L-графа, соответствующего a-му правилу с левой частью n.

Определение 5: Определим функцию генерации вершин. Пусть нетерминал A имеет m правил $r_1, r_2, ..., r_m \in R$. Для правила $r_k : A \to \alpha_1...\alpha_n, 1 \le k \le m$ функция генерирует множество вершин:

$$Vertices(A \to \alpha_1...\alpha_n) = \{v^A_{beg}, v^A_{k,1}, ..., v^A_{k,n-1}, v^A_{end}\}.$$

Определение 6: Определим функцию преобразования правил в их графовое представление:

$$GRep : R \to (V \cup N \cup T \cup \{\varepsilon\})^*,$$

где:
R — множество правил грамматики;
V — множество вершин L-графа;
N — множество нетерминальных символов грамматики;
T — множество терминальных символов грамматики.

Пусть нетерминал A имеет m правил $r_1, r_2, ..., r_m \in R$. Для правила $r_k : A \to \alpha_1...\alpha_n, 1 \le k \le m$ функция генерирует множество вершин:

$$GRep(A \to \alpha_1...\alpha_n) = v^A_{beg}\alpha_1 v^A_{k,1}\alpha_2...\alpha_{n-1}v^A_{k,n-1}\alpha_n v^A_{end}.$$

Пример 2: Дано правило:

$$S \to aBc$$

Его соответствующее графовое представление $GRep(S \to aBc) = v^S_{beg}av^S_{1,1}Bv^S_{1,2}cv^S_{end}$. выглядит следующим образом (рис. 2):

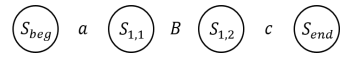

Рис. 2. Построенные графовые представления

B. Описание алгоритма

Чтобы избежать громоздких построений, мы предполагаем, что входные грамматики являются приведёнными, то есть в них отсутствуют недостижимые и бесплодные символы.

Алгоритм 1: $A_{G\to L}$:
Вход: бесконтекстная грамматика $G = \langle T, N, R, S\rangle$
Выход: бесконтекстный L-граф $L = \langle V,\ \Sigma,\ P,\ E,\ I,\ F\rangle$
Шаг 1. Положим $\Sigma = T$.
Шаг 2. Построим множество вершин V, множества начальных и конечных вершин I и F.

Применяя функцию $Vertices$ ко всем правилам G, получаем множество вершин V:

$$V = \bigcup_{r\in R} Vertices(r). \tag{3}$$

Положим $I = \{v^s_{beg}\}, F = \{v^s_{end}\}$.
Шаг 3. Построим множество дуг E.

Определим отображение графового представления правила на дуги:

$$\eta : (V \cup N \cup T \cup \{\varepsilon\})^* \to \mathcal{P}(E),$$

принимающее графовые представления правила в качестве аргумента.

Определим функцию генерации дуг из тройки:

$$arc : V \times (T \cup N \cup \{\varepsilon\}) \times V \to \mathcal{P}(E),$$

$$arc(v_1\alpha v_2) = \begin{cases} \langle v_1, \alpha, \varepsilon, \varepsilon, v_2\rangle, \\ \text{если } \alpha \in T \cup \{\varepsilon\}; \\ \{\langle v_1, \varepsilon, d_{v_1\alpha v_2}, \varepsilon, v^\alpha_{beg}\rangle, \langle v^\alpha_{end}, \varepsilon, \phi(d_{v_1\alpha v_2}), \varepsilon, v_2\rangle\}, \\ \text{если } \alpha \in N; \end{cases} \tag{4}$$

$$d_{v_1\alpha v_2}, \phi(d_{v_1\alpha v_2}) \in P.$$

Для графового представления правила общего вида $\omega = v_1\alpha_1 v_2 ... v_n\alpha_n v_{n+1}, v_i \in V, \alpha_i \in (N \cup T \cup \{\varepsilon\}), n \geq 1, i = \overline{1,n}$,

$$\eta(\omega) = \bigcup_{i=1}^{n} arc(v_i\alpha_i v_{i+1}). \quad (5)$$

Множество дуг

$$E = \bigcup_{r \in R} \eta(GRep(r)). \quad (6)$$

Замечание: Обычно для удобства обозначения, для нетерминала, кроме первой и последней вершины, остальные вершины нумеруются в порядке возрастания. Например, если нетерминал A соответствует вершинам $v^A_{beg}, v^A_{1,1}, v^A_{1,2}, v^A_{2,1}, v^A_{end}$, их можно записать как $v^A_{beg}, v^A_1, v^A_2, v^A_3, v^A_{end}$. И обычно индексы скобок обозначаются последовательно возрастающими номерами.

Пример 3: Дана бесконтекстная-грамматика G:
$P \to S@$
$S \to aSb|cA$
$A \to cA|\varepsilon$
Построить по G L-граф $L = \langle V, \Sigma, P, E, I, F\rangle$.
Шаг 1.
$\Sigma = T = \{a, b, c, @\}$
Шаг 2.
$V = \bigcup_{r \in R} Vertices(r) =$
$\{v^P_{beg}, v^P_1, v^P_{end}\} \cup \{v^S_{beg}, v^S_1, v^S_2, v^S_{end}\} \cup$
$\{v^S_{beg}, v^S_3, v^S_{end}\} \cup \{v^A_{beg}, v^A_1, v^A_{end}\} \cup \{v^A_{beg}, v^A_{end}\}$
$= \{v^P_{beg}, v^P_1, v^P_{end}, v^S_{beg}, v^S_1, v^S_2, v^S_3, v^S_{end}, v^A_{beg}, v^A_1, v^A_{end}\}$,
$I = \{v^P_{beg}\}$
$F = \{v^P_{end}\}$

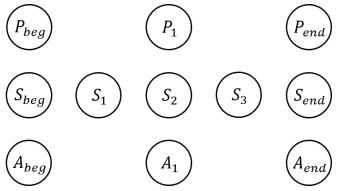

Рис. 3. Построенные вершины

Все графовые представления приведены ниже:

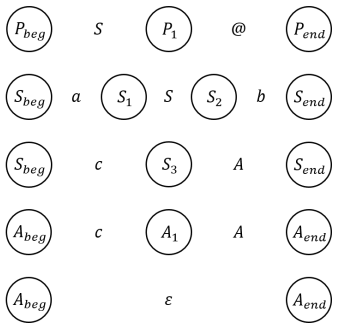

Рис. 4. Построенные графовые представления

Шаг 3.
$E = \bigcup_{r \in R} \eta(GRep(r))$,
$\eta(v^P_{beg} S v^P_1 @ v^P_{end}) = arc(v^P_{beg} S v^P_1) \cup arc(v^P_1 @ v^P_{end})$,
$\eta(v^S_{beg} a v^S_1 S v^S_2 b v^S_{end}) =$
$arc(v^S_{beg} a v^S_1) \cup arc(v^S_1 S v^S_2) \cup arc(v^S_2 b v^S_{end})$,
$\eta(v^S_{beg} c v^S_3 A v^S_{end}) = arc(v^S_{beg} c v^S_3) \cup arc(v^S_3 A v^S_{end})$,
$\eta(v^A_{beg} c v^A_1 A v^A_{end}) = arc(v^A_{beg} c v^A_1) \cup arc(v^A_1 A v^A_{end})$,
$\eta(v^A_{beg} \varepsilon v^A_{end}) = arc(v^A_{beg} \varepsilon v^A_{end})$.
Полученные дуги:
$E =$
$\{\langle v^P_{beg}, \varepsilon, (_1, \varepsilon, v^S_{beg}\rangle, \langle v^S_{end}, \varepsilon,)_1, \varepsilon, v^P_1\rangle, \langle v^P_1, @, \varepsilon, \varepsilon, v^P_{end}\rangle,$
$\langle v^S_{beg}, a, \varepsilon, \varepsilon, v^S_1\rangle, \quad \langle v^S_1, \varepsilon, (_2, \varepsilon, v^S_{beg}\rangle, \quad \langle v^S_{end}, \varepsilon,)_2, \varepsilon, v^S_2\rangle,$
$\langle v^S_2, b, \varepsilon, \varepsilon, v^S_{end}\rangle, \quad \langle v^S_{beg}, c, \varepsilon, \varepsilon, v^S_3\rangle, \quad \langle v^S_3, \varepsilon, (_3, \varepsilon, v^A_{beg}\rangle,$
$\langle v^A_{end}, \varepsilon,)_3, \varepsilon, v^S_{end}\rangle, \quad \langle v^A_{beg}, c, \varepsilon, \varepsilon, v^A_1\rangle, \quad \langle v^A_1, \varepsilon, (_4, \varepsilon, v^A_{beg}\rangle,$
$\langle v^A_{end}, \varepsilon,)_4, \varepsilon, v^A_{end}\rangle, \langle v^A_{beg}, \varepsilon, \varepsilon, \varepsilon, v^A_{end}\rangle\}$.
$P = \{(_1,)_1, (_2,)_2, (_3,)_3, (_4,)_4\}$.
Полученный L-граф:

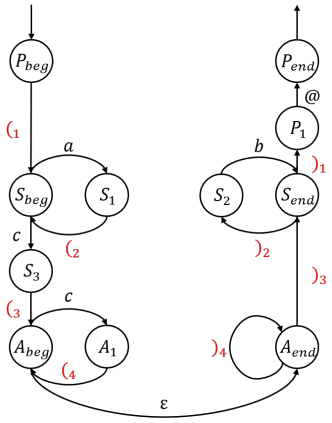

Рис. 5. L-граф соответствующий грамматике G

C. Доказательство эквивалентности

Теорема 1: Для любой бесконтекстной грамматики G, $L(A_{G \to L}(G)) = L(G)$.
Доказательство:

Для доказательства эквивалентности бесконтекстной грамматики $G = \langle T, N, R, S\rangle$ и соответствующего ей L-графа $L = \langle V, \Sigma, P, E, I, F\rangle$ необходимо доказать два утверждения:

1) $L(G) \subseteq L(A_{G \to L}(G))$. Любая цепочка α, порождаемая грамматикой G, соответствует успешному пути T в L-графе L, такому что $\omega(T) = \alpha$.
2) $L(A_{G \to L}(G)) \subseteq L(G)$. Любой успешный путь T в L-графе L соответствует цепочке $\omega(T)$, порождаемой грамматикой G.

Доказательство основано на следующем свойстве.

Свойство 1: Дана бесконтекстная грамматика: $G = (N, T, R, S)$. Для любого $A \in N$ существует новая бесконтекстная грамматика $G_A = (N, T, R, A)$.

То есть для любого нетерминала в бесконтекстной грамматике, если его рассматривать как начальный символ, получится другая бесконтекстная грамматика.

Это свойство обеспечивает рекурсивность построения L-графа: подграф L_A, соответствующий нетерминалу A, строится на основе правил грамматики G_A.

Таким образом, все базисы и предположения индукции в доказательстве применимы и к подграфам.

Лемма 1: Для любой бесконтекстной грамматики G, $L(G) \subseteq L(A_{G \to L}(G))$. То есть для любой цепочки $S \Rightarrow^* \alpha$, где $\alpha \in T^*$, существует успешный путь T в L-графе, такой что $\alpha = \omega(T)$.
Доказательство:

Базис индукции: Для 1-шагового вывода $S \Rightarrow^1 \alpha$, где $\alpha \in T^*$, алгоритм строит графовое представление: $GRep(S \to \alpha) = v^S_{beg}\alpha_1 v^S_{1,1}\alpha_2 \dots v^S_{1,n-1}\alpha_n v^S_{end}$.

Применяя функцию η, получаем дуги: $\eta(GRep(S \to \alpha)) = \bigcup_{i=1}^{n-1} arc(v^S_{1,i}\alpha_i v^S_{1,i+1}) \cup arc(v^S_{beg}\alpha_1 v^S_{1,1}) \cup arc(v^S_{1,n-1}\alpha_n v^S_{end})$.

По определению функции arc, очевидно существует путь T из v^s_{beg} в v^s_{end}, для которого $\omega(T) = \alpha$.

Предположение индукции: Предположим, что для k-шагового вывода $S \Rightarrow^k \alpha$ существует путь T, такой что $\omega(T) = \alpha$.

Шаг индукции: Для вывода $S \Rightarrow^{k+1} \alpha$ разложим его на:

1) $S \Rightarrow^1 \beta A\gamma$ (где $A \in N$),
2) $A \Rightarrow^k \delta$, причём $\beta\delta\gamma = \alpha, \delta \in T^*$.

По предположению индукции, для $A \Rightarrow^k \delta$ существует путь T_A с $\omega(T_A) = \delta$.

Для $S \Rightarrow^1 \beta A\gamma$ алгоритм строит графовое представление: $GRep(S \to \beta A\gamma) = v^S_{beg}\beta_1 v_1 \beta_2 \dots \beta_m v_l A v_l \gamma_1 v_{l+1}\gamma_2 \dots \gamma_n v^S_{end}$.

Применяя η, получаем дуги: $\langle v^S_{beg},\ \beta_1,\ \varepsilon,\ \varepsilon,\ v_1\rangle, \dots, \langle v_m,\ \varepsilon,\ d_{v_m\alpha v^A_{beg}},\ \varepsilon,\ v^A_{beg}\rangle$, $\langle v^A_{end},\ \varepsilon, \phi(d_{v_m\alpha v^A_{beg}}),\ \varepsilon,\ v_l\rangle, \dots, \langle v_{l+n-1},\ \gamma_n,\ \varepsilon,\ \varepsilon,\ v^S_{end}\rangle$.

Таким образом, существует путь T, для которого $\omega(T) = \beta\delta\gamma = \alpha$. Следовательно, $L(G) \subseteq L(A_{G\to L}(G))$ □

Лемма 2: Для любой бесконтекстной грамматики G, $L(A_{G\to L}(G)) \subseteq L(G)$. То есть для любого успешного пути T в L-графе, цепочка $\omega(T)$ может быть выведена в грамматике G: $S \Rightarrow^* \alpha$.

Доказательство:

Для удобства доказательства необходимо определить вспомогательную функцию, описывающую максимальную глубину вложенности скобок в пути. Данное определение схоже по смыслу с функцией $depth$, определённой в [3], и в данной статье используется исключительно в качестве вспомогательного средства для доказательства, поэтому приводится в упрощённой формулировке.

Определение 7: Определим $depth(T)$ как *максимальную глубину вложенности скобок* в пути T.

Пусть дан успешный путь $T = \pi_1\pi_2 \dots \pi_n$. Определим функцию $f : \{0, 1, \dots, n\} \to \mathbb{Z}$ следующим образом:

$$f(0) = 0, f(k) = f(k-1) + \delta(\pi_k), k = 1, \dots, n \qquad (7)$$

где

$$\delta(\pi) = \begin{cases} 1, & \text{если } \iota_1(\pi) \in P_\langle \\ -1, & \text{если } \iota_1(\pi) \in P_\rangle \\ 0, & \text{в остальных случаях} \end{cases} \qquad (8)$$

Тогда $depth$ пути определяется как:

$$depth(T) = \max_{0\le k\le n} f(k) \qquad (9)$$

Пример 4: Пусть дан успешный путь T такой, что $\iota_1(T) = (_1(_2)_2)_1(_3)_3$, имеет место $depth(T) = 2$.

Базис индукции ($depth(T) = 0$): Если в пути T нет скобок, то он соответствует одношаговому выводу $S \Rightarrow^1 \alpha$, и $\omega(T) = \alpha$.

Предположение индукции. Пусть для всех путей T с $depth(T) \le k$ существует вывод $S \Rightarrow^* \omega(T)$.

Шаг индукции. Доказать, что это верно и для ($depth(T) = k + 1$). Для удобства доказательства предположим, что в пути есть только одно место с $depth$, равной $k + 1$. Для случаев с несколькими такими местами доказательство можно провести по аналогии.

Путь T можно разложить на $T_1\pi_1 T_A \pi_2 T_2$, где $\omega(T_1) = \gamma_1$, $\omega(T_A) = \delta$, $\omega(T_2) = \gamma_2$, $\alpha = \gamma_1\delta\gamma_2$, $depth(T_A) = k$, $\iota_1(\pi_1)$ и $\iota_1(\pi_2)$ — парные скобки.

По предположению индукции, для T_A существует вывод $A \Rightarrow^* \delta$. И поскольку $depth(T) = k + 1$, а $depth(T_A) = k$, то обязательно существуют дуги $\langle v^S_m,\ \varepsilon,\ d_{v_m\alpha v^A_{beg}},\ \varepsilon,\ v^A_{beg}\rangle$, $\langle v^A_{end},\ \varepsilon, \phi(d_{v_m\alpha v^A_{beg}}),\ \varepsilon,\ v^S_{m+1}\rangle$, согласно алгоритму, обязательно существует правило $S \to \Gamma_1 A \Gamma_2, A \in N, \Gamma_1, \Gamma_2 \in \{T \cup N\}^*$.

Теперь добавим новое правило $S \to \Gamma_1\delta\Gamma_2$ в грамматику G, получив G_1. Очевидно, что $L(G_1) = L(G)$. В этом случае мы можем игнорировать скобки на дугах π_1 и π_2 в пути T, тогда $depth(T) \le k$. Согласно индуктивному предположению, получим, что $\omega(T)$ соответствует выводу $S \Rightarrow^* \gamma_1\delta\gamma_2$. Следовательно, $L(A_{G\to L}(G)) \subseteq L(G_1) = L(G)$.

Таким образом, можно сделать вывод: при любом значении $depth(T)$ путь T соответствует цепочке, порождённой грамматикой G, то $L(A_{G\to L}(G)) \subseteq L(G)$. □

Из Лемм 1 и 2 следует, что для любой бесконтекстной грамматики G: $L(A_{G\to L}(G)) = L(G)$. □

IV. СИНТАКСИЧЕСКИЙ АНАЛИЗ С ПОМОЩЬЮ L-ГРАФА

Для удобства демонстрации здесь в качестве примера взята относительно простая $LL(1)$-грамматика, чтобы показать, как использовать L-граф для синтаксического анализа. Что касается определения $LL(1)$-грамматики, то на него можно сослаться в учебнике [5].

По аналогии с общим определением детерминированного L-графа [6], дадим определение детерминированного бесконтекстного L-графа.

Определение 8: Определим функцию $direct(\pi)$, где π дуга L-графа $G = \langle V, \Sigma, P, E, I, F\rangle$: $direct(\pi) = \{a \in \Sigma |$существуют такие пути T_1, T_2, что $T_1\pi T_2$ — успешный путь в L-графе, $\omega(\pi T_2) = a\alpha, \alpha \in \Sigma^*\} \cup \{\varepsilon |$существуют такие пути T_1, T_2, что $T_1\pi T_2$ — успешный путь в L-графе, $\omega(\pi T_2) = \varepsilon\}$.

Определение 9: Бесконтекстный L-граф $G = \langle V, \Sigma, P, E, I, F\rangle$ называется детерминированным, если для любой вершины $v \in V$ и для любых двух дуг $\pi_1, \pi_2 \in E$, таких, что $\pi_1 \ne \pi_2, beg(\pi_1) = beg(\pi_2) = v$, выполняется условие: $direct(\pi_1) \cap direct(\pi_2) \ne \emptyset \to (\iota_1(\pi_1), \iota_1(\pi_2) \in P_\rangle)\&(\iota_1(\pi_1) \ne \iota_1(\pi_2))$.

В данном определении не рассматривается второй скобочный след, поскольку он пуст для любой дуги бесконтекстного L-графа.

Для заданной цепочки w процесс проверки её соответствия грамматике, представленной L-графом, с использованием L-графа состоит в том, чтобы найти путь $T = p_0\pi_1 p_1 \dots \pi_k p_k$ от первой начальной вершины L-графа до последней заключительной вершины, так что пометка пути T совпадает с заданной цепочкой w. В рамках данного процесса требуется организовать стек. Каждый раз, когда мы проходим через открывающую скобку, мы добавляем эту скобку в стек, а когда мы встречаем закрывающую скобку, мы удаляем с верхушки стека

парную ей открывающую. В конце процесса анализа стек должен быть пуст, то это означает, что все скобки вдоль пути являются сбалансированными.

Процесс поиска пути в L-графе, соответствующем $LL(1)$-грамматике, относительно прост. Начиная с начальной вершины L-графа, ищется дуга с пометкой, соответствующей первому символу искомой цепочки α, то есть π, где $\omega(\pi) = \alpha$. После её нахождения поиск продолжается от $end(\pi)$ для следующего символа и так далее, пока последовательно не будут найдены все символы, не будет достигнута конечная вершина L-графа и стек, хранящий скобки, не станет пустым. Если такой путь не находится, это означает, что цепочка не соответствует грамматике.

В L-графе, построенном алгоритмом $A_{G\to L}$, степень исхода вершины больше 1 только в двух случаях.

1) Первый случай — когда нетерминал имеет несколько продукций, то есть соответствующий ему подграф имеет несколько ветвей, тогда начальная вершина этого подграфа имеет несколько исходящих дуг, соответствующих каждой продукции.
2) Второй случай — когда нетерминал многократно встречается в правых частях продукций грамматики, то есть имеется несколько вершин, соединяющих соответствующий подграф, тогда конечная вершина этого подграфа имеет несколько исходящих дуг, на которых закрывающие скобки соответствуют открывающим скобкам при входе в подграф.

В первом случае, если среди исходящих дуг вершины есть дуга с пометкой, соответствующим текущему искомому символу, можно напрямую выбрать этот путь. Если же ни одна исходящая дуга не соответствует текущему символу, но есть дуга с открывающей скобкой, то есть есть возможность выбора подграфа, проверяется, входит ли искомый символ в $direct(\pi)$, при условии что $beg(\pi)$ является вершиной с исходящей степенью больше единицы. Если входит, выбирается соответствующая дуга. Из свойств $LL(1)$ легко следует, что выбор каждой ветви единственен.

Во втором случае, чтобы обеспечить соответствие скобок, выбор пути является детерминированным.

L-граф, соответствующий $LL(1)$-грамматике, является детерминированным, правильный путь единственен, и его поиск не требует возвратов.

Пример 5: Рассмотрим грамматику и L-граф из примера 3 и продемонстрируем изменения стеке (рис. 6).

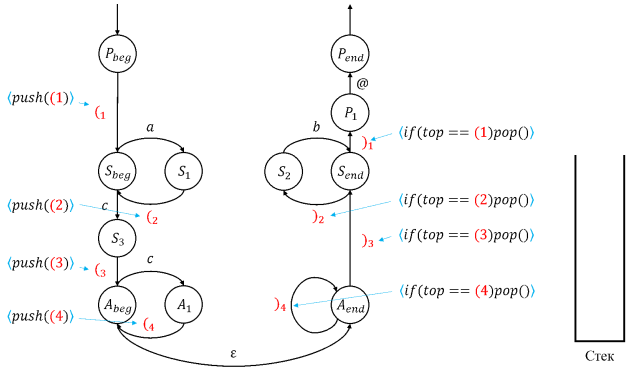

Рис. 6. Синтаксический анализ с помощью L-графа

Теперь для цепочки $acb@$ с помощью L-графа мы найдем путь:

$$T = P_{beg}\pi_0 S_{beg}\pi_1 S_1\pi_2 S_{beg}\pi_3 S_3\pi_4 A_{beg}\pi_5 A_1\pi_6 A_{beg}\pi_7 A_{end}\pi_8 A_{end}\pi_9 S_{end}\pi_{10} S_2\pi_{11} S_{end}\pi_{12} P_1\pi_{13} P_{end},$$

и процесс работы со стеком выглядит как показано на рис. 7.

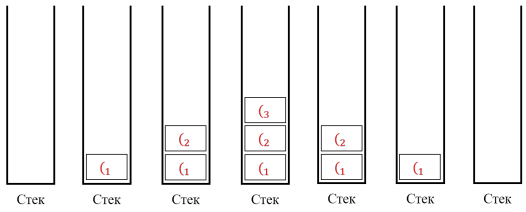

Рис. 7. Преобразование в стеке

Для бесконтекстной грамматики из примера 3 был построен магазинный автомат методом имитации левого вывода [7], как показано на рис. 8. Данный метод был впервые предложен в [8].

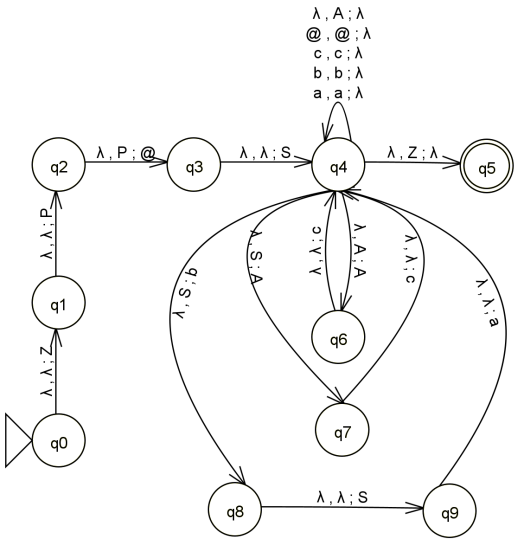

Рис. 8. Магазинный автомат для грамматики из примера 3

Вот последовательность изменений в стеке при анализе цепочки $acb@$ с использованием данного автомата:

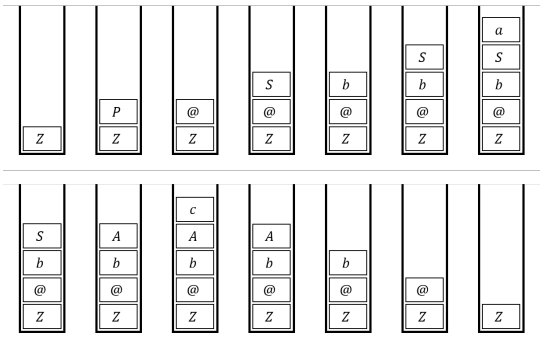

Рис. 9. Преобразование в стеке при работе с магазинным автоматом

Можно заметить, что по сравнению с использованием магазинного автомата, анализ грамматики с помощью L-

графа происходит проще и нагляднее, тогда как применение магазинного автомата менее интуитивно и содержит избыточные операции.

Для не $LL(1)$-грамматик также можно использовать соответствующий L-граф для синтаксического анализа — для этого можно применить алгоритм поиска в глубину для нахождения пути.

БЛАГОДАРНОСТИ

Результаты настоящего исследования были получены при финансовой поддержке Правительства г. Шэньчжэня и Университета МГУ-ППИ в Шэньчжэне.

Список литературы

[1] Вылиток А. А., Сутырин П. Г. Характеризация формальных языков графами // Сборник тезисов научной конференции «Тихоновские чтения». — Москва, МГУ имени М. В. Ломоносова, факультет ВМК, 2010. — С. 82–83.

[2] Станевичене Л. И. О некоторых определениях класса КС-языков видеоданных // Программирование. — 1999. — № 5. — С. 15–25. — УДК 519.682.1.

[3] Станевичене Л. И. К теории бесконтекстных языков. — М., 2000. — Деп. в ВИНИТИ РАН 29.05.2000, №1546-В00.

[4] Вылиток А. А. Кондратьев Г. Д. Синтаксический анализ с помощью L-графов // Ломоносовские чтения: Научная конференция. — Москва, факультет ВМК МГУ имени М.В.Ломоносова, 2017. — С. 124–125. — Тезисы докладов.

[5] Хопкрофт Джон Э., Мотвани Раджив, Ульман Джеффри Д. Введение в теорию автоматов, языков и вычислений / Под ред. А. Б. Ставровский. — 2 изд. — М. : Издательский дом «Вильямс», 2008. — 528 с. — Пер. с англ.: Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. 2nd ed.

[6] Вылиток А. А., В. Генералова Т. Условия регулярности L-графов без псевдоциклических путей // Ломоносовские чтения-2020. Секция Вычислительной математики и кибернетики. — Москва, МГУ имени М. В. Ломоносова, факультет ВМК, 2020. — С. 48–49.

[7] Sipser Michael. Introduction to the Theory of Computation. — 2nd edition. — Boston, MA : Thomson Course Technology, 2005. — ISBN: 0-534-95097-3.

[8] Context-free grammars and pushdown storage : QPR (Quarterly Progress Report) : 65 / Massachusetts Institute of Technology ; Executor: N. Chomsky. — Cambridge, MA : 1962. — P. 187–194. — Research Laboratory of Electronics.

ЛИ Цзямянь,
аспирант Университета МГУ – ППИ в Шэньчжэне
(http://szmsubit.ru/),
email: lijiamian0804@live.com.

Algorithms for transforming context-free grammars into L-graphs

Li Jiamian

***Abstract*—Formal language theory holds a significant place in computer science, where context-free grammars, due to their powerful expressiveness, are widely applied in areas such as programming language design, natural language processing, and compiler construction. Traditionally, pushdown automata have been the primary tool for recognizing and processing context-free languages. However, the fragmentation of state transition descriptions and stack operations into separate commands complicates the analysis of computations, hindering practical application. L-graphs are conceived as a more advanced and compact tool for representing and analyzing formal languages, thanks to their intuitive graphical structure and the clear semantics of graph paths that reflect computations.**

This paper presents a systematic transformation algorithm capable of converting an arbitrary context-free grammar into an equivalent context-free L-graph. This method performs an intuitive, one-to-one conversion of grammatical structures into graphical ones. Unlike pushdown automata, L-graphs do not involve explicit stack operations, instead using bracket labels to implement recursive structures, which makes their usage more intuitive. Furthermore, the study demonstrates the feasibility of efficient syntactic parsing using L-graphs, highlighting their practical value. While retaining the expressive power of context-free grammars, L-graphs are more amenable to optimization (such as eliminating redundant vertices and arcs, reducing cyclic complexity, etc.), enhancing their effectiveness for both theoretical research and practical applications.

***Keywords*—formal languages, context-free grammar, L-graph, syntactic analysis.**

References

[1] Vylitok A. A., Sutyrin P. G. Characterization of formal languages by graphs // Collection of abstracts of the scientific conference “Tikhonov Readings”. — Moscow, Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, 2010. P. 82–83 (in Russian).

[2] Stanevichene L. I. On some definitions of the class of context-free languages for video data // Programmirovanie. 1999. No. 5. P. 15–25 (in Russian).

[3] Stanevichene L. I. On the theory of context-free languages. — Moscow, 2000. — Deposited in VINITI RAS 29.05.2000, No. 1546-B00 (in Russian).

[4] Vylitok A. A., Kondratiev G. D. Syntactic analysis using L-graphs // Lomonosov Readings: Scientific Conference. — Moscow, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, 2017. P. 124–125 (in Russian).

[5] Hopcroft John E., Motwani Rajeev, Ullman Jeffrey D. Introduction to Automata Theory, Languages, and Computation / Ed. by A. B. Stavrovsky. — 2nd ed. — Moscow : Williams Publishing House, 2008. — 528 p. — Translation from English: Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. 2nd ed.

[6] Vylitok A. A., Generalova V. G. Regularity conditions for L-graphs without pseudocyclic paths // Lomonosov Readings-2020. Section of Computational Mathematics and Cybernetics. — Moscow, Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, 2020. P. 48–49 (in Russian).

[7] Sipser Michael. Introduction to the Theory of Computation. — 2nd edition. — Boston, MA : Thomson Course Technology, 2005. — ISBN: 0-534-95097-3.

[8] Context-free grammars and pushdown storage : QPR (Quarterly Progress Report) : 65 / Massachusetts Institute of Technology ; Executor: N. Chomsky. — Cambridge, MA : 1962. — P. 187–194. — Research Laboratory of Electronics.

LI Jiamian,
Post-graduate student of Shenzhen MSU – BIT University,
China (http://szmsubit.ru/),
email: lijiamian0804@live.com.