Корректность алгоритма разбора LC (k)-грамматик

Шпилевой Денис

17 мая 2024 г.

1 Основные определения

Введем основные определения, которые мы будем использовать на протяжении всего доклада.

Одна из главных задач синтаксического анализа — уметь задавать множества слов (языки) простыми математическими конструкциями и минимизировать асимптотику, за которую можно определить, принадлежит ли слово данному языку. Введем базовые понятия и обозначения.

- Алфавитом Σ будем называть любое множество символов.
- Слово (цепочка) в алфавите Σ:
    1. ε — цепочка в Σ (пустое слово);
    2. Если x — цепочка в Σ и $a \in \Sigma$, то xa — цепочка в Σ;
    3. y — цепочка в $\Sigma \Leftrightarrow$ она является таковой в силу (1) или (2).
- Язык в алфавите Σ — множество слов в данном алфавите.
- Разбор слова - процесс нахождения синтаксической структуры данного слова в языке.

Также введем наиболее распространенную математическую конструкцию, которой можно задавать почти любые языки.

Определение 1 ***Грамматика*** *— четвёрка* $G = (N, \Sigma, P, S)$*, где*

- N *— конечное множество нетерминальных символов (нетерминалов)*
- Σ *— непересекающееся с* N *множество терминальных символов (терминалов)*
- P *— конечное подмножество множества* $(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$ *Представляет набор правил, посредством применения которых производится разбор слова в данной грамматике.*
- S *— стартовый нетерминал.*

В иерархии Хомского грамматики разделяются на 4 вложенных друг в друга класса: право-сторонние (регулярные), контекстно-свободные, контекстно-зависимые и порождающие. Их можно моделировать на: конечных автоматах, магазинных автоматах, машинах Тьюринга с ограниченной доп. памятью, и на машинах Тьюринга соответственно. Забегая вперёд, мы будем работать только с контекстно-свободными грамматиками, конкретно с подклассом КС-грамматик - LC-грамматиками.

Определение 2 *Грамматика называется* ***контекстно-свободной****, если каждое правило имеет вид* $A \to \alpha$*, где* $A \in N$*,* $\alpha \in (N \cup \Sigma)^*$.

КС-грамматики задают КС-языки, которые представляют из себя широкий класс формальных языков. Они достаточно удобны для того, чтобы задавать языки программирования (один из примеров, язык GO).

Определение 3 *Для КС-грамматики* $G = (N, \Sigma, P, S)$ *определим функцию:* $\mathrm{FIRST}_k(\alpha) = \{x | \alpha \Rightarrow_l^* x\beta \text{ и } |x| = k, \text{ или } \alpha \Rightarrow^* x \text{ и } |x| < k\}$.

2 LC (k)-грамматики

Теперь мы можем приступить к самой теме работы. Существует класс LC-грамматик, которые аналогично LL-грамматикам читают введенное слово слева направо, но проводят разбор по левому участку (left corner).
Что имеется в виду:

- Аналогично с левым разбором LL-грамматик будет раскрываться самый левый нетерминал;
- Аналогично с левым разбором LL-грамматик слово будет читаться слева направо;
- Но, в отличие от левого разбора LL-грамматик, при известном левом выводе $S \Rightarrow_{lc}^* \omega A \delta$ мы можем определить, что к A нужно применить правило $A \to X_1 X_2 \dots X_n$, когда будет известна часть входного слова, выведенного из X_1, и следующие после него k символов, либо, если X_1 — терминал, $k - 1$ символ.

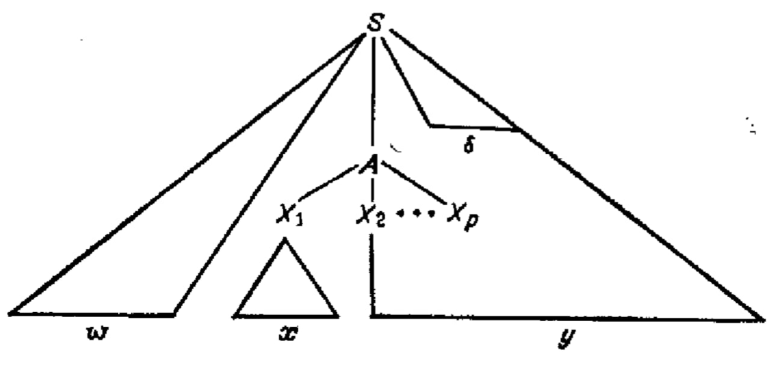

Рис. 1: Разбор по левому участку.

В контексте деревьев вывода при разборе по левому участку можно сказать, что мы определим, что к A нужно применить правило $A \to X_1 X_2 \ldots X_n$, если мы прочитали ωx, и знаем $\mathrm{FIRST}_k(y)$ (либо $\mathrm{FIRST}_{k-1}(y)$, если X_1 — терминал).

Чтобы четко понимать, с какими объектами мы работаем, введем определения LC-грамматики и вывода в LC-грамматике.

Определение 4 *Пусть G - КС-грамматика. Будем писать $S \Rightarrow^*_{lc} \omega A \delta$, если $S \Rightarrow^*_{lc} \omega A \delta$ и нетерминал A не является левым участком того правила, благодаря которому он в ходе этого вывода оказался в левовыводимой цепочке $\omega A \delta$.*
КС-грамматика $G = (N, \Sigma, P, S)$ называется $\mathrm{LC}(k)$-грамматикой, если она удовлетворяет таким условиям:
Допустим, что $S \Rightarrow^*_{lc} \omega A \delta$. Тогда для каждой цепочки $u \in \Sigma^{*k}$ и вывода $A \Rightarrow^* B\gamma$ существует не более одного правила $B \to \alpha$, что

(1) (а) если $\alpha = C\beta$, где $C \in \mathbb{N}$, то $u \in \mathrm{FIRST}_k(\beta\gamma\delta)$,

(б) если, кроме того, $C = A$, то $u \notin \mathrm{FIRST}_k(\delta)$;

(2) (а) если α начинается терминалом, то $u \in \mathrm{FIRST}_k(\alpha\gamma\delta)$.

3 Алгоритм разбора по левому участку для $\mathrm{LC}(1)$-грамматик

Опишем для $\mathrm{LC}(1)$-грамматик алгоритм построения анализатора, управляющего разбором по левому участку. Для этого введем основные понятия.

- Анализатор по левому участку α, такой что

  $$\tau(\mathcal{A}) = \{(x, \pi) \,|x \in L(G) \text{ и } \pi \text{ — разбор по левому участку цепочки } x\}.$$

  В этом случае будем называть алгоритм разбора $\mathcal{A}$ грамматики G корректным.

- Анализатор управляет алгоритмом разбора, совершая тактовые операции над конфигурациями — тройками вида: (необработанная часть слова, магазин, разбор). Такты мы опишем в терминах отношения достижимости $\vdash$ — наименьшего рефлексивного транзитивного отношения над множеством конфигураций.

- Множество магазинных символов — $\Gamma = N \cup \Sigma \cup (N \times N) \cup \{\$\}$, стартовое состояние — S. В случае, если верхний символ магазина — пара нетерминалов вида $[A, B]$, то говорится, что A — цель, а B — левый участок, который только что распознался алгоритмом.

Для удобства построим T — таблицу, управляющую разбором по левому участку:

$$T : \Gamma \times (\Sigma \cup \{\varepsilon\}) \to (\Gamma^* \times (P \cup \{\varepsilon\})) \cup \{\textbf{выброс, допуск, ошибка}\}.$$

Свяжем значения таблицы разбора с конфигурациями. Достижимость на множестве конфигураций задаётся так:

- Если $T(X, a) = (\beta, i)$, где $X \in N \cup (N \times N)$, то будем писать $(a\omega, X\alpha\$, \pi) \vdash (a\omega, \beta\alpha\$, \pi)$;
- Если $T(a, a) =$ **выброс**, то будем писать $(a\omega, a\alpha\$, \pi) \vdash (a\omega, \alpha\$, \pi)$;
- Будем говорить, что π — разбор слова ω, если $(\omega, S, \varepsilon) \vdash (\varepsilon, \$, \pi)$, $(\omega, S\$, \varepsilon)$ — стартовая, а $(\varepsilon, \$, \pi)$ — завершающая конфигурация;
- В случае, если завершающая конфигурация недостижима из стартовой, алгоритм при первом достижении ошибочной конфигурации выводит ошибку, что означает, что слово не лежит в языке.

А л г о р и т м 1 .
Представим алгоритм построения таблицы управляющей разбором по левому участку T для LC (1)-грамматик.

(1) Допустим, что $B \rightarrow \alpha$ - правило из P с номером i.

   (а) Если $\alpha = C\beta$, то $T([A, C], a) = (\beta[A, B], i)\, \forall A \in N$ и $\forall a \in$ FIRST$_1$ $(\beta\gamma\delta)$ $\mid S \Rightarrow^*_{lc} \omega A\delta$ и $A \Rightarrow^* B\gamma$. Здесь анализатор распознает левые участки снизу вверх. Заметим, что A — это либо S, либо левый участок правила, так что в какой-то момент разбора A станет целью.

   (б) Если α не начинается нетерминалом, то

$$T(A, a) = (\alpha[A, B], i)\, \forall A \in N \text{ и } \forall a \in \mathrm{FIRST}_1\, (\beta\gamma\delta) \mid S \Rightarrow^*_{lc} \omega A\delta \text{ и } A \Rightarrow^* B\gamma;$$

(2) $T([A, A], a) = (\varepsilon, \varepsilon)\, \forall A \in N$ и $\forall a \in$ FIRST$_1$ (δ) $\mid S \Rightarrow^*_{lc} \omega A\delta$;

(3) $T(a, a) =$ **выброс** $\forall a \in \Sigma$;

(4) $T(\$, \varepsilon) =$ **допуск**;

(5) В остальных случаях $T(X, a) =$ **ошибка**.

Докажем корректность алгоритма построения таблицы управляющей разбором по левому участку T для LC (1)-грамматик.

Теорема 1 *Алгоритм 1 строит корректную управляющую таблицу для произвольной* LC (1)*-грамматики.*

Доказательство: Для начала положим следующие замечания. Слово ω принадлежит LC-языку задаваемому КС-грамматикой $G = (N, \Sigma, P, S)$ тогда и только тогда, когда у этого слова существует вывод по левому участку в G, то-есть: $\omega \in L(G) \Leftrightarrow S \Rightarrow^*_{lc} \omega$. Хотим показать, что если слово лежит в языке, то оно разбирается алгоритмом разбора LC (1)-грамматик. Хотим доказывать следующий факт: $(\omega, S\$, \varepsilon) \vdash (\varepsilon, \$, \pi) \Leftrightarrow S\ {}_{\pi} \Rightarrow^*_{lc} \omega$.

Для этого докажем более общий факт по индукции, представленный в лемме.

Лемма 1 $(xy, \alpha\gamma, \pi_1) \vdash^* (y, \beta\gamma, \pi_1\pi_2) \Leftrightarrow \alpha_{\pi_2} \Rightarrow^*_{lc} x\beta$.

Доказательство: В прямую сторону.

Вначале заметим, что если G — LC(k)-грамматика, то на шагах алгоритма (1) и (2) в каждой клеточке таблицы будет не более одного значения. После чего, во время шагов (3), (4) и (5) таблица будет заполнена полностью.
Обусловимся тем, что некоторые магазинные символы, не являющиеся нетерминалами, будут выводить то же, что они порождают в соответствии с алгоритмом. Это можно задать гомоморфизмом на множестве магазинных символов, сохраняющим свойства языка.

n = 0: Знаем, что: $(xy, \alpha\gamma\$, \pi_1) \vdash^0 (y, \beta\gamma, \pi_1\pi_2)$.
В этом случае не произошла ни одна тактовая операция, не был считан ни один символ ($x = \varepsilon$) и не было раскрыто ни одно правило ($\pi_2 = \varepsilon$). Тогда: $(y, \alpha\gamma\$, \pi_1) \vdash^0 (y, \beta\gamma\$, \pi_1)$. Следовательно $\beta = \alpha$, и в силу рефлексивности выводимости очевидно верно, что $\alpha \Rightarrow^0 x\beta = \alpha$, чтд.

n = 1: Знаем, что: $(xy, \alpha\gamma\$, \pi_1) \vdash^1 (y, \beta\gamma\$, \pi_1\pi_2)$. Рассмотрим все возможные случаи, которые могут произойти при одном шаге разбора, не приводящие к ошибке.
1) Считывание. В этом случае $x = a$, и $\alpha = a\alpha' : (ay, a\alpha'\gamma\$, \pi_1) \vdash^1 (y, \beta\gamma\$, \pi_1\pi_2)$. Тогда $\beta = \alpha', \pi_2 = \varepsilon$: $(ay, a\alpha'\gamma\$, \pi_1) \vdash^1 (y, \alpha'\gamma\$, \pi_1)$. Тогда $\alpha' = \beta$. Опять же, и в силу рефлексивности выводимости $\alpha = a\alpha' \Rightarrow^0 a\alpha' = x\beta$.
2) Раскрытие правила. В этом случае $x = \varepsilon$: $(y, \alpha\gamma\$, \pi_1) \vdash^1 (y, \beta\gamma\$, \pi_1\pi_2)$. Рассмотрим случаи:

- $\pi_2 = \varepsilon$. При раскрытии правила это возможно только в том случае, если алгоритм совершил переход по пункту (2), то-есть $\alpha = [A, A]\,\alpha' : (y, [A, A]\,\alpha'\gamma\$, \pi_1) \vdash^1 (y, \alpha'\gamma\$, \pi_1)$, где $[A, A]$ выводит ε. Тогда $\beta = \alpha'$, и мы получаем $\alpha \Rightarrow^1_{lc} \alpha' = \beta = x\beta$.

- $\pi_2 \neq \varepsilon$. При раскрытии правила это возможно только в том случае, если алгоритм совершил переход по пункту $(1) : (y, \alpha\gamma\$, \pi_1) \vdash^1 (y, \beta\gamma\$, \pi_1 i)$. Рассмотрим пункт (a) — случай, когда первый символ правой части правила с номером i является нетерминалом. Тогда i-е правило имеет вид $B \to C\delta$ и $\alpha = [A, C]\,\alpha'$.
  Тогда $(y, [A, C]\,\alpha'\gamma\$, \pi_1) \vdash^1 (y, \delta\,[A, B]\,\alpha'\gamma\$, \pi_1 i)$, следовательно $\beta = \delta\,[A, B]\,\alpha'$, и мы получаем: $\alpha = [A, C] \Rightarrow^1 \delta\,[A, B]\,\alpha' = \beta = x\beta$,
  Аналогично рассматриваем пункт (б).

Предположение индукции. Докажем корректность перехода от n тактов к $n+1$ тактам работы анализатора.

Знаем: $(x''y'', \alpha''\gamma''\$, \pi_1'') \vdash^n (y'', \beta''\gamma'', \pi_1''\pi_2'') \Rightarrow \alpha''_{\pi_2''} \Rightarrow^*_{lc} x''\beta''$.
Хотим: $(xy, \alpha\gamma\$, \pi_1) \vdash^{n+1} (y, \beta\gamma, \pi_1\pi_2) \Rightarrow \alpha_{\pi_2} \Rightarrow^*_{lc} x\beta$.

Существует конфигурация $(uy, \delta\gamma, \pi_1\pi')$, $u \in \Sigma \cup \varepsilon$, из которой мы за один шаг алгоритма пришли в конфигурацию $(y, \beta\gamma, \pi_1\pi_2)$. Рассмотрим случаи, которые могли произойти на последнем шагу работы алгоритма.

- Последний шаг — считывание. Тогда $u = a \in \Sigma, \delta = u\beta$, $\pi' = \pi_2$. Возьмём $x = vu$ и применим предположение: $(vuy, \alpha\gamma\$, \pi_1) \vdash^n (uy, u\beta\gamma, \pi_1\pi_2) \ \vdash^1 (y, \beta\gamma, \pi_1\pi_2)$.
  По предположению мы знаем, что $\alpha \Rightarrow^*_{lc} vu\beta = x\beta$.
  Но по $n = 1$ пункт 1), прибегнув к необходимой для понимания тавтологии, мы так же знаем, что $u\beta$, взятое из слота магазина предпоследней конфигурации, выводит $u\beta$, где u взято из слота входной ленты предпоследней конфигурации а β из слота мага-

зина последней конфигурации, т.е. $u\beta \Rightarrow^0 u\beta$. Скомбинируем данные утверждения и получим то, что хотели доказать.

- Последний шаг — раскрытие правила. Докажем формально случай, при котором алгоритм сработал по пункту (2), остальные — (1) (а) и (1) (б) будут доказыватся аналогично. $u = \varepsilon, \delta = [A, A]\,\beta$, где $[A, A]$ выводит $\varepsilon, \pi' = \pi_2$. Применим предположение: $(xy, \alpha\gamma\$, \pi_1) \vdash^n (y, [A, A]\,\beta\gamma, \pi_1\pi_2) \ \vdash^1 (y, \beta\gamma, \pi_1\pi_2)$. Комбинируем предположение с $n = 1$ пункт 2).

В обратную сторону доказательство проводится аналгочнио, индукцией по количеству шагов вывода. ■
Доказательство самой теоремы. Возьмём $\alpha = S, \beta = \varepsilon, \gamma = \varepsilon, x = \omega, y = \varepsilon, \pi_1 = \varepsilon, \pi_2 = \pi$. Получим $(\omega, S\$, \varepsilon) \vdash (\varepsilon, \$, \pi) \Leftrightarrow S \ _\pi\!\Rightarrow^*_{lc} \omega$, что мы и хотели доказать.

В дальнейших исследованиях будет сформулирована и доказана аналогичная теорема для алгоритма разбора LC (k)-грамматик.

Список литературы

[1] Alfred V. Aho and Jeffrey D. Ullman. *Theory of Parsing, Translation, and Compiling, Volume 1: Parsing.* Prentice-Hall, 1972.

[2] 11th Annual Symposium on Switching and Automata Theory (SWAT 1970). Deterministic Left Corner Parsing, pages 139-152.