Метод рекурсивного спуска

Описание и применимость метода. Сравнение восходящих и нисходящих распознавателей

Асанов Дамир, Басалаев Даниил

14 декабря 2024 г.

Метод рекурсивного спуска

Метод рекурсивного спуска реализует разбор цепочки сверху вниз следующим образом: для каждого нетерминального символа грамматики создается процедура, носящая его имя. Задача этой процедуры – начиная с указанного места исходной цепочки найти подцепочку, которая выводится из этого нетерминала.

Если такую подцепочку найти не удается, то процедура завершает свою работу вызовом процедуры обработки ошибок, которая выдает сообщение о том, что цепочка не принадлежит языку грамматики, и останавливает разбор. Если подцепочку удалось найти, то работа процедуры считается нормально завершенной, и осуществляется возврат в точку вызова. Тело каждой такой процедуры составляется непосредственно по правилам вывода соответствующего нетерминала, при этом терминалы распознаются самой процедурой, а нетерминалам соответствуют вызовы процедур, носящих их имена

Пример грамматики с использованием метода РС

Рассмотрим грамматику $G = \langle N, T, R, S \rangle$, где:

$N = \{E, P, M\}$

$T = \{b, +, -, *, /, (,), \$\}$

R:

- $S \to E$
- $E \to P + E \mid P - E \mid P$
- $P \to F * P \mid F/P \mid F$
- $F \to M \mid (E)$
- $M \to b$

Тогда цепочка $7 + 2 * (5 - 3)$ будет обработана как:

- $E(7 + 2 * (5 - 3) - 4) \Rightarrow^* P(7) + P(2 * (5 - 3))$
- $P(7) \Rightarrow^+ M(7) \Rightarrow 7$
- $P(2 * (5 - 3)) \Rightarrow^+ F(2) * F((5 - 3))$
- $F(2) \Rightarrow M(2) \Rightarrow 2$
- $F((5 - 3)) \Rightarrow E(5 - 3) \Rightarrow^+ M(5) - M(3) \Rightarrow 5 - 3$

$LL(k)$-грамматика

Дадим теперь формальное определение $LL(k)$-грамматики.

Пусть $G = \langle \Sigma, N, S, P \rangle$ — КС-грамматика. Рассмотрим два произвольных левосторонних вывода слова w в этой грамматике:

$$S \Rightarrow^* pA\beta \Rightarrow p\alpha\beta \Rightarrow^* py\eta$$

$$S \Rightarrow^* pA\beta \Rightarrow p\alpha'\beta \Rightarrow^* py\xi$$

где p и y — цепочки из терминалов, уже разобранная часть слова w, A — нетерминал грамматики, в которой есть правила $A \to \alpha$ и $A \to \alpha'$, причём $\alpha, \alpha', \beta, \eta, \xi$ — последовательности из терминалов и нетерминалов.

Если из выполнения условий, что $|y| = k$ или $|y| < k$, $\eta = \xi = \varepsilon$, следует равенство $\alpha = \alpha'$, то G называется **LL(k)-грамматикой**.

$LL(1)$-грамматика является частным случаем. Её определение почти такое же, только вместо строки y один символ $c \in \Sigma \cup \{\varepsilon\}$.

Неформально это означает, что, посмотрев на очередной символ после уже выведенной части слова, можно однозначно определить, какое правило из грамматики выбрать.

Множества *FIRST* и *FOLLOW* [1/2]

Ключевую роль в построении парсеров для $LL(1)$-грамматик играют множества *FIRST* и *FOLLOW*. Пусть c — символ из алфавита Σ, α, β — строки из нетерминалов и терминалов (возможно пустые), S, A — нетерминалы грамматики (начальный и произвольный соответственно), \$ — символ окончания слова. Тогда определим

- $FIRST(A) = \{c | A \Rightarrow *c\beta\} \cup \{\epsilon \text{ if } A \Rightarrow \epsilon\}$
- $FOLLOW(A) = \{c | S \Rightarrow *\alpha A c\beta\} \cup \{\$ \text{ if } S \Rightarrow \alpha A\}$

Другими словами, $FIRST(A)$ — все символы (терминалы), с которых могут начинаться всевозможные выводы из A, а $FOLLOW(A)$ — всевозможные символы, которые встречаются после нетерминала A во всех небесполезных правилах грамматики.

Множества *FIRST* и *FOLLOW* [2/2]

Множества *FIRST* и *FOLLOW* могут отличаться даже для одной грамматики, если она задана разными правилами. Рассмотрим пример двух различных грамматик для языка правильных скобочных последовательностей:

- $A \to (A)A \mid \varepsilon$
- $B \to BB \mid (B) \mid \varepsilon$

Правило	FIRST	FOLLOW
A	$\{(, \varepsilon\}$	$\{), \$\}$
B	$\{(, \varepsilon\}$	$\{(,), \$\}$

Теорема о связи $LL(1)$-грамматики с множества $FIRST$ и $FOLLOW$ [1/2]

Грамматика $G = \langle \Sigma, N, S, P \rangle$ является $LL(1)$-грамматикой, если выполняются следующие условия:

- $A \Rightarrow \alpha, A \Rightarrow \beta, A \in N \Rightarrow FIRST(\alpha) \cap FIRST(\beta) = \emptyset$
- $A \Rightarrow \alpha, A \Rightarrow \beta, A \in N, \epsilon \in FIRST(\alpha) \Rightarrow FOLLOW(A) \cap FIRST(\beta) = \emptyset$

Достаточность: Предположим, что данная грамматика не является $LL(1)$-грамматикой. Это значит, что у какого-то слова w существует два различных левосторонних вывода:

- $S \Rightarrow^* pA\gamma \Rightarrow p\alpha\gamma \Rightarrow^* pc\alpha'\gamma$
- $S \Rightarrow^* pA\gamma \Rightarrow p\beta\gamma \Rightarrow^* pc\beta'\gamma$

Но это противоречит тому, что $FIRST(\alpha) \cap FIRST(\beta) = \emptyset$.

Аналогично проверяется второе условие. Если, например, $\alpha \Rightarrow^* \epsilon$, то $\epsilon \in FIRST(\alpha)$, и $FOLLOW(A) \cap FIRST(\beta) = \emptyset$.

Теорема о связи $LL(1)$-грамматики с множества $FIRST$ и $FOLLOW$ [2/2]

Необходимость: Предположим, что существуют два различных правила $A \to \alpha$ и $A \to \beta$ такие, что $c \in FIRST(A) \cap FIRST(\beta)$. Тогда:

- $S \Rightarrow^* pA\gamma \Rightarrow p\alpha\gamma \Rightarrow^* pc\alpha'\gamma$
- $S \Rightarrow^* pA\gamma \Rightarrow p\beta\gamma \Rightarrow^* pc\beta'\gamma$

Последний переход можно совершить, так как c лежит в пересечении множеств $FIRST$ двух правил вывода. Так как грамматика G является $LL(1)$-грамматикой, то из определения следует, что $\alpha = \beta$. Это противоречит предположению, что α и β — различные правила.
Второе условие проверяется аналогичным образом.

Применимость метода [1/2]

Метод рекурсивного спуска применим к грамматике, если правила вывода грамматики имеют один из следующих видов:

- $A \Rightarrow \alpha$, где $\alpha \in (VT \cap VN)^*$, и это единственное правило для нетерминала A;
- $A \Rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \ldots \mid a_n\alpha_n$, где $a_i \in VT$ $a_i \neq a_j$ $i \neq j$ $\alpha_i \in (VT \cap VN)^*$

Если для нетерминала A существует несколько альтернативных правых частей правил вывода, то они должны начинаться с терминальных символов, причем эти терминальные символы должны быть различными. Изложенные выше ограничения являются достаточными, но не необходимыми.

Применимость метода [2/2]

Если $FIRST(A) \cap FOLLOW(A) \neq \emptyset$, то метод рекурсивного спуска неприменим к данной грамматике, потому что в случае

$$A \to \alpha_1 A \mid \ \ldots \ \mid \alpha_n A \mid \beta_1 \mid \ \ldots \ \mid \beta_m \mid \varepsilon$$

$$B \to \alpha A \beta$$

, где $\mathrm{FIRST}(A) \cap \mathrm{FOLLOW}(A) \neq \emptyset$ из-за вхождения A в правило вывода для B имеем косвенную левую рекурсию.

Принимая во внимание теорему о связи $LL(1)$-грамматики с $FIRST$ и $FOLLOW$, делаем вывод, что Метод рекурсивного спуска применим только для $LL(1)$-грамматик.

Преобразования грамматик для применимости РС

Если обнаружено, что метод РС не применим, можно ли написать эквивалентную грамматику, допускающую анализ методом РС?

В общем случае задача алгоритмически неразрешима!

Можно применить эквивалентные преобразования КС-грамматик, которые способствуют приведению грамматики к требуемому виду, но не гарантируют его достижения.

Наличие леворекурсивных нетерминалов

Если в грамматике есть нетерминалы, правила вывода которых леворекурсивны, т. е. имеют вид:

- $A \Rightarrow A\alpha_1 \mid ... \mid A\alpha_n \mid \beta_1 \mid ... \ \beta_m$, где $\alpha_i \in (VT \cup VN)+$ для $i = 1, 2, ..., n$;
- $\beta_i \in (VT \cap VN)^*$

То, воспользовавшись методами удаления прямой левой рекурсии, например, заменить левую рекурсию правой (см. доклад 1.3. ч.7 Корпусова С., Афанасьев А. об методе устранения левой рекурсии):

- $A \Rightarrow \beta_1 A' \mid ... \mid \beta_m A$
- $A' \Rightarrow \alpha_1 A' \mid ... \mid \alpha_n A' \mid \epsilon$

Будет получена грамматика, эквивалентная данной, так как из нетерминала A по-прежнему выводятся цепочки вида $(\beta_j)(\alpha_i)*$, где i = 1, 2, ..., n; j = 1, 2, ..., m

Наличие нетерминалов, у которого несколько альтернативных правил начинаются одинаковыми терминальными символами

Т.е. имеющие вид:

- $A \Rightarrow a\alpha_1 \mid a\alpha_2 \mid \ldots \mid a\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$
- где $a \in VT; \alpha_i, \beta_j \in (VT \cup VN)^*$
- β_j не начинается с a, где i = 1, 2, ..., n; j = 1, 2, ..., m

то можно преобразовать правила вывода данного нетерминального символа, объединив правила с общими началами в одно правило:

- $A \Rightarrow aA' \mid \beta_1 \mid \ldots \mid \beta_m$
- $A' \Rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$

В результате будет получена грамматика, эквивалентная исходной. (левая факторизация)

Наличие нетерминального символа с несколькими альтернативными правыми частями, и среди них есть альтернативы, начинающиеся нетерминальными символами

Т.е. правила вида:

- $A \Rightarrow B_1\alpha_1 \mid B_2\alpha_1 \mid \dots \mid B_n\alpha_n \mid \alpha_1\beta_1 \mid \dots \mid \alpha_m\beta_m$
- где $B_i \in VN$; $a_j \in VT$ α_i, β_j, $\gamma_{ij} \in (VT \cap VN)^*$

То можно заменить вхождения нетерминальных символов B_i соответствующими им правыми частями правил в надежде, что правило для нетерминального символа A станет удовлетворять требованиям метода рекурсивного спуска (попытка устранения косвенной левой рекурсии):
$A \Rightarrow \gamma_{11}\alpha_1 \mid \dots \mid \gamma_{1k}\alpha_1 \mid \dots \mid \gamma_{n1}\alpha_n \mid \dots \mid \gamma_{np}\alpha_n \mid \alpha_1\beta_1 \mid \dots \mid \alpha_m\beta_m$

Вывод

С учетом достаточности становится понятно, что метод рекурсивного спуска применим к весьма узкому подклассу КС-грамматик. Известны более широкие подклассы КС-грамматик, для которых существуют эффективные распознаватели, обладающие тем же свойством, что и распознаватели на основе рекурсивного спуска: входная цепочка считывается один раз слева направо, процесс разбора полностью детерминирован, и время работы такого алгоритма линейно зависит от длины входной цепочки. К таким грамматикам относятся LL(k)-грамматики, LR(k)-грамматики, грамматики предшествования и некоторые другие

Сравнение восходящих и нисходящих распознавателей КС-языков [1/2]

Нисходящие и восходящие распознаватели КС-языков различаются подходом к анализу и построению дерева разбора. Нисходящие распознаватели начинают анализ с начального символа грамматики и пытаются вывести входную строку, постепенно "спускаясь" по правилам грамматики. Этот подход интуитивно понятен, легко реализуется вручную (например, методом рекурсивного спуска) и подходит для грамматик, где выбор правил однозначен (LL-грамматики). Однако они плохо справляются с грамматиками, содержащими левую рекурсию или неоднозначности, и могут потребовать значительной модификации исходной грамматики для корректной работы

Сравнение восходящих и нисходящих распознавателей КС-языков [2/2]

Восходящие распознаватели, напротив, начинают с терминальных символов входной строки и пытаются "подняться" к начальному символу, объединяя последовательности символов в соответствии с правилами грамматики. Они лучше справляются с леворекурсивными и более сложными грамматиками (LR-грамматики), обеспечивая большую мощность анализа. Однако их реализация сложнее, чем у нисходящих методов, а автоматическая генерация таблиц для восходящего анализа может быть вычислительно затратной.

Главным преимуществом нисходящих распознавателей является их простота, что делает их удобными для ручной реализации и небольших грамматик. Восходящие распознаватели, в свою очередь, обладают большей выразительностью и лучше подходят для более сложных грамматик, но требуют автоматизации и более сложных алгоритмов. Выбор метода зависит от требований к грамматике и системе синтаксического анализа.