Контекстно-свободные языки и грамматики

Содержание

Контекстно-свободные языки и грамматики

Перейдем от рассмотрения регулярных языков к более широкому классу языков, которые называются контекстно-свободными. Они описываются в виде контекстно-свободных грамматик. Эти грамматики играют главную роль в технологии компиляции с начала 1960-х годов; они превратили непростую задачу реализации синтаксических анализаторов, распознающих структуру программы, из неформальной в рутинную, которую можно решить за один вечер.

Позже контекстно-свободные грамматики стали использоваться для описания форматов документов в виде так называемых определений типа документов (document-type definition — DTD), которые применяются в языке XML (extensible markup language) для обмена информацией в Internet.

В контекстно-свободных грамматиках важным является понятие дерева разбора, изображающего структуру, которую грамматика налагает на цепочки языка. Дерево разбора представляет собой выход синтаксического анализатора языка программирования и одновременно общепринятый способ выражения структуры программы.

Контекстно-свободные языки также описываются с помощью автоматов с магазинной памятью (иными словами автоматы с магазинной памятью являются распознавателями в КС-языках).

Примеры контекстно-свободных языков и грамматик

Рассмотрим следующий пример:

- $\epsilon, 0, 1$ являются палиндромами (базис рекурсии)
- Если w - палиндром, то $0w0$ и $1w1$ - также палиндром (шаг рекурсии). Ни одна цепочка не является палиндромом, если не определяется базисом или шагом рекурсии.

Например, такие цепочки $0110, 11011$ являются палиндромами, а $0011, 0101$ - нет.
Воспользуемся леммой о накачке, чтобы показать, что данный язык не является регулярным.
Если бы данный язык был регулярным, то существовала бы константа n, начиная с которой любую цепочку длины больше n можно было бы накачивать. Рассмотрим палиндром $w = 0^n10^n$, представим $w = xyz$, где $xy = 0^n$, эту подцепочку уже можно накачать, в ней y состоит из одного или нескольких нулей, но тогда $0^{n-k}(y^k)^m10^n$ - уже не палиндром.
Данный язык можно задать еще следующей порождающей грамматикой: $P \to \epsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1$. Эта запись по сути повторяет рекурсивное определение. В целом контекстно-свободная грамматика представляет собой формальную запись подобных рекурсивных определений языков. Грамматика состоит из одной или нескольких переменных, которые представляют классы цепочек, или языки. В данном примере нужна только одна переменная, представляющая множество палиндромов.

Устранение бесплодных символов

Определение

Нетерминальный символ является бесплодным, если из него нельзя вывести ни одной цепочки терминальных символов.

В простейшем случае символ является бесплодным, если во всех правилах, где этот символ стоит в левой части, он также встречается и в правой части. Более сложные варианты предполагают зависимости между цепочками бесплодных символом, когда они в любой последовательности вывода порождают друг друга.
Алгоритм устранения бесплодных символов прост: 1. мы начинаем с пустого множества нетерминальных символов $Y = \varnothing$. 2. добавляем в множество Y только такие нетерминалы A, для которых имеем $A \to \beta$, где β состоит из одних терминальных символов. 3. последовательно пополняем множество Y нетерминальными символами B, из которых следует цепочки терминальных символом, а также символы самого множества Y. 4. выполняем п.3 пока множество Y изменяется.

Устранение недостижимых символов

Определение

Символ называется недостижимым, если он не участвует ни в одной цепочке вывода из целевого символа грамматики.

Очевидно, недостижимые символы можно удалить. Алгоритм удаления недостижимых символов строит множество достижимых символов X. Первоначально в такое множество входит только целевой символ S, затем оно пополняется на основе правил грамматики. Все символы, которые не войдут в данное множество, являются недостижимыми и могут быть исключены в новой грамматике.

Пример устранения бесплодных и недостижимых символов

Пусть грамматика задана правилами:

$$S \to aAB|E$$

$$A \to aA|bB$$

$$B \to ACb|b$$

$$C \to A|bA|cC|aE$$

$$E \to cE|aE|Eb|ED|FG$$

$$D \to a|c|Fb$$

$$F \to BC|EC|AC$$

$$G \to Ga|Gb$$

Важно соблюдать последовательность преобразования: сначала удаляются бесплодные символы, потом недостижимые.

Пример устранения бесплодных и недостижимых символов

Удаляем бесплодные символы

1. $Y_1 = \{B, D\}$
2. $Y_2 = \{B, D, A\}$
3. $Y_3 = \{B, D, A, S, C\}$
4. $Y_4 = \{B, D, A, S, C, F\}$
5. $Y_5 = Y_4$

Окончательно в множестве продукций оставляем только те символы, которые включает в себя множество Y_4:

$$S \to aAB$$

$$A \to aA|bB$$

$$B \to ACb|b$$

$$C \to A|bA|cC$$

$$D \to a|c|Fb$$

$$F \to BC|AC$$

Пример устранения бесплодных и недостижимых символов

Удаляем недостижимые символы

Итак, теперь из грамматики удалим недостижимые символы:

$$S \to aAB, A \to aA|bB, B \to ACb|b$$

$$C \to A|bA|cC, D \to a|c|Fb, F \to BC|AC$$

1. $X_1 = \{S\}$
2. $X_2 = \{S, A, B\}$
3. $X_3 = \{S, A, B, C\}$
4. $X_4 = X_3$

Окончательно оставляем в грамматике только правила, включающие в себя достижимые символы:

$$S \to aAB$$

$$A \to aA|bB$$

$$B \to ACb|b$$

$$C \to A|bA|cC$$

Неукорачивающиеся и ϵ-свободные грамматики

Определение

Грамматика, которая задается общим правилом $A \to \beta$, где A - один нетерминальный символ, β - цепочка терминальных и нетерминальных символов, называется контекстно-свободной грамматикой (КС-грамматика).

Определение

КС-грамматика называется неукорачивающей, если она не включает продукции вида $A \to \epsilon$.

Определение

КС-грамматика называется ϵ-свободной, если она неукорачивающая или в ней существует ровно одна продукция вида $S \to \epsilon$, где S - начальный нетерминал, и S не встречается в правой части ни одной из продукций.

Очевидно, что любой шаг вывода в неукорачивающей грамматике не может уменьшить длину выводимой цепочки - отсюда и ее название.

Устранение ϵ-правил

Теорема

По любой КС-грамматике может быть построена эквивалентная ϵ-свободная КС-грамматика. По любой КС-грамматике, порождающей язык, не включающий пустую цепочку, может быть построена эквивалентная неукорачивающая грамматика.

Доказательство.

Для любой пустой продукции $A \to \epsilon$ в грамматике найдем и скопируем все продукции, включающие A в правой части, с выбрасыванием символа A из правых частей этих копий, причем если A входит несколько раз в правую часть продукции, то такое выбрасывание производится во всех возможных сочетаниях. Далее продукция $A \to \epsilon$ выбрасывается из множества правил. Если A - начальный нетерминал, то выбираем новый начальный нетерминал S и к продукциям грамматики добавляем две новые: $S \to \epsilon, S \to A$. □

Пример устранения ϵ правил

Пусть задана грамматика начальным нетерминалом A:

$$A \to \epsilon | AaB | BbB$$

$$B \to Ba | \epsilon$$

Вводим новый символ $S \to \epsilon | A$. Удаляем A в правой части продукций во всех сочетаниях: $A \to AaB | aB | BbB$. Удаляем B в правой части продукций: $B \to Ba | a$, $A \to AaB | Aa | aB | a | BbB | bB | Bb | b$.
Окончательно имеем:

$$S \to \epsilon | A$$

$$A \to AaB | Aa | aB | a | BbB | bB | Bb | b$$

$$B \to Ba | a$$

Устранение циклических символов

Определение

Продукции вида $A \to B$ называются сингулярными.
Если в результате цепного применения правил вида $A \to B$ приходим к выводу, что из $A \Rightarrow A$, то такой вывод называется циклическим.

Любую грамматику можно преобразовать в эквивалентную ей грамматику без сингулярных продукций и циклических выводов. Для этого вместо каждой сингулярной продукции вида $A \to B$ включаем продукции $A \to \beta$, такие, что $B \to \beta$ несингулярна.

Пример устранения циклических символов

Пример 1.

$$S \to BA$$

$$A \to C|ac$$

$$B \to b$$

$$C \to A$$

В данной грамматике есть две сингулярные продукции: $A \to C, C \to A$, из которых фактически следует циклический вывод $A \Rightarrow A$. Заменим продукцию $C \to A$ на несингулярную $A \to \beta$: $C \to ac$. Окончательно получаем:

$$S \to BA$$

$$A \to ac$$

$$B \to b$$

$$C \to ac$$

В результате такого приведения видно, что символ C не выводим (еще говорят недостижим), а значит продукция $C \to ac$ может быть вообще исключена.

Пример устранения циклических символов

Пример 2.

$$S \to S + T|S - T|T$$
$$T \to T * E|T/E|E$$
$$E \to (S)|a|b$$

Здесь мы видим две сингулярности: $S \to T, T \to E$. Поскольку для E имеем несингулярную продукцию $E \to (S)|a|b$, то вместо $T \to E$, можно написать $T \to (S)|a|b$, или в целом $T \to T * E|T/E|(S)|a|b$, которая теперь перестает быть сингулярной, а значит вместо $S \to T$ можно написать $S \to T * E|T/E|(S)|a|b$. Окончательно получаем:

$$S \to S + T|S - T|T * E|T/E|(S)|a|b$$
$$T \to T * E|T/E|(S)|a|b$$
$$E \to (S)|a|b$$

Устранение левой рекурсии

Определение

Нетерминальный символ A называется рекурсивным, если в грамматике существует вывод $A \Rightarrow \alpha A \beta$.

Определение

КС-грамматика называется леворекурсивной, если в ней существует вывод $A \Rightarrow A\beta$.

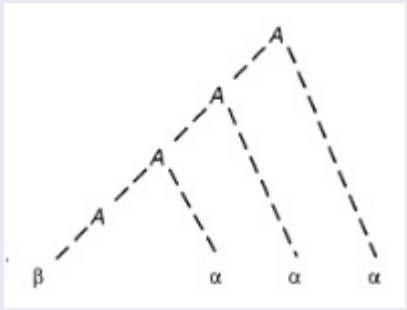
Определение

КС-грамматика называется праворекурсивной, если в ней существует вывод $A \Rightarrow \alpha A$.

Полностью исключить рекурсию в КС-грамматике нельзя, но можно полностью исключить либо левую, либо правую.
Многие алгоритмы синтаксического анализа не могут применяться к грамматикам с левой рекурсией, поэтому возникает необходимость ее устранения.

Устранение левой рекурсии

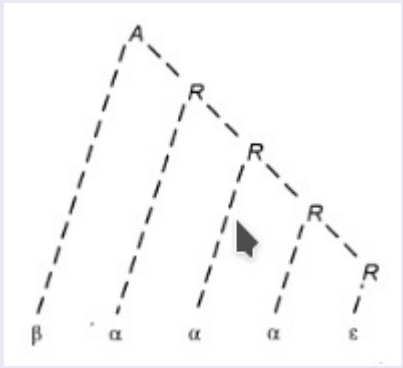
Рассмотрим простейший случай - пусть в грамматике есть продукции $A \to A\alpha|\beta$. Очевидно, что тогда в дереве вывода присутствует следующий фрагмент:



Тогда такие продукции можно заменить нелеворекурсивными:

$$A \to \beta R, R \to \alpha R|\epsilon$$

со следующим деревом вывода:



Устранение левой рекурсии

Пример

Рассмотрим правила грамматики арифметических выражений:

$$E \to E + T | T$$

Используя рассмотренный подход, получаем:

$$E \to TR$$

$$R \to +TR | \epsilon$$

Этот метод обобщается, когда нетермилал имеет несколько альтернатив с левой рекурсией. Более сложный случай непрямой рекурсии требует более сложного алгоритма.

Приведенные грамматики

Определение

Приведенные грамматики - это КС-грамматики, которые не содержат недостижимых, бесплодных символов, циклов и ϵ - правил.

Для того, чтобы преобразоватьпроизвольную грамматику к приведенному виду, необходимо:

1. удалить все бесплодные символы;
2. удалить все недостижимые символы;
3. удалить все ϵ - правила;
4. удалить цепные правила

Все эти преобразования необходимо выполнять строго в указанном порядке.

Нормальная форма Хомского

Определение

КС-грамматика представлена в нормальной форме Хомского, если она неукорачивающая и каждое ее правило имеет одну из следующих форм:

$$A \to BC, A \to a$$

Грамматику $G = \{T, N, S, R\}$ можно привести к форме Хомского $G' = \{T', N', S, R'\}$ следующими преобразованиями:

- В множество продукций R' включаем все продукции вида $A \to BC, A \to a$.
- Для каждой продукции из R вида $A \to X_1X_2 \dots X_k$ включаем в R' множество продукций:

$$A \to X_1' < X_2 \dots X_k >, < X_2 \dots X_k > \to X_2' < X_3 \dots X_k >, \dots,$$
$$< X_{k-1}X_k > \to X_{k-1}'X_k',$$

где $X_i' = X_i$, если X_i - нетерминал, иначе, если $X_i = a$, вводим новый нетерминал X_i', а также правило $X_i' \to a$
$< X_2 \dots X_k >, < X_3 \dots X_k >, \dots < X_{k-1}X_k >$ - новые нетерминалы.

Пример

Пусть дана грамматика:

$$S \to ByA$$

$$A \to BS$$

$$A \to x$$

$$B \to zA$$

Вместо продукции $S \to ByA$ вводим $S \to BD, D \to EA, E \to y$
Вместо продукции $B \to zA$ вводим $B \to ZA, Z \to z$
Окончательно получаем следующую грамматику:

$$S \to BD$$

$$D \to EA$$

$$E \to y$$

$$A \to BS$$

$$A \to x$$

$$B \to ZA$$

$$Z \to z$$

Задание 1

Выбросить все бесполезные продукции:

$$S \to aSbBc$$

$$B \to cD$$

$$S \to BdD$$

$$C \to bCc$$

$$A \to BcS$$

$$C \to cDA$$

$$A \to acb$$

$$D \to cSD$$

$$B \to bE$$

$$E \to ce$$

Задание 2

Построить ϵ-свободную грамматику.

$$S \to AaAb$$

$$S \to BbBa$$

$$A \to \epsilon$$

$$B \to \epsilon$$

Задание 3

По грамматике построить ациклическую грамматику:

$$S \to SS|(S)|\epsilon$$

Задание 4

Удалить левую рекурсию:

$$S \to SS|(S)|AaAb, A \to AbSa|ab$$

Задание 5

Преобразовать к нормальной форме Хомского:

$$S \to SbBc$$

$$B \to cD$$

$$S \to BdD$$

$$B \to b$$

$$D \to DBa$$

$$D \to c$$

Нормальная форма Грейбаха

Определение

КС-грамматика представлена в нормальной форме Грейбаха, если она ϵ-свободная и все ее продукции имеют вид: $A \to a\gamma$, где a - один терминальный символ, γ - произвольная цепочка нетерминалов, возможно пустая.

Существуют простые правила перевода КС-грамматики в нормальную форму Грейбах. Рассмотрим пример. Пусть дана грамматика:

$$S \to BA, A \to BS$$

$$A \to x, B \to zAy$$

Преобразуем ее в нормальную форму Грейбаха. Для этого в двух первых продукциях заменим символ B на цепочку zAy. Далее для последней продукции введем новый нетерминал Y вместо y, а также добавим правило $Y \to y$. Окончательно получаем:

$$S \to zAYA, A \to zAYS$$

$$A \to x, B \to zAY, Y \to y$$

Теорема

Каждую ϵ-свободную КС-грамматику можно преобразовать к нормальной форме Грейбаха.

Задание 6

Для грамматики:

$$S \to ABABABA, A \to Aa, A \to \epsilon, B \to b$$

Найти нормальную форму Хомского, нормальную форму Грейбаха.

Задание 7

Для грамматики:

$$S \to SS, B \to aa, S \to BS, B \to bb, S \to SB$$

Найти нормальную форму Хомского, нормальную форму Грейбаха.

Функции FIRST, FOLLOW

При разработке алгоритмов синтаксического анализа КС-языков часто используются функции FIRST и FOLLOW.

Определение

Для произвольной строки α из терминальных и нетерминальных символов $FIRST(\alpha)$ определяет множество тех терминальных символов, с которых могут начинаться строки, выводимые из α. Если $\alpha \Rightarrow \epsilon$, то $\epsilon \in FIRST(\alpha)$.

$$FIRST(\alpha) = \{a \in T : \alpha \Rightarrow a\beta\} \cup \{\epsilon : \alpha \Rightarrow \epsilon\}$$

Рассмотрим правила вычисления множества $FIRST$:

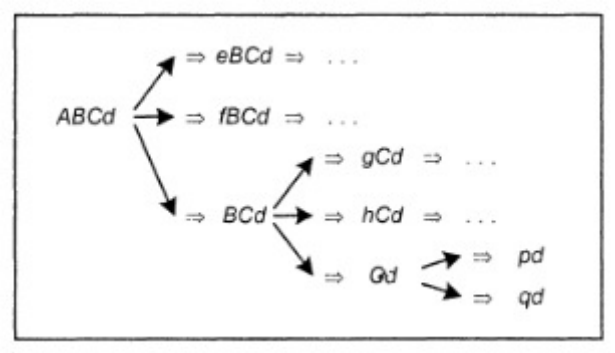
1. Если α начинается с терминала a, то $FIRST(\alpha) = \{a\}$.
2. Если $\alpha \Rightarrow \epsilon$, то $FIRST(\alpha) = FIRST(\alpha) \cup \{\epsilon\}$.
3. Если α начинается с нетерминала A, то $FIRST(\alpha) = FIRST(A) \setminus \{\epsilon\}$. Данное правило применяется рекурсивно.

Пример вычисления функции FIRST

Пусть дана грамматика:

$$S \to ABCd, A \to e|f|\epsilon, B \to g|h|\epsilon, C \to p|q$$

Требуется вычислить $FIRST(S)$. Цепочки вывода демонстрирует рисунок:



Таким образом, $FIRST(S) = \{e, f, g, h, p, q\}$

Функция FOLLOW

Определение

Функция $FOLLOW(A)$ для нетерминала A определяется как множество таких терминальных символов, которые могут следовать за A в какой-либо форме вывода:

$$FOLLOW(A) = \{a \in T : S \Rightarrow \alpha A \beta, a \in FIRST(\beta)\}$$

Рекурсивный алгоритм для нахождения множества $FOLLOW(A)$ состоит в следующем: просматриваются все продукции, в правой части которых встречается символ A, и для каждой такой продукции вида $B \to \alpha A \beta$ все элементы $FIRST(\beta)$, кроме символа ϵ помещаются в $FOLLOW(A)$. Если в этой продукции $\beta \Rightarrow \epsilon$, то все элементы $FOLLOW(B)$ помещаются в $FOLLOW(A)$.

Пример нахождения FIRST, FOLLOW

Найти множества $FIRST, FOLLOW$ для каждого нетерминала грамматики арифметических выражений:

$$S \to E\$$$
$$E \to TE'$$
$$E' \to +TE'|\epsilon$$
$$T \to FT'$$
$$T' \to *FT'|\epsilon$$
$$F \to (E)|i$$

Решение

Поскольу для нетерминала E имеется только одно правило $E \to TE'$, в котором первый символ - нетерминал, то $FIRST(E) = FIRST(T)$, аналогично $FIRST(T) = FIRST(F)$. Наконец, по двум правилам для F находим $FIRST(F) = \{(, i\}$.
Итак: $FIRST(E) = FIRST(T) = FIRST(F) = \{(, i\}$
Также имеем:

$$FIRST(E') = \{+, \epsilon\}$$

$$FIRST(T') = \{*, \epsilon\}$$

При нахождении $FOLLOW(E)$ существуют две продукции, в которых E находится в правой части: $S \to E\$$ и $F \to (E)$. Отсюда:

$$FOLLOW(E) = \{\$,)\}$$

Для нахождения $FOLLOW(E')$ найдем продукции, в которых E' входит в правую часть: $E \to TE', E' \to +TE'$. Поскольку в этих продукциях E' - последний символ, то в $FOLLOW(E')$ следует включить $FOLLOW(E)$ и $FOLLOW(E')$. Значит:

$$FOLLOW(E') = FOLLOW(E) = \{\$,)\}$$

Решение (продолжение)

Для нетерминала T в грамматике есть два правила: $E \to TE', E' \to +TE'$. Следовательно, в $FOLLOW(T)$ включаем множество $FIRST(E') \setminus \{\epsilon\} = \{+\}$. В тоже время $E' \Rightarrow \epsilon$, поэтому к $FOLLOW(T)$ следует добавить множество $FOLLOW(E')$:

$$FOLLOW(T) = \{+,), \$\}$$

Также имеем:

$$FOLLOW(F) = FIRST(T') \setminus \{\epsilon\} \cup FOLLOW(T') = \{*, +,), \$\}$$

Алгоритмические проблемы для КС-языков

Проблема называется алгоритмически разрешимой, если существует разрешающий ее алгоритм. В противном случае проблема называется алгоритмически неразрешимой. К сожалению, многие содержательные проблемы неразрешимы для КС-языков - не существует соответствующего алгоритма в принципе. Среди них:

- проблема эквивалентности - порождают ли две КС-грамматики один и тот же язык;
- проблема пересечения - пересекаются ли множества цепочек, порождаемых двумя КС-грамматиками;
- проблема определения того, порождает ли данная КС-грамматика язык, включающий все терминальные цепочки.

Задание 8

Для грамматики построить множества $FIRST$ и $FOLLOW$ для каждого нетерминала:

$$S \to SbBc$$

$$B \to cD$$

$$S \to BdD$$

$$B \to \epsilon$$

$$D \to DBa$$

$$D \to \epsilon$$