Задание 10

КС-языки: замкнутость и LL-анализ

Ключевые слова [1]:*язык, контекстно-свободный язык, магазинный автомат, грамматика, метод математической индукции.*

1 Лемма о накачке

Одна из целей изучения леммы о накачке для регулярных языков – упрощение понимания леммы о накачке для КС-языков, где она уже жизненно необходима. Сформулируем лемму.

Лемма 1. *Для любого КС-языка L существует константа p, такая что для любого слова w из языка L, такого что $|w| \geqslant p$ существует разбиение слова $w = xuyvz$, причём*

- $|uyv| \leqslant p$
- $|uv| \geqslant 1$
- $\forall i \geqslant 0 : w_i = xu^iyv^iz \in L$

Пример 1. Язык $L = a^nb^nc^n$ не является КС-языком.

Доказательство. Предположим, что $L \in \mathsf{CFL}$, тогда для некоторого числа p выполнена лемма о накачке. Рассмотрим слово $w = a^pb^pc^p$ (заметим, что число p является параметром, а не каким-то конкретным числом). Тогда подслово uyv из разбиения слова w, существующего по лемме о накачке, либо состоит из одинаковых букв (a^l или b^l или c^l) или имеет вид a^lb^r или b^lc^r. Три различные буквы подслово uyv содержать не может, поскольку его длина ограниченна числом p. Но тогда uv – слово, в котором нет одной из трёх букв. Пусть это будет буква c для определённости. Взяв $i = 0$, получаем, что по лемме о накачке $w_0 = a^{n-k}b^{n-m}c^n \in L$, при этом $k + m \geqslant 1$, откуда следует, что слово w_0 не принадлежит языку L. □

[1] минимальный необходимый объем понятий и навыков по этому разделу)

Упражнение 1. Покажите, что КС-языки замкнуты относительно объединения.

Упражнение 2. Покажите, что КС-языки не замкнуты относительно пересечения.

Упражнение 3. Покажите, что КС-языки не замкнуты относительно дополнения.

Задача 1. Постройте КС-грамматику, порождающую язык $\Sigma^* \setminus \{a^n b^n c^n \mid n \geqslant 0\}$.

Задача 2. Верно ли, что язык $\{a^n b^m b^n c^m \mid n, m \geqslant 0\}$ является КС-языком? В случае положительного ответа построить КС-грамматику или МП-автомат для данного языка.

Задача 3. Пусть $A, B \in \mathsf{CFL}$, $R \in \mathsf{REG}$. Верны ли следующие утверждения (для произвольных A, B, R):

1) $A \setminus R \in \mathsf{CFL}$; 2) $R \setminus A \in \mathsf{CFL}$; 3) $A^R \in \mathsf{CFL}$?

Задача 4. Докажите, что язык $\{wtw^R \mid |w| = |t|\} \subseteq \{a, b\}^*$ не является КС-языком.

2 Функция FIRST

При построении (детерминированных) анализаторов по грамматике, нам потребуется определять множество первых k символов слов, выводимых из цепочки $\alpha \in (N \cup T)^*$. Для этого мы будем использовать функцию FIRST_k, которая определена через функцию FIRST_1 или просто FIRST. Таким образом умение вычислять функцию FIRST является ключевым при построении анализаторов.

Формально,

$$\mathrm{FIRST}_k(\alpha) = \{w[1,k] \mid \alpha \Rightarrow w, |w| > k\} \cup \{w \mid \alpha \Rightarrow w, |w| < k\}$$

Если $\alpha \Rightarrow \varepsilon$, то пустое слово лежит в $\mathrm{FIRST}_k(\alpha)$.

Приведём процедуру вычисления функции $\mathrm{FIRST}(\alpha)$.

Идея алгоритма: Если $\alpha = X_1X_2 \dots X_n$ начинается с терминала σ, то первым символом может быть только этот терминал, таким образом, мы сразу получаем ответ σ. Если же α начинается с нетерминала, то $\mathrm{FIRST}(\alpha) = \mathrm{FIRST}(X_1)$, если из нетерминала X_1 не выводится пустое слово, и $\mathrm{FIRST}(\alpha) = \mathrm{FIRST}(X_1) \cup \mathrm{FIRST}(X_2X_3 \dots X_n)$, если $X_1 \Rightarrow \varepsilon$.

Таким образом, мы описали вычисление функции FIRST на множестве терминалов и цепочек, начинающихся с терминалов, осталось описать вычисление функции на множестве нетерминалов, как видно вычисление функции FIRST на множестве сентенциальных форм сводится к вычислению функции на отдельных нетерминалах.

Пусть мы вычисляем функцию $\mathrm{FIRST}(X)$ для нетерминала X. Рассмотрим все правила вида $X \to \beta$. Очевидно, что $\mathrm{FIRST}(\beta)$ является подмножеством $\mathrm{FIRST}(X)$, но просто добавляя множество $\mathrm{FIRST}(\beta)$ к $\mathrm{FIRST}(X)$ мы получим порочный круг, в случае правил вида $X \to Xa$. Как нам избежать порочного круга при вычислении множества $\mathrm{FIRST}(X)$?

Определим множества $F_i(Y),\ Y \in N$. При $i = 0$ для любого нетерминала Y, множество $F_i(Y) = \emptyset$, или $\{\varepsilon\}$, если есть правило $Y \to \varepsilon$. На $i-$ом шаге алгоритма будем вычислять множества $F_i(X)$ следующим образом. В начале шага $F_i(X)$ включает себя множество $F_{i-1}(X)$ Если есть правило $X \to \beta = Y_1Y_2 \dots Y_n$ и Y_1 – терминал или β – пустое слово, то добавим к множеству $F_i(X)$ элемент Y_1 (быть может пустое слово). Если же Y_1 – нетерминал, и при этом пустое слово не лежит в $F_{i-1}(Y_1)$, то добавим к множеству $F_i(X)$ множество $F_{i-1}(Y_1)$ и вычислим множество $F_i(Y_1)$. Если же $\varepsilon \in F_{i-1}(Y_1)$, то добавим к $F_i(X)$ множество $F_i(Y_1) \setminus \{\varepsilon\}$ и повторим описанную операцию для $\beta = Y_2 \dots Y_n$.

Алгоритм останавливается, как только для каждого нетерминала Y, множества F_i и F_{i-1} совпадают.

Алгоритм:

Шаг 0. Для каждого терминала σ положим $F_i(\sigma) = \sigma$ для любого i. Для каждого нетерминала Y, если есть правило $Y \to \varepsilon$, положим $F_0(Y) = \{\varepsilon\}$, иначе положим $F_0(Y) = \emptyset$.

Шаг i. Добавить к множеству $F_i(X)$ множество $F_{i-1}(X)$. Для каждого правила $X \to \beta = Y_1 \dots Y_n$ выполнить:

$j = 1$

Пока $\varepsilon \in F_{i-1}(Y_j)$ и $j \leqslant n$

добавить $F_{i-1}(Y_j) \setminus \{\varepsilon\}$ к $F_i(X)$,

увеличить j.

Если $\varepsilon \in F_{i-1}(Y_1) \cap F_{i-1}(Y_2) \cap \dots \cap F_{i-1}(Y_n)$, добавить ε к $F_i(X)$,

Остановка. $F_i(Y) = F_{i-1}(Y)$ для любого Y из N. Положить $\mathrm{FIRST}(X) = F_i(X)$.

Упражнение 4. Доказать корректность данного алгоритма.

3 Функция FOLLOW

Помимо префикса порождаемого цепочкой β нас будет интересовать также и множество слов, которые могут следовать после слова, выведенного из цепочки β. Запишем сначала формальное определение функции FOLLOW_k.

$$\mathrm{FOLLOW}_k(\beta) = \{w \mid S \Rightarrow \alpha\beta\gamma, w \in \mathrm{FIRST}_k(\gamma)\}.$$

Неформально, в множестве $\mathrm{FOLLOW}_k(\beta)$ содержатся те слова, которые могут следовать за словом, выведенным из β, в цепочке $\alpha\beta\gamma$, выводимой из аксиомы. Длина этих слов ограничена k, что означает, что если $\gamma \Rightarrow w$ и длина слова w меньше k, то w лежит в множестве $\mathrm{FOLLOW}_k(\beta)$, а если же длина слова w больше k, то в множестве $\mathrm{FOLLOW}_k(\beta)$ лежит префикс w длины k.

Аналогично функции FIRST, мы будем обозначать FOLLOW_1 как FOLLOW.

Мы будем часто пользоваться функцией FOLLOW_k в теоретических целях и для обозначения объектов, однако на практике мы будем вычислять функцию FOLLOW только на множестве нетерминалов.

Приведём алгоритм для вычисления функции FOLLOW.

Идея алгоритма: Если в грамматике есть правило $A \to \alpha X \beta$, то за словом, выведенным из нетерминала X следует слово выведенное из β, таким образом множество $\mathrm{FOLLOW}(X)$ включает в себя множество $\mathrm{FIRST}(\beta)$. Если, при этом из цепочки β выводимо пустое слово, то

за словом, выводимым из нетерминала X следует слово из множества FOLLOW(A), поскольку из вывода

$$S \Rightarrow^* \gamma A w \Rightarrow \gamma \alpha X \beta w \Rightarrow \gamma \underbrace{\alpha X}_{A} w$$

следует, что если элемент FIRST(w) лежит в множестве FOLLOW(A), то элемент FIRST(w) лежит так же в множестве FOLLOW(X). Таким образом, по определению функции FOLLOW, если в грамматике есть правило $\mathcal{A} \to \alpha X \beta$ и при этом из цепочки β выводимо пустое слово, то множество FOLLOW(X) включает в себя множество FOLLOW(A). В частности, возможно что $\beta = \varepsilon$, поэтому при наличии в грамматике правила $A \to \alpha X$, справедливо FOLLOW(X) $\supseteq$ FOLLOW(A).

В итоге, мета-алгоритм сводится к следующим шагам:

- Вычислить множества FIRST для грамматики G;
- Для правил $A \to \alpha X \beta$ добавить FIRST(β) $\setminus \{\varepsilon\}$ к FOLLOW(X);
- Для правил $A \to \alpha X \beta$, таких что, $\varepsilon \in$ FIRST(β) добавить FOLLOW(A) к FOLLOW(X).

Упражнение 5. Доказать корректность данного мета-алгоритма. То есть, что все элементы множеств FOLLOW будут найдены и ничего лишнего найдено не будет.

Замечание 1. *По хорошему, возникает проблема с тем, лежит ли пустое слово в* FOLLOW(X). *Эта проблема решается следующим образом: ко всем словам, порождаемым G добавляется маркер конца слова, и если этот маркер оказывается в* FOLLOW(X), *то пустое слово принадлежит* FOLLOW(X). *Для этого по грамматике G строится пополненная грамматика G', которая содержит правило $S' \to S\$$, где $\$$ – маркер конца слова. Все остальные правила грамматики G' взяты из грамматики G. На практике, функция* FOLLOW *используется в анализаторах, на вход которым и так подаётся пополненная грамматика, поэтому решать проблему наличия пустого слова в множестве* FOLLOW(X) *не надо.*

Теперь опишем сам алгоритм. Идея алгоритма схожа с индуктивным вычислением множеств FIRST.

Алгоритм:

Шаг 0. Для каждого нетерминала Y положим $F_0(Y) = \emptyset$. Вычислим значение функции FIRST для грамматики G.

Шаг i. Положить множество $F_i(X)$ равным множеству $F_{i-1}(X)$. Для каждого правила $A \to \alpha X \beta$ добавить $\mathrm{FIRST}(\beta) \setminus \{\varepsilon\}$ к $F_i(X)$; Если $\varepsilon \in \mathrm{FIRST}(\beta)$ добавить $F_{i-1}(A)$ к $F_i(X)$.

Остановка. Как только $F_i(Y) = F_{i-1}(Y)$ для любого Y из N. Положить $\mathrm{FOLLOW}(X) = F_i(X)$.

Задача 5. Грамматика $\mathsf{Expr} = \langle \{E, T, F, E', F'\}, \{\mathrm{id}, +, \times, (,)\}, P, E \rangle$ имеет множество правил P :

$$E \to TE'; \quad E' \to +TE' \mid \varepsilon; \quad T \to FT';$$

$$T' \to \times FT' \mid \varepsilon; \qquad F \to (E) \mid \mathrm{id}.$$

Вычислите функции FIRST и FOLLOW для всех нетерминалов грамматики Expr.