Лаборатория непрерывного математического образования

Введение в программирование: Марковский алгорифм, грамматики и конечные автоматы

Аннотация

Данный курс в той или иной форме рассказывается новичкам, поступающим в ЛНМО; обычно это происходит в летней школе. Курс предназначен для первоначального ознакомления с предметом. Данный документ содержит краткий конспект теоретической части курса.

1 Алгорифм Маркова

1.1 Синтаксис алгорифма Маркова

Определение 1.1. Алфавит — конечное множество символов.

Определение 1.2. Слово — конечная последовательность символов. Слово в алфавите A — конечная последовательность символов из алфавита A.

Определение 1.3. Язык — множество слов.

Определение 1.4. Правило — запись, имеющая такой вид

$$\alpha \quad \to \quad \beta$$

или такой вид:

$$\alpha \quad \to . \quad \beta$$

Здесь греческие буквы (α и β) обозначают произвольные слова в некотором алфавите.

Определение 1.5. Чтобы задать алгорифм Маркова, необходимо задать:

- Алфавит
- Упорядоченный набор правил

1.2 Семантика алгорифма Маркова

Определение 1.6. Применимое правило. Назовем правило применимым к некоторой строке, если его левая часть входит в эту строку.

Определение 1.7. Шаг алгорифма Маркова. Пусть задана некоторая строка. Найдем первое применимое правило алгорифма Маркова. Возьмем самое левое вхождение левой части этого правила и заменим его на правую часть правила. Получившаяся строка будет называться результатом применения шага алгорифма Маркова к исходной строке.

Если ни одно правило не применимо, то мы будем говорить, что сделать шаг невозможно.

Определение 1.8. Результат применения алгорифма Маркова. Пусть задана некоторая строка. Будем применять шаги алгорифма Маркова, каждый следующий шаг применяя к результату предыдущего шага, пока шаги возможно выполнять и пока не было выполнено ни одного правила с точкой.

В случае, если какое-то из условий нарушено (т.е. не осталось применимых правил, либо выполнилось правило с точкой), то результат последнего выполненного шага назовем результатом применения алгорифма Маркова.

2 Грамматика

Определение 2.1. Чтобы задать грамматику, необходимо задать:

- Алфавит (A), состоящий из двух непересекающихся конечных множества символов, называемых алфавитами терминальных (A_T) и нетерминальных (A_H) символов:

$$A = A_T \cup A_H, \quad A_T \cap A_H = \varnothing$$

- Набор правил без точек в алфавите A, в каждом из которых которых левая часть содержит по крайней мере один нетерминальный символ
- Начальный символ — некоторый символ из A_H.

Определение 2.2. Будем говорить, что слово s задаётся грамматикой, если выполнены следующие два утверждения:

- Оно целиком состоит из терминальных символов данной грамматики;
- Существует последовательность применения правил, которая преобразует начальный символ в слово s. В данном случае мы разрешаем применять правила в произвольном порядке к произвольным вхождениям в строке, не требуя применения самого первого правила к самому левому вхождению.

Определение 2.3. Будем говорить, что язык задается грамматикой, если грамматика задает те и только те слова, которые входят в язык.

3 Сокращения записи

Для упрощения записи мы будем пользоваться следующими сокращениями, которые можно раскрыть в традиционную грамматику.

3.1 Альтернатива

Будем писать $\alpha \to \beta|\gamma$ вместо двух строчек

$$\begin{aligned} \alpha &\to \beta \\ \alpha &\to \gamma \end{aligned}$$

Данная запись означает «строчка α может быть преобразована либо в β, либо в γ».

3.2 Необязательная часть

Будем писать $\alpha \to [\beta]$ вместо двух строчек

$$\begin{aligned} \alpha &\to \\ \alpha &\to \beta \end{aligned}$$

Данная запись означает «строчка α может быть преобразована либо в β, либо в пустую строку».

3.3 Повторение

Будем писать $\alpha \to \{\beta\}^*$ вместо таких строчек

$$\begin{aligned} \alpha &\to S \\ S &\to \beta S \\ S &\to \varepsilon \end{aligned}$$

Здесь S — некоторый новый, ранее не встречавшийся в данной грамматике нетерминальный символ.

Данная запись означает повторение строчки β ноль или более раз.

Аналогично мы можем рассмотреть повторение строки один или более раз: $\alpha \to \{\beta\}^+$ вместо строчек

$$\begin{aligned} \alpha &\to S \\ S &\to \beta S \\ S &\to \beta \end{aligned}$$

Эти два типа повторения легко выражаются один через другой: как $\{\alpha\}^+$ можно представить как $\alpha\{\alpha\}^*$, так и $\{\alpha\}^*$ — как $\varepsilon|\{\alpha\}^+$.

4 Иерархия Хомского

Определение 4.1. Назовем грамматику неукорачивающей, если все правила в этой грамматике имеют в левой части не больше символов, чем в соответствующей правой части. Грамматика также может содержать правило вида $S \to \varepsilon$, но только если нетерминал S не содержится в правых частях правил.

Определение 4.2. Назовем грамматику бесконтекстной (контекстно-свободной), если все правила в этой грамматике имеют в левой части в точности один нетерминальный символ и ничего, кроме него.

Определение 4.3. Назовем грамматику построенной по регулярному выражению, если:

- Алфавит нетерминальных символов состоит в точности из одного символа
- Грамматика имеет единственное правило, левая часть которого — нетерминальный символ, а правая состоит только из терминальных символов. Также, в правой части разрешено использовать сокращения записи (операции «альтернатива», «необязательная часть» и «повторение»).

Определение 4.4. Будем говорить, что язык — неукорачивающий (бесконтекстный, автоматный), если он может быть задан соответствующей грамматикой.

Определение 4.5. Иерархия Хомского имеет 4 уровня:

Уровень в иерархии	Тип грамматики
0	Грамматика общего вида
1	Неукорачивающая
2	Бесконтекстная
3	Построенная по регулярному выражению

Легко заметить, что если грамматика (язык) принадлежит уровню k иерархии Хомского, то он принадлежит и всем уровням, меньшим k.

5 Регулярные выражения и конечные автоматы

Определение 5.1. Конечным автоматом мы назовем совокупность:

- двух алфавитов — алфавита терминальных символов A_T и алфавита состояний A_C;
- таблицы переходов, сопоставляющей паре символов — терминальному и состоянию — некоторое новое состояние: $T : A_T \times A_C \to A_C$;
- начального состояния и множества допускающих состояний.

Определение 5.2. Шаг применения конечного автомата. Если автомат находится в некотором состоянии $s \in A_C$, то результатом применения автомата к некоторому символу $a \in A_T$ будет новое состояние $T[s, a]$.

Определение 5.3. Будем говорить, что автомат допускает некоторую строку, если последовательное применение автомата ко всем символам из строки в порядке их записи в слове приводит автомат из начального в допускающее состояние.

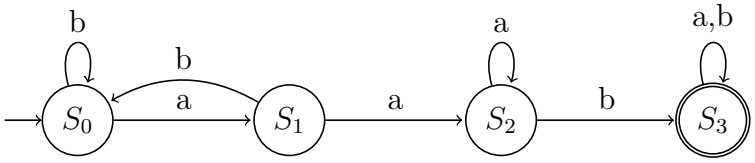
Определение 5.4. Будем говорить, что автомат задаёт язык, если он допускает те и только те слова, которые принадлежат языку.

5.1 Недетерминированные конечные автоматы

Введем альтернативный способ изображения конечных автоматов — в виде графа. Будем сопоставлять состояниям автомата точки, а клеткам таблицы переходов — дуги. Допускающие состояния будем обозначать двойным кружком, начальное состояние — входящей стрелкой. Если дуги для символа для какого-то состояния не указано, то появление такого символа ведёт к недопуску строки.

Пример задания одного и того же автомата с помощью таблицы и с помощью графа:

	a	b
S_0	S_1	S_0
S_1	S_2	S_0
S_2	S_2	S_3
S_3^*	S_3	S_3



Определение 5.5. Недетерминированный конечный автомат — это конечный автомат, в котором в каждой клетке таблицы переходов записано не одно состояние, а множество состояний. То есть, функция переходов задаётся иначе: $T : A_T \times A_C \to 2^{A_C}$.

Определение 5.6. Мы будем говорить, что недетерминированный конечный автомат допускает строку, если на каждом шаге применения автомата к символам входной строки (то есть, на символе $t \in A_T$ и в состоянии $s \in A_C$) найдётся такой переход (из множества допустимых переходов $T[t, s]$), что после последнего шага автомата он окажется в допускающем состоянии.

То же самое можно выразить несколько парадоксальной фразой: недетерминированный автомат на каждом шагу «угадывает», какой из вариантов перехода выбрать, чтобы достигнуть допускающего состояния.

Определение 5.7. Недетерминированный конечный автомат с эпсилон-переходами — конечный автомат, в котором к алфавиту терминальных символов добавлен «пустой» символ ε, не встречающийся во входной строке.

Определение 5.8. Назовём состояние r ε-достижимым из состояния s, если существует последовательность ε-переходов, позволяющая достичь r из s.

Определение 5.9. Мы будем говорить, что недетерминированный конечный автомат с ε-переходами допускает строку, если на каждом шаге применения автомата к символам входной строки (то есть, на символе $t \in A_T$ и в состоянии $s \in A_C$) найдётся такое ε-достижимое состояние r и такой переход из него (из множества допустимых переходов $T[t, r]$), что после последнего шага автомата допускающее состояние окажется ε-достижимо.

Иными словами, недетерминированный конечный автомат с ε-переходами может в любой момент самопроизвольно выполнить любое количество ε-переходов, если это ведёт к цели (к допуску строчки).

Несложно показать, что по любому недетерминированному автомату с ε-переходами можно построить детерминированный, задающий тот же язык (для этого надо рассмотреть автомат с алфавитом состояний 2^{A_C}). Поэтому классы языков, задаваемые и детерминированными и недетерминированными автоматами, одинаковы. Однако, в разных задачах удобен разный тип автоматов.

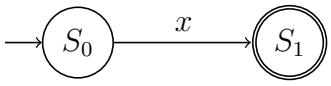
5.2 Соответствие регулярных выражений конечным автоматам

Известно, что грамматики, построенные по регулярным выражениям, задают в точности то же множество языков, что и конечные автоматы. Мы можем это показать, например, двумя включениями: показав, как заданный грамматикой язык задать конечным автоматом — и как язык, заданный конечным автоматом, задать с помощью грамматики, построенной по регулярному выражению.

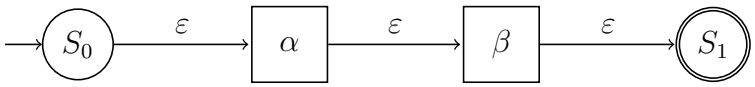
Мы покажем, как преобразовать регулярное выражение в недетерминированный конечный автомат, задающий тот же язык. Обратное преобразование также можно произвести, но оно останется за рамками данного курса.

В регулярном выражении возможны 4 типа конструкций: литерал (терминальный символ), конкатенация ($\alpha\beta$), альтернатива ($\alpha|\beta$) и повторение ($\{\alpha\}^+$). Нам достаточно предложить способ построения автомата для литералов и способы построения более сложных автоматов на основе уже имеющихся в остальных трёх типах конструкций.

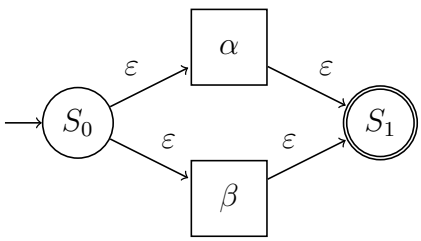
1. Литерал. Автомат, допускающий символ x (и только его), выглядит так:



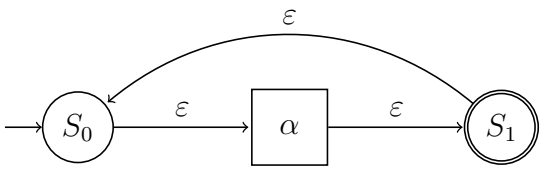
2. Конкатенация. Пусть даны два автомата, задающие α и β (они схематично изображены квадратами на графе). Тогда автоматом, задающим $\alpha\beta$, будет



3. Альтернатива. Пусть даны два автомата, задающие α и β. Тогда автоматом, задающим $\alpha|\beta$, будет



4. Повторение. Пусть дан автомат, задающий α. Тогда автоматом, задающим $\{\alpha\}^+$, будет



При помощи этих четырёх примитивов можно последовательно построить автомат, соответствующий любому регулярному выражению.