Занятие 7

Алгоритмы Маркова (нормальные алгорифмы) Программа (схема) алгоритма Маркова представляет собой конечный линейно упорядоченный набор редукций (правил) двух видов: $u \to v$ (обыкновенное правило) и $u \to \cdot v$ (заключительное правило), где u, v — слова в алфавите Σ. Вычисление начинается с входного слова $P \in \Delta^* \subseteq \Sigma^*$. Применяется первая редукция, левая часть которой (редекс) u входит в P. Применение редукции состоит в замене самого левого вхождения редекса u на контрактум v. Затем те же действия производят с полученным словом, и т.д. Процесс прерывается в двух случаях: когда применилась заключительная редукция или когда ни одна из редукций не может быть применена. В обоих случаях полученное слово считается результатом вычисления.

В следующих задачах требуется построить нормальные алгорифмы, реализующие указанные преобразования слов.

1. Приписать заданное слово Q к входному слову а) слева, б) справа.

$$\text{a)} \quad \to \cdot \; Q \qquad\qquad \text{b)} \quad \begin{array}{rcll} \alpha\xi & \to & \xi\alpha & (\xi \in \Delta) \\ \alpha & \to \cdot & Q & \\ & \to & \alpha & \end{array}$$

2. Отсортировать входное слово $P \in \{0,1\}^*$ по возрастанию.

$$10 \;\to\; 01$$

3. Извлечь из входного слова $P \in \{0,1,\#\}^*$ двоичное слово, расположенное между первым и вторым вхождением разделителя $\#$.

$$\begin{array}{rcll} \gamma\eta & \to & \gamma & (\eta \in \{0,1,\#\}) \\ \gamma & \to \cdot & & \\ \beta\xi & \to & \xi\beta & (\xi \in \{0,1\}) \\ \beta & \to & \gamma & \\ \xi\alpha & \to & \alpha & (\xi \in \{0,1\}) \\ \alpha & \to & \beta & \\ \# & \to & \alpha & \end{array}$$

Преобразование таково:
$\mathbf{u}\#\mathbf{v}\#\mathbf{w} \to \mathbf{u}\alpha\mathbf{v}\#\mathbf{w} \twoheadrightarrow \alpha\mathbf{v}\#\mathbf{w} \to \beta\mathbf{v}\#\mathbf{w} \twoheadrightarrow \mathbf{v}\beta\#\mathbf{w} \to \mathbf{v}\gamma\#\mathbf{w} \twoheadrightarrow \mathbf{v}$.

4. Удвоить входное слово $P \in \{0,1\}$. Использовать дополнительные буквы-двойники — $\bar{0}$ и $\bar{\bar{0}}$ для 0, а также $\bar{1}$ и $\bar{\bar{1}}$ для 1:
   $P \to \alpha P \twoheadrightarrow \bar{P}P\alpha \twoheadrightarrow \bar{\bar{P}}\,P\alpha \twoheadrightarrow PP\alpha \to PP.$

$$
\begin{array}{rcll}
\xi\bar{\eta} & \to & \bar{\eta}\xi & (\xi,\eta \in \{0,1\}) \\
\alpha\xi & \to & \bar{\xi}\xi\alpha & (\xi \in \{0,1\}) \\
\bar{\xi} & \to & \bar{\bar{\xi}} & (\xi \in \{0,1\}) \\
\bar{\bar{\xi}} & \to & \xi & (\xi \in \{0,1\}) \\
\alpha & \to & \cdot & \\
 & \to & \alpha &
\end{array}
$$

Вычислимые функции и алгоритмы Объем понятия «вычислимая функция» — это те функции, которые можно запрограммировать.

Основной вопрос: какими средствами, какой язык программирования? Желаемый ответ, который не удается формализовать, — следует допустить использование всех языков программирования, даже тех, которые пока не придумали.

Наблюдение, которое несколько упрощает дело: все имеющиеся языки допускают компиляцию в один, очень простой. В качестве такого простого языка можно выбрать, например, ассемблер, машины Тьюринга, алгоритмы Маркова и др. . "Запас вычислимости" оказывается одним и тем же! Но эти простые языки очень неудобны для реального программирования. **Тезис Маркова: Все вычислимые словарные функции могут быть вычислены алгоритмами Маркова.**

Действие — блок команд, который никогда не вызывает зацикливания (можно рассматривать как одну макрокоманду). *Проверка* — действие с результатом 0 или 1. Достаточно общая схема программы вычисления функции на языке "низкого уровня":

```
<инициализация>          // действие
WHILE <проверка>  DO
   <тело цикла>          // действие
DONE
RETURN ...               // действие
```

Одна итерация — один "шаг" работы алгоритма.

5. Конкретизировать схему применительно к алгоритмам Маркова.

Ограничиваемся подмножествами множества Σ^* всех слов в алфавите Σ и вычислимыми частичными функциями из Σ^* в Σ^*.

Напр., $\Sigma = \{0, 1\}$. Имеется представление $\Sigma^* = \{\Lambda, S(\Lambda), S(S(\Lambda)), \ldots\}$, где S — вычислимая функция, а также вычислимые в обе стороны биекции между $(\Sigma^*)^2$ и Σ^*. Указанное представление позволяет использовать Σ^* также и в качестве N: число n представляется словом $S^n(\Lambda)$. Аналогично, слова в алфавите Σ можно использовать в качестве представления пар слов.

6. Для $\Sigma = \{0, 1\}$ описать алгоритмы вычисления S и биекций. (В качестве описания алгоритма подходит любая инструкция, которую понятно как запрограммировать на любом из известных языков программирования.)

Определение. Множество $A \subseteq \Sigma^*$ называется *разрешимым*, если вычислима его характеристическая функция

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \in \Sigma^* \setminus A \end{cases} .$$

7. Проверить разрешимость множеств:
   - всех нечетных чисел;
   - данного конечного множества слов;
   - множества всех обратимых 2×2-матриц с коэффициентами из N.
   Взять $\Sigma = \{0, 1, \#\}$.

8. Если $f : \Sigma^* \to \Sigma^*$ — вычислимая частичная функция, у которой нет тотального вычислимого продолжения (пример на лекции), то ее область определения неразрешима.

Определение. Множество $A \subseteq \Sigma^*$ называется *перечислимым*, если
(Вариант 1) A есть область значений некоторой вычислимой функции.
(Вариант 2) A есть область значений некоторой тотальной вычислимой функции или $A = \emptyset$.
(Вариант 3) A есть область определения некоторой вычислимой функции.

(Вариант 1) $\Leftrightarrow$ (Вариант 2) $\Leftrightarrow$ (Вариант 3)

9. Проверить перечислимость множеств:

   (а) каждого разрешимого множества;
   (б) множества всех натуральных чисел, представимых в виде разности квадратов двух чисел;

10. Если множество $A \subseteq \Sigma^*$ и его дополнение $\Sigma^* \setminus A$ перечислимы, то оба они разрешимы.

Домашнее задание

11. Ниже натуральные числа представляются словами в унарном алфавите: $\bar{n} = \underbrace{||\ldots|}_{n \text{ раз}}$. Реализовать сложение и вычисление модуля разности двух натуральных чисел:

$$\bar{m}\#\bar{n} \twoheadrightarrow \overline{(m+n)}, \qquad \bar{m}\#\bar{n} \twoheadrightarrow \overline{|m-n|}.$$

12. Преобразовать натуральное число n в пару $[n/2]$, $n \ mod \ 2$ (числа — в унарной записи; выбрать произвольный разделитель).

13. Вычислить НОД двух натуральных чисел $d = (x, y)$ с помощью равенства $(x, y) = (y, x - y)$ для $x \geq y > 0$. Предполагаемая реализация использует три дополнительные буквы a, b, c; $m = \min\{x, y\}$, $M = \max\{x, y\}$:
$\underbrace{|\ldots|}_{x} * \underbrace{|\ldots|}_{y} \twoheadrightarrow \underbrace{a\ldots a}_{m} * \underbrace{b\ldots b}_{M-m} \twoheadrightarrow \underbrace{c\ldots c}_{m} * \underbrace{|\ldots|}_{M-m} \twoheadrightarrow \underbrace{|\ldots|}_{m} * \underbrace{|\ldots|}_{M-m} \twoheadrightarrow \ldots$

14. Пусть тотальная вычислимая функция f монотонна в следующем смысле: если слово u является собственным началом слова v, то $f(u)$ — собственное начало слова $f(v)$. Доказать разрешимость области значений функции f.

15. Пусть все правила порождающей грамматики G имеют вид $u \to v$, где u — одна служебная буква, а $v \neq \Lambda$. Тогда порожденный язык $L(G)$ разрешим.

16. Доказать, что объединение и пересечение двух перечислимых множеств перечислимо.

17. Графиком частичной функции $f : \Sigma^* \to \Sigma^*$ называется множество всевозможных пар $(x, f(x))$, где x пробегает область определения функции f. Доказать, что функция f вычислима в том и только в том случае, когда ее график перечислим. (Пары слов условимся представлять одним словом в алфавите Σ с помощью вычислимой биекции $(\Sigma^*)^2$ на Σ^*.)