Практика №4 по курсу «Теория алгоритмов» «Альтернативные модели вычислений»

Группы ФТ-301, ФТ-302

Разберём упражнение с одной из первых практик: является ли множество разрешимых языков замкнутым относительно операции гомоморфизма?

Напомним, что гоморфизм языка $\mathcal{L} \subset \Sigma^*$ задается функцией $\mathbf{h} : \Sigma \mapsto \Delta^*$ таким образом, что для любого слова $w \in \mathcal{L}$ значение $\mathbf{h}(w)$ определяется как конкатенация образов букв слова $w = c_1 c_2 \cdots c_n$: $\mathbf{h}(w) = \mathbf{h}(c_1) \cdot \mathbf{h}(c_1) \cdots \mathbf{h}(c_n)$.

Несложно показать, что множество перечислимых языков замкнуто относительно операции гомоморфизма. Действительно, пусть есть перечислимый язык $\mathcal{L}$ и дана МТ M, перечисляющая слова этого языка. Тогда можно построить M' для перечисления слов $\mathbf{h}(\mathcal{L})$ следующим образом: запустим МТ M и при появлении очередного слова $w \in \mathcal{L}$ запустим процедуру применения гомоморфизма $\mathbf{h}(w)$, после чего сравним получившееся слово с уже перечисленными ранее и в случае его уникальности напечатаем $\mathbf{h}(w)$ на ленте.

Заметим также, что несложно показать, что множество разрешимых языков замкнуто относительно операции **нестирающего** гомоморфизма, то есть любой разрешимый язык $\mathcal{L}$ под действием функции $\mathbf{h} : \Sigma \mapsto \Delta^+$ переходит в разрешимый язык. Для этого достаточно заметить, что длина образа нестирающего гоморфизма всегда не меньше, чем длина исходного слова $|w| \leq |\mathbf{h}(w)|$, а это значит, что для проверки принадлежности слова w' языку $\mathbf{h}(\mathcal{L})$ достаточно перебрать все слова $w \in \mathcal{L}$ не длиннее чем $|w'|$ и проверить, могли ли они являться прообразами для слова w'.

Для доказательства того, что множество разрешимых языков не замкнуто относительно операции **стирающего** гомоморфизма, можно построить следующий контрпример. Рассмотрим разрешимый язык $\mathcal{L} = \{\langle M, x, t\rangle \mid$ машина Тьюринга M останавливается на входе x за t шагов $\}$. Пусть алфавит данного языка состоит из трёх непересекающихся множеств символов $\Sigma = \Sigma_M \cup \Sigma_x \cup \Sigma_t$, где каждая часть используется для кодирования конкретной компоненты кортежа. Тогда если рассмотреть гомоморфизм $\mathbf{h}(x) = \begin{cases} \varepsilon, & x \in \Sigma_t \\ x, & \text{иначе} \end{cases}$, то получим язык $\mathbf{h}(\mathcal{L}) = \{\langle M, x\rangle \mid$ машина Тьюринга M останавливается на входе $x\}$, который не является разрешимым.

1 Альтернативные модели вычислений

Помимо частично рекурсивных функций и машин Тьюринга есть другие модели, созданные примерно в то же время для формализации понятия алгоритм. Рассмотрим ещё пару формализмов таких как алгорифмы и грамматики.

1.1 Алгорифмы (нормальные алгоритмы Маркова)

Нормальный алгоритм Маркова задается упорядоченным списком R, состоящим из обычных правил $\mathcal{L} \to \mathcal{D}$ или же терминальных правил $\mathcal{L} \to \cdot\mathcal{D}$. Алгорифм с заданными списком правил работает итеративно, где применение одной итерации к слову w осуществляется следующим образом:

- Если среди правил нет такого, что левая часть $\mathcal{L}$ входит в w как подслово, то алгорифм заканчивает свою работу и его выходом является слово w
- Иначе, находится первое правило из упорядоченного списка R такое, что его левая часть $\mathcal{L}$ входит в w как подслово
- Выбирается самое левое вхождение левой части в w, то есть такое разбиение $w = A \cdot \mathcal{L} \cdot B$, что $|A|$ — минимален
- Результатом данной итерации является слово $w' = A \cdot \mathcal{D} \cdot B$
- Если выбранное правило было терминальным, то алгорифм заканчивает свою работу и его выходом является слово w'

1.1.1 Пример. Удаление повторящихся букв

Рассмотрим следующий алгоритм, удаляющий повторяющиеся буквы для слов над бинарным алфавитом $\{a, b\}$:

$$R = \begin{cases} aa \to a \\ bb \to b \end{cases}$$

Последовательное применение алгорифма к слову $aabbba$ породит следующую цепочку промежуточных результатов:

$$aabbba \to abbba \to abba \to aba$$

где aba является результатом работы алгорифма

1.1.2 Пример. Перевод числа из бинарной системы в унарную

Рассмотрим более содержательный пример перевода числа из бинарной системы в унарную. Поставим задачу следующим образом: дано двоичное число $w \in \{0,1\}^+$ по которому необходимо построить слово $w' \in \{x\}^+$ такое, что $|w'| = \texttt{int(w, base: 2)}$. Например, для $w = 101$ необходимо построить $w' = xxxxx$.

Одна из возможных структур алгорифма может выглядеть следующим образом:

- Заменим все 1 на подстроки $0x$
- Каждую пару $x0$ можно преобразовать в подстроку $0xx$, соответствующую переносу слагаемого 2^i в предыдущий разряд $i-1$
- Удалить все нули

Если записывать данное описание в виде правил алгорифма получится следующий список:

$$R_1 = \begin{cases} 1 \to 0x \\ x0 \to 0xx \\ 0 \to \varepsilon \end{cases}$$

Ход вычислений для строки 101 будет выглядеть следующим образом:

$$101 \to \mathbf{0x}01 \to 0x0\mathbf{0x} \to 0\mathbf{0xx}0x \to 00x\mathbf{0xx}x \to 00\mathbf{0xx}xxx \to 00xxxxx \to 0xxxxx \to xxxxx$$

Несложно доказать корректность работы алгорифма:

- Ясно, что сначала будет применено первое правило, после чего в строке не останется единиц и первое правило больше никогда не будет применяться
- Для каждого x-а из строки $w' \in \{0,x\}^+$ определим его «вес» как 2^z, где z — это количество нулей после этого x-а. Например, в строке $000x0xx0x$ получатся следующие веса соответственно: $[4,2,2,1]$
- Ясно, что после применения первого правила суммарный вес x-ов в строке будет равен значению $\texttt{int(w, base: 2)}$
- Легко показать, что применение второго правила не меняет вес строки, так как однин x с весом 2^z заменяется на два с весом 2^{z-1}
- Так как третее правило будет применятся к строке вида $0^n x^m$ (иначе было бы применимо второе правило), то удаление нулей из неё также не меняет веса строки, а значит в конце концов получится строка x^k, где $k = \texttt{int(w, base: 2)}$

1.1.3 Небольшое усложнение

Немного усложним себе задачу и постараемся перевести число в унарную систему таким образом, чтобы финальная строка содержала символы 1, а не вспомогательные символы x. То есть результатом работы алгорифма на строке 101 должна быть строка 11111.

Заметим, что данную модификацию задачи нельзя решить простым добавлением ко множеству правил трансформации $x \to 1$:

$$R_1' = \begin{cases} 1 \to 0x \\ x0 \to 0xx \\ 0 \to \varepsilon \\ x \to 1 \end{cases}$$

так как в результате алгорифм зациклится на любой строке, содержащей хотя бы одну единицу:

$$1 \to \mathbf{0x} \to x \to 1 \to \cdots$$

Для того, чтобы решить данную задачу построим алгорифм следующим образом:

- Запустим алгоритм, решающий исходную формулировку задачи
- После его работы добавим в начало строки символ $\square$
- Произведем замену пары символов $\square x$ на $1\square$
- Удалим символ $\square$

Таким образом получится следующий набор правил:

$$R_2 = \begin{cases} \square x \to 1\square \\ \square \to \cdot\varepsilon \\ 1 \to 0x \\ x0 \to 0xx \\ 0 \to \varepsilon \\ \varepsilon \to \square \end{cases}$$

Такой алгорифм будет работать следующим образом:

$$101 \to \mathbf{0x}01 \to 0x0\mathbf{0x} \to 0\mathbf{0xx}0x \to 00x\mathbf{0xx}x \to 00\mathbf{0xx}xxx \to 00xxxxx \to 0xxxxx \to xxxxx \to \square xxxxx \to \mathbf{1}\square xxxx \to 1\mathbf{1}\square xxx \to 11\mathbf{1}\square xx \to 111\mathbf{1}\square x \to 1111\mathbf{1}\square \to 11111$$

1.1.4 Теорема о сочетании алгорифмов (*не было на практике*)

Не самым тривиальным образом контруируется алгорифм, являющийся результатом последовательного применения пары алгорифмов R_1 и R_2, то есть такой набор правил R_0, что $R_0(w) = R_2(R_1(w))$. Для упрощения доказательства будем считать, что алгорифмы R_1 и R_2 содержат **только обычные правила** и оперируют символами общего алфавита Σ. Один из возможных подходов для комбинации алгорфимов выглядит следующим образом:

- Сконвертируем все символы алфавита Σ в соответствующие символы алфавита $\Sigma' = \{x' \mid x \in \Sigma\}$
- Запустим модифицированный алгорифм R_1 для полученной строки
- Сконвертируем все символы алфавита Σ' в соответствующие символы алфавита $\Sigma'' = \{x'' \mid x \in \Sigma\}$
- Запустим модифицированный алгорифм R_2 для полученной строки

Чтобы добиться такой последовательности выполнения породим в начале строки три вспомогательных символа $\square_1, \square_2, \square_3$ и добавим правила конвертации символов между алфавитами следующим. Итоговый набор правил будет выглядеть следующим образом:

$$R_0 = \begin{cases} \square_1 x \to x'\square_1 & \text{для всех } x \in \Sigma \\ \square_1 \to \varepsilon & \\ \mathcal{L}' \to \mathcal{D}' & \text{для всех } \mathcal{L} \to \mathcal{D} \in R_1 \\ \square_2 x' \to x''\square_2 & \text{для всех } x \in \Sigma \\ \square_2 \to \varepsilon & \\ \mathcal{L}'' \to \mathcal{D}'' & \text{для всех } \mathcal{L} \to \mathcal{D} \in R_2 \\ \square_3 x'' \to x\square_3 & \text{для всех } x \in \Sigma \\ \square_3 \to \cdot\varepsilon & \\ \varepsilon \to \square_3\square_2\square_1 & \end{cases}$$

1.2 Грамматики (*не было на практике*)

Вершиной иерархии грамматик Хомского являются неограниченные грамматики, которые представляют из себя набор правил $p \to q$, где $p \in (\Sigma \cup N)^* N (\Sigma \cup N)^*, q \in (\Sigma \cup N)^*$ (то есть в левой части должен быть хотя бы один нетерминал), где Σ — это алфавит терминальных символов, а N — алфавит нетерминальных символов. Формально, грамматика задается четвёркой $G = \langle N, \Sigma, P, S \rangle$, где P — это множество правил $p \to q$, а $S \in N$ — это стартовый нетерминал. Для конкретной грамматики G и пары слова x будем говорить, что из x непосредственно выводится слово y ($x \Rightarrow_G y$), если $\exists u, v, p, q \in (\Sigma \cup N)^* : (x = upv) \wedge (p \to q \in P) \wedge (y = uqv)$. Определим отношение $\Rightarrow_G^*$ как транзитивное замыкание отношения $\Rightarrow_G$. Тогда грамматика G задает язык всех слов из терминальных символов, которые выводятся из стратового нетерминала S за конечное число шагов, то есть $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

Множество языков, задаваемых неограниченными грамматиками совпадает с перечислимыми языками.

1.2.1 Пример. Язык всех квадратов натуральных чисел

Построим грамматику, задающую следующий язык $\mathcal{L} = \{1^c \mid c \text{ - составное число}\}$.

Воспользуемся для построения грамматики тем, что для любого составного числа c существует пара чисел $a, b > 1$ такие, что $c = a \cdot b$. Используя этот факт построим грамматику, которая сгенерирует строку вида $A^a B^b$, после чего перенесёт все нетерминалы B в левую часть, где при каждом переносе через A будет добавляться новая единица в строку. Таким образом правила грамматики будут выглядеть следующим образом:

$$\begin{cases} S \to LAA\square BBR \\ \square B \to \square BB \\ A\square \to AA\square \\ \square \to \varepsilon \\ AB \to B1A \\ 1B \to B1 \\ LB \to L \\ AR \to R \\ 1R \to R1 \\ LR \to \varepsilon \end{cases}$$