ЛЕКЦИЯ 9

ФУНКЦИОНАЛЬНЫЕ ЭЛЕМЕНТЫ. ЗАДАЧИ РАСПОЗНАВАНИЯ. ВЫЧИСЛЕНИЯ С ПОДСКАЗКОЙ

1. Функциональные элементы

Ранее в качестве вычислительной машины фигурировала машина Тьюринга (детерминированная, недетерминированная, альтернирующая). Но помимо машины Тьюринга существуют и другие модели вычисления. Примером может послужить **схема из функциональных элементов**. На рисунке ниже приведены модели элементов отрицания, конъюнкции и дизъюнкции. В элементах конъюнкции и дизъюнкции может содержаться произвольное число входов.

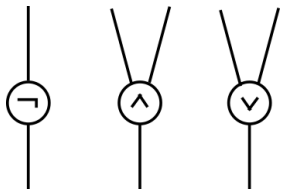

Рис. 9.1

Эти базовые элементы работают как логические операторы. Функциональными элементами можно моделировать вычисления на чипе (микропроцессоре), а машиной Тьюринга — на процессоре.

Рассмотрим схему элемента XOR.

В формуле $(x \wedge \neg y) \vee (\neg x \wedge y)$ два раза учитывается x. Если в качестве x будет фигурировать длинная формула или сложное выражение, то экономичность схемы станет очевидной (выгоднее учитывать сложное выражение один раз).

Определим понятие схемы формально.

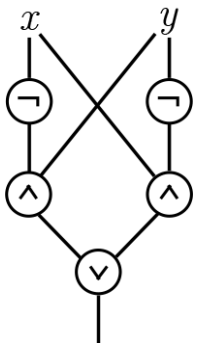

Рис. 9.2

Определение 28: Схема *— это ориентированный граф без циклов с несколькими источниками, одним истоком и метками на вершинах (¬, ∧, ∨). Имеется ограничение на входные степени: если метка ¬, то входная степень 1.* ♣

Основные характеристики схемы — это **размер** (количество вершин в графе) и **глубина** (максимальная длина пути от источника до стока).

Теорема 14 Любую функцию $\{0,1\}^n \to \{0,1\}$ можно выразить схемой размера $O(2^n)$.∗

Док-во: Можно воспользоваться принципом Дирихле с рассмотрением константы C. Если $C < 2$, то существует функция, для которой необходима схема размера $\Omega(C^n)$.

2. Задачи распознавания

Определение 29: *$SIZE(S(n))$ — это множество языков, которые можно распознать семейством схем размера $O(S(n))$.* ♣

Речь идет именно о семействе схем, потому что для любой длины слова n существует своя схема. В этом состоит их ключевое отличие от понятия алгоритма (потому что в алгоритме используется одна и та же закономерность для входов любой длины).

Определение 30: $P/poly = \bigcup\limits_{c=1}^{\infty} SIZE(n^c)$. ♣

Класс P/poly нельзя реализовать на практике, потому что в нем есть невычислимые языки. Любой унарный язык входит в P/poly (**унарный язык** — это язык, состоящий из одних единиц, т. е. содержащий слова 1^k).

$$L \cap \{0,1\}^n = \emptyset \quad \Rightarrow \quad \text{взять тождественную ложь (конъюнкцию P и ¬P)},$$

$$L \cap \{0,1\}^n = \{1\}^k \quad \Rightarrow \quad \text{взять конъюнкцию.}$$

Среди унарных языков существует невычислимые. Например, следующий язык неразрешим, но входит в P/poly:

$$\{1^k | M_k(k) \text{ останавливается}\}.$$

Конспект не проходил проф. редактуру, создан студентами и, возможно, содержит смысловые ошибки. Следите за обновлениями на lectoriy.mipt.ru.

Теорема 15 $\mathrm{P} \subsetneq \mathrm{P/poly}$. *

Док-во: Можно эмулировать машину Тьюринга при помощи схем. У машины Тьюринга есть протокол (упорядоченный список конфигураций в процессе ее работы). Любой символ конфигурации зависит от небольшого числа символов предыдущей конфигурации. Эту зависимость можно записать в виде схемы. Можно копировать схему и создать таблицу из копий схемы, которая симулирует протокол. При этом общий размер будет равен $\mathrm{O}(p(n))$, где $p(n)$ — время работы исходной машины Тьюринга.

3. Вычисления с подсказкой

Определение 31: *Алгоритм M распознает L за полиномиальное время $T(n)$ с подсказкой длины $a(n)$, если существует последовательность $\{\alpha_n\}_{n=1}^{\infty}$, такая что*

$$|\alpha_n| \leqslant a(n), \quad x \in L \quad \Rightarrow \quad M(x, \alpha_{|x|}) = 1,$$

$M(x, \alpha_{|x|})$ вычисляется за полиномиальное время.

Существует соответствующий класс — $\mathrm{DTIME}(T(n))/a(n)$.

Любой унарный язык лежит в $\mathrm{DTIME}(n)/1$ (поскольку нужен один бит чтобы сказать, лежат ли n единиц в языке или нет).

Теорема 16 $\mathrm{P/poly} = \bigcup\limits_{c,d=1}^{\infty} \mathrm{DTIME}(n^c)/n^d$. *

Док-во: $\boxed{\subset}$ Пусть α_n — схема, M — вычисление схемы. В качестве подсказки можно взять саму схему, и алгоритм будет вычислением схемы.

$\boxed{\supset}$ Преобразуем M в схему и зафиксируем α_n.

Теорема 17 (Карпа–Липтона) Если $\mathrm{NP} \subseteq \mathrm{P/poly}$, то полиномиальная иерархия обрывается на втором уровне:

$$\mathrm{NP} \subseteq \mathrm{P/poly} \quad \Rightarrow \quad \mathrm{PH} = \Sigma_2^p.$$

Док-во: Достаточно доказать, что $\Sigma_2^p = \Pi_2^p$. Для этого докажем, что $\Pi_2^p\text{-SAT} = \Sigma_2^p$. Напомним, что

$$\Pi_2^p\text{-SAT} = \{\phi | \, \forall\, u \, \exists\, v \; \phi(u, v) = 1\},$$

где ϕ — некая булева формула, u, v — группы переменных.

$$\Pi_2^p = \{L : \exists\, M \; x \in L \quad \Leftrightarrow \quad \forall\, u \, \exists\, v \; M(x, u, v) = 1\}.$$

Π_2^p-SAT — полная задача в Π_2^p. Проводя аналогию с теоремой Кука–Левина, можно установить, что по x можно построить следующую формулу ϕ:

$$x \to \phi : \quad \phi(u, v) = 1 \quad \Leftrightarrow \quad M(x, u, v) = 1.$$

Для подготовки к экзаменам пользуйтесь учебной литературой. Об обнаруженных неточностях и замечаниях просьба писать на pulsar@phystech.edu

Если полная задача лежит в Σ_2, то поскольку любая другая задача из Π_2 к ней сводится, то и любая сводящаяся задача тоже лежит в Σ_2. Если Π_2 — подмножество Σ_2, то они равны.

$$\phi \in \Pi_2\text{-SAT} \quad \Leftrightarrow \quad \forall\, u\ \phi(u, *) \in \text{SAT},$$

где «*» — фиксированный аргумент ϕ.

Если NP $\subset$ P/poly, то выполнимость распознается некоторой схемой C полиномиального размера. Также существует некоторая другая схема D, которая преобразует u в $\phi(u, *)$. Таким образом,

$$\phi(u, *) \in \text{SAT} \quad \Leftrightarrow \quad C(D(u)) = 1.$$

Если сущетвует схема, которая распознает выполнимость, то существует схема C', которая находит выполняющий набор.

$$\exists\, C' : \forall\, u\ \phi(u, C'(u)) = 1 \quad \Leftrightarrow \quad \forall\, u\ \phi(u, *) \in \text{SAT}.$$

ϕ — формула из Σ_2. Размер C' будет полиномиальным. Значит, Π_2-SAT $\in \Sigma_2^p$.