Классические и квантовые вычисления

А. Шень (вр.и.о. А. Китаева); написал М. Вялый

4 лекция (5/03/98)

$$\begin{array}{ccc} \mathrm{NP} & & \text{co-NP} \\ \cup & & \cup \\ & \mathrm{P} & \end{array}$$

На этой лекции будет дано объяснение формулировки последней теоремы из предыдущей лекции и доказательство этой теоремы. На предыдущих лекциях были введены классы $\mathcal{P}$, $\mathcal{NP}$. Есть ещё класс co-$\mathcal{NP}$, дополнения к языкам из этого класса принадлежат $\mathcal{NP}$. Взаимное соотношение этих классов можно изобразить диаграммой включений, показанной на врезке (обратите внимание на то, что знак строгого включения подразумевает и возможное равенство этих классов — никому ещё не удалось доказать, что $\mathcal{P} \neq \mathcal{NP}$ или $\mathcal{P} \neq$ co-$\mathcal{NP}$, или хотя бы что $\mathcal{NP} \neq$ co-$\mathcal{NP}$).

1. Игры, в которые играют машины

Рассмотрим игру, в которую играют два игрока, будем их называть белые (Б) и чёрные (Ч). Игрокам сообщается некоторое слово x и они делают ходы по очереди. Ходы могут быть описаны словами длины $p(|x|)$, где $p(\cdot)$ — некоторый полином. После того как игра завершена, её результат определяется некоторым предикатом $W(x, w_1, b_1, \dots) \in \mathcal{P}$ (выиграли ли белые).

Каждой игре можно сопоставить множество $L = \{x|\text{Б имеет выигрышную стратегию}\}$. Разнообразные классы можно определить как множества, соответствующие тем или иным видам игр.

- $\mathcal{P}$: никто не делает ходов.
- $\mathcal{NP}$: 1 ход белых. Другими словами, это множества вида $\{x|\exists w_1 W(x, w_1)\}$.
- co-$\mathcal{NP}$: 1 ход белых, но в этот класс включаются множества $\{x|\text{ Ч имеет выигрышную стратегию}\}$.
- Σ_2: 1 ход белых, 1 ход чёрных. Включаются множества, для которых выигрышную стратегию имеют белые, т. е. $\{x|\exists w_1 \forall b_1 W(x, w_1, b_1)\}$.
- Π_2: 1 ход белых, 1 ход чёрных. Включаются множества, для которых выигрышную стратегию имеют чёрные, т. е. $\{x|\forall w_1 \exists b_1 W(x, w_1, b_1)\}$.

Теорема. $\mathcal{BPP} \subset \Sigma_2 \cap \Pi_2$.

Доказательство. (Лаутеман?)

Для доказательства нужно научиться формулировать свойство "множество содержит много элементов" с использованием кванторов универсальности и существования. Мы сделаем это, предполагая, что рассматриваются подмножества некоторой конечной группы. Пусть G — группа, множество $X \subset G$. Свойство, которым мы будем отличать большие множества от малых, состоит в том, что некоторым количеством сдвигов X можно покрыть всю группу

$$\exists g_1, \dots, g_k : \bigcup_i g_i X = G. \qquad (*)$$

Чтобы найти подходящее значение k, найдём те случаи, когда условие $(*)$ заведомо выполняется, и те, когда оно заведомо не выполняется.

Если
$$k|X| < |G|, \tag{1}$$
условие $(*)$ заведомо ложно.

Условие $(*)$ заведомо истинно, если вероятность события $\bigcup_i g_i X = G$ больше 0 для случайных независимых $g_1, \ldots, g_k$. Другими словами, $Pr(\bigcup_i g_i X \subset G) < 1$. Вероятность того, что случайный сдвиг X не покрывает (не содержит) некоторый фиксированный элемент равна по очевидным причинам $1 - \dfrac{|X|}{|G|}$. Поскольку покрывается $|G|$ элементов k сдвигами, вероятность события $\bigcup_i g_i X \subset G$ меньше $|G|\left(1 - \frac{|X|}{|G|}\right)^k$. Итак, при
$$|G|\left(1 - \frac{|X|}{|G|}\right)^k < 1 \tag{2}$$
условие $(*)$ заведомо выполняется.

Рассмотрим теперь некоторый язык $L \in \mathcal{BPP}$. Для него, как объяснялось на прошлой лекции, можно найти полиномиально вычислимый предикат $R(x,y)$ и полином $p(x)$ такие, что доля элементов множества $X_x = \{y| \ |y| = p(|x|) \& R(x,y)\}$, т. е. число $|X_x|/2^{p(|x|)}$ различает слова, принадлежащие языку (для них это число больше $1-\varepsilon$), и слова, языку не принадлежащие (для таких слов это число меньше ε). Параметр ε мы выберем позже, сейчас отметим, что его величина может быть экспоненциально мала.

Введём искусственно структуру группы на множестве y так, чтобы произведение в этой группе было полиномиально вычислимым (например, покомпонентное сложение по модулю два). Запишем следующее Σ_2-условие
$$\exists g_1, \ldots, g_k \ \forall y (|y| = p(|x|)) \Rightarrow ((yg_1 \in X_x) \wedge (yg_2 \in X_x) \wedge \ldots (yg_k \in X_x)) \, .$$

Условия (1) и (2) принимают такой вид $\varepsilon k < 1 \quad (1')$ и $2^{p(|x|)}\varepsilon^k < 1 \quad (2')$. Они выполняются при подходящем выборе параметров. Можно взять k порядка $p(|x|)$ и ε порядка $p^{-1}(|x|)$. $\square$

Замечание. Имеется рассуждение, которое “показывает”, что время вычисления функций из класса $\mathcal{BPP}$ по-видимому можно сделать меньше 2^{n^ε} для любого ε. Идея состоит в том, чтобы использовать генераторы *псевдослучайных* чисел. Такой генератор по набору бит длины k строит набор длины k', где $k' \gg k$. При этом если выбирать короткие наборы случайно, то длинные наборы будут распределены так, что вычислительное устройство с ограниченными ресурсами (например, полиномиальная машина Тьюринга) не сможет отличить их от по-настоящему случайных. Это пояснение заменяет точное определение псевдослучайного генератора, которое нам не понадобится.

Существуют ли псевдослучайные генераторы, неизвестно. Если $\mathcal{P} = \mathcal{NP}$, то их заведомо нет. Но в этом случае $\mathcal{P} = \Sigma_2 = \mathcal{BPP}$.

Если же $\mathcal{P} \neq \mathcal{NP}$, то псевдослучайные генераторы, по-видимому, есть. Тогда можно в предикат $R(x,y)$ подставлять вместо случайных y псевдослучайные и существенно сократить время детерминированного вычисления этого предиката.

Остаётся небольшой дефект в этом рассуждении: оно не проходит при $\mathcal{P} \neq \mathcal{NP}$ и отсутствии псевдослучайных генераторов. Такая ситуация считается маловероятной.

2. Класс $\mathcal{PSPACE}$

В этот класс попадают те функции, которые могут быть вычислены на МТ, использующей память, ограниченную полиномом от длины входного слова. Класс $\mathcal{PSPACE}$ можно описать на введённом в начале лекции языке игр.

Теорема. $L \in \mathcal{PSPACE} \iff$ *существует такая игра с полиномиальным по длине входного слова числом ходов, что* $L = \{x \mid$ *Б имеет выигрышную стратегию*$\}$.

Доказательство. $\Longleftarrow$ Покажем, что язык, определяемый игрой, принадлежит $\mathcal{PSPACE}$. Пусть число ходов ограничено $p(|x|)$. Определим такой набор МТ: M_k по заданному началу игры $x, w_1, b_1, \ldots, w_k, b_k$ определяет наличие выигрышной стратегии у Б. Последней в этом ряду машине $M_{p(|x|)}$ нужно просто вычислить предикат $W(x, w_1, \ldots)$. Каждая машина M_k перебирает все возможные варианты $(k+1)$-го хода и консультируется с M_{k+1} по поводу окончательных результатов игры. Её оценка игры составляется очень просто: если текущий ход у Б, то достаточно найти один ход, при котором M_{k+1} гарантирует выигрышную стратегию для Б. Если текущий ход у Ч, то при всех вариантах M_{k+1} должна обнаружить выигрышную стратегию для Б. Машина M_0 определяет наличие выигрышной стратегии для Б в самом начале игры и для её работы нам нужно использовать всю последовательность машин M_k. Но каждая из этих машин занимает небольшую (полиномиально ограниченную) память, так что весь процесс потребует лишь полиномиально ограниченной памяти.

$\Longrightarrow$ Пусть есть $\mathcal{PSPACE}$-машина M, распознающая вхождение слова в язык L. Во-первых, заметим, что вычисление на памяти S бессмысленно проводить дольше, чем время $2^{O(S)}$ (всё начнет повторяться после того, как мы исчерпаем все состояния нашей системы, а их не более чем $2^S \cdot |Q| \cdot |S|$, где Q — множество состояний рассматриваемой МТ). Поэтому можно считать без ограничения общности, что время работы машины M ограничено $2^{O(p(|x|))} = 2^q$.

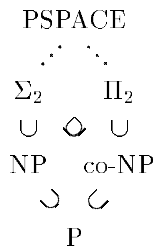
Чтобы было проще описывать игру, потребуем, чтобы после завершения вычисления МТ сохраняла без изменений достигнутое состояние.

Игра заключается в следующем. Белые (которые утверждают, что ответ на входном слове x равен "да") своим первым ходом сообщают состояние машины M (внутреннее состояние, положение читающей головки и строка, записанная на ленте) после 2^{q-1} хода. Черные своим ходом выбирают один из промежутков: от начала до 2^{q-1} хода или от 2^{q-1} хода до конца. Белые указывают состояние M в середине этого промежутка. Далее всё повторяется: Ч выбирает одну из половинок, Б называет состояние M в середине выбранной половинки и т. д.

Игра заканчивается, когда длина промежутка становится 2. Результат определяется так: для обоих концов этого промежутка в процессе игры были названы состояния M. Если состояние правого конца получается из состояния левого конца за один такт работы M, то Б выиграли, иначе выиграли Ч.

Если ответ M на слове x действительно "да", то белым нужно всё время говорить правду, это гарантирует им выигрыш.

Если ответ M на слове x — "нет", то при любом ходе белых на одном из промежутков (или на обоих) обязательно обнаружится ошибка. Ч должен указывать каждый раз именно этот промежуток. $\square$



Теперь можно замкнуть диаграмму классов, нижняя часть которой была приведена в начале лекции (см. слева). Внизу располагается класс $\mathcal{P}$, отвечающий играм с 0 ходов, затем идут дополняющие друг друга классы, отвечающие играм с конечным числом ходов (для одного хода это $\mathcal{NP}$ и co-$\mathcal{NP}$, для двух ходов — Σ_2 и Π_2 и т. д.). Завершается эта диаграмма классом $\mathcal{PSPACE}$, который определяется произвольными играми с одним естественным условием — время игры должно быть полиномиально ограничено размером входного слова.

В классе $\mathcal{PSPACE}$ существуют полные задачи (относительно полиномиальной сводимости). Простейший вариант получается применением предыдущей теоремы.

Задача TQBF. $TQBF(x) \Leftrightarrow$ x есть истинная булева формула с кванторами (True Quantified Boolean Formula); т. е. формула вида $\mathsf{Q}_1\, p_1 \dots \mathsf{Q}_n\, p_n F(p_1, \dots, p_n)$, где F — некоторая логическая формула, а Q_i — либо $\forall$, либо $\exists$. По определению, $(\forall\, p_1\, A(p_1)) \stackrel{\text{def}}{=} (A(true) \wedge A(false))$, а $(\exists\, p_1\, A(p_1)) \stackrel{\text{def}}{=} (A(true) \vee A(false))$,

Теорема. TQBF $\mathcal{PSPACE}$*-полна.*

Доказательство. Построим сведéние любого языка $L \in \mathcal{PSPACE}$ к задаче TQBF. Для этого превратим МТ, вычисляющую результат игры (предикат $W(\cdot)$), в схему, а ходы игроков закодируем булевыми переменными. Тогда наличие выигрышной стратегии у белых задаётся условием

$$\exists w_1^1 \exists w_1^2 \dots \exists w_1^{p(|x|)} \forall b_1^1 \dots \forall b_1^{p(|x|)} \dots S(x, w_1^1, w_1^2, \dots),$$

где $S(\cdot)$ обозначает результат вычисления по схеме.

Чтобы превратить S в булеву формулу, добавим новые переменные y_i (значение, вычисленное при i-том присваивании в схеме) и заменим $S(\cdot)$ на формулу вида

$$\exists y_1, \dots y_{\text{размер схемы}}\, (y_2 \Leftrightarrow (\text{первое присваивание})) \wedge \dots \wedge (y_2 \Leftrightarrow (\text{последнее присваивание}))\,.$$

После этой подстановки получим квантифицированную булеву формулу, истинную тогда и только тогда, когда $x \in L$. □

3. Содержание дальнейшей части курса

Далее будут квантовые вычисления. Квантовый компьютер очень похож на классический. Но производятся операции не с битами, а с *квантовыми состояниями* системы битов. МТ, работающую с памятью размера n, можно описать как автомат на множестве состояний $\{0,1\}^n$. Теперь будем считать, что (0,1)-векторы индексируют ортонормированный базис в пространстве $\mathbb{C}^{2^n}$, снабженном скалярным произведением. Каждый такт вычисления будем представлять унитарным преобразованием этого пространства, имеющим специальную структуру. Если представить $\mathbb{C}^{2^n}$ как тензорную степень $\left(\mathbb{C}^2\right)^{\otimes n}$, то допустимые преобразования представляются как тензорные произведения оператора, действующего на двух сомножителях и тождественного оператора на остальных сомножителях.

Результат вычисления будет определяться с некоторой вероятностью. Класс $\mathcal{BQP}$ функций, вычислимых на квантовом компьютере за полиномиальное время с ограниченной ошибкой, определяется аналогично классу $\mathcal{BPP}$. По всей видимости, класс $\mathcal{BQP}$ не совпадает с уже известными классами. Пока можно лишь утверждать (это будет позже доказано), что $\mathcal{BPP} \subseteq \mathcal{BQP} \subseteq \mathcal{PSPACE}$.

Есть примеры (на самом деле, пример) задач, скорее всего сложных для обычных компьютеров, но решаемых за полиномиальное время на квантовом компьютере. Примером может служить задача разложения на множители, для которой есть только экспоненциальные алгоритмы в классическом случае и кубический алгоритм в квантовом.

Реализация квантового компьютера "в железе" является крайне сложной задачей, решения которой трудно ожидать в ближайшие 10 лет. Одна из возникающих трудностей состоит в том, что нужно научится проводить сколь угодно длинные вычисления на квантовых компьютерах с конечной вероятностью ошибки (т. е. вместо требуемого оператора U, такое устройство вычисляет некоторый оператор $\tilde{U}$, где $\|\tilde{U} - U\| < \delta$).

Оказывается, и об этом будет рассказано, что достаточно ограничиться конечной величиной ошибки δ, если использовать квантовые аналоги кодов, исправляющих ошибки.