

# Глава 1

## Конечные игры и класс $PH$ .

### 1.1 Конечные игры.

Играют два игрока (A) и (M). Начальная конфигурация игры задается входным словом  $x \in \{0, 1\}^*$ . Правила игры специфицируются следующими параметрами:

- фиксированным (конечным) количеством ходов  $k$  и указанием, кто начинает;
- полиномом  $p$ , задающим длину одного хода как функцию от  $|x|$ ; ход – это слово в алфавите  $\{0, 1\}$  длины  $p(|x|)$ ;
- предикатом  $R \in P$ , задающим условие выигрыша (M) при данной последовательности ходов  $w_1, b_1, w_2, b_2, \dots$ ,

$$R = R(x, w_1, b_1, w_2, b_2, \dots).$$

(Количество аргументов фиксировано, оно на единицу больше числа ходов.)

Игроки по очереди обмениваются ходами. Каждый ход – слово длины  $p(|x|)$ . После завершения партии с помощью предиката  $R$  вычисляется, кто выиграл. Такая игра называется *конечной игрой* или *ограниченным детерминированным интерактивным протоколом*.

Совокупность всех вариантов розыгрыша, начинающихся с данной начальной конфигурации  $x$ , удобно представлять себе в виде *дерева игры*. Это дерево порядка ветвления  $2^{p(|x|)}$ , вершины которого соответствуют всевозможным “промежуточным” конфигурациям, т.е. состояниям игры, возникающим после  $i$  произвольных ходов,  $i \leq k$ .

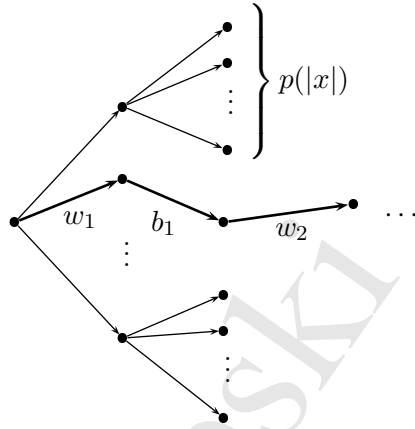


Рис. 1.1: Дерево игры.

## 1.2 Определение класса $PN$ .

Пусть начальная конфигурация  $x$  фиксирована. Множество всевозможных партий конечно, поэтому для одного из игроков обязательно существует выигрышная стратегия – функция, определяющая его очередной ход по всем предыдущим ходам так, что если руководствоваться ей при выборе ходов, то игрок обязательно выиграет.

**Пример.** Если ходов 2 и начинает (А), то выигрышная стратегия  $F$  для (М) – это функция, которая по каждому возможному ходу  $w_1$  игрока (А) вычисляет ответный ход  $b_1$  для игрока (М). Она должна удовлетворять условию  $\forall w_{|w|=p(|x|)} R(x, w, F(w))$ . Такая функция существует, если

$$\forall w_{|w|=p(|x|)} \exists b_{|b|=p(|x|)} R(x, w, b).$$

Для той же игры выигрышная стратегия для (А) – это функция, которая по пустому слову (игрок (А) ходит первым) вычисляет правильный первый ход. Она должна удовлетворять условию  $\forall b_{|b|=p(|x|)} \neg R(x, F(\cdot), b)$ . Выигрышная стратегия для (А) существует, если

$$\exists w_{|w|=p(|x|)} \forall b_{|b|=p(|x|)} \neg R(x, w, b).$$

Легко видеть, что условия существования этих двух стратегий взаимно дополняющие.

**Определение 1.1** Язык  $L \in \{0, 1\}^*$  распознается игрой, если условие “ $x \in L$ ” равносильно “для начальной конфигурации  $x$  игрок (М) обла-

дает выигрышной стратегией”. Семейство всех таких языков образует класс  $PН$ .

**Замечание.** Обозначение для класса  $PН$  происходит от термина полиномиальная иерархия (Polynomial Hierarchy), обсуждаемого в следующей лекции. Собственно полиномиальная иерархия есть набор сложных классов, структурирующих класс  $PН$ , а элементы  $L \in PН$  – это все языки, сложность которых может быть измерена с помощью полиномиальной иерархии.

**Интерпретация.** Игроки - юный Артур и его учитель Мерлин. Великий Мерлин обладает неограниченными вычислительными возможностями, в частности, может вычислять выигрышные стратегии, когда они существуют. Он знает, что  $x \in L$  и хочет передать это знание Артуру. Возможности Артура меньше – доступные ему вычисления полиномиально ограничены по времени. При этом он не вполне доверяет Великим и хочет получить не только сам факт, но и некоторое доступное ему подтверждение. Но он – Избранный и обычно добивается успеха, если только это возможно. За него фактически играет Провидение, которое не слабее Мерлина. Собственных сил Артуру хватает лишь на то, чтобы проследить за правильностью присуждения выигрыша после окончания игры. В такой ситуации (пока Провидение не покинуло Артура) в качестве подтверждения Мерлину оказывается достаточным выиграть у Артура в соответствующей игре, хотя на самом деле Артур получает знание

“если он все еще Избранный, то  $x \in L$ ”

и вынужден вечно сомневаться.

Например, они изучают тактику захвата замков. Начальной конфигурацией игры служит план замка. Предполагается, что способов защиты и нападения экспоненциально много. Мерлин имитирует защитников и старается убедить Артура, что замок неприступен. Он расставляет силы защитников, а Артур предлагает план атаки, после чего разыгрывается сражение (за полиномиальное время). Получается двухходовая игра, распознающая предикат “замок  $x$  неприступен”.

**Замечание.** Две игры будем считать эквивалентными, если они распознают один и тот же язык. Легко видеть, что за счет добавления одного хода в начале можно преобразовать любую ограниченную игру в эквивалентную, в которой начинает заданный игрок, например, (M). Соответствующий предикат выигрыша просто будет зависеть от добавленного хода фиктивно. Аналогично можно управлять выбором последнего

хода. Можно также при необходимости повысить степень полинома, задающего длину хода, и/или количество ходов, т.к. лишние биты может отбрасывать “сам” предикат выигрыша.

**Замечание.** Следующие модификации правил игры легко моделируются исходными (за счет модификации предиката выигрыша): (а) длина хода не в точности равна, а не превосходит величины  $p(|x|)$ ; (б) количество ходов не фиксировано, а зависит от входного слова, но ограничено сверху константой. Более точно это означает, что если язык  $L$  распознается модифицированной игрой, то существует стандартная конечная игра, которая также распознает  $L$ . Идеи моделирования таковы:

- (а) Длина хода в соответствующей стандартной игре будет  $n^{c+1}$ , где  $c$  есть степень многочлена  $p$ . Ход  $v = v_1 \dots v_k \in \{0, 1\}^*$ ,  $|v| \leq p(|x|)$  модифицированной игры моделируется в стандартной игре ходом

$$v_1 v_1 \dots v_k v_k \underbrace{0100 \dots 0}_{|x|^{c+1} - 2k - 2} .$$

“Стандартный” предикат выигрыша сначала по своим аргументам восстанавливает последовательность ходов модифицированной игры, а затем вычисляет значение “модифицированного” предиката выигрыша на этой последовательности.

- (б) При такой модификации предикат выигрыша имеет два аргумента – начальную конфигурацию  $x$  и последовательность ходов  $w_1 b_1 w_2 b_2 \dots$ , записанную одним словом без разделителей. Его значение вычисляется после каждого хода, а не только в конце партии. Окончание партии происходит, когда значение предиката выигрыша после очередного хода стало 1 или когда число ходов достигло максимального, допустимого для данной игры ( $k$ ). В соответствующей стандартной игре все партии – по  $k$  ходов. “Стандартный” предикат выигрыша определяется следующим образом:

$$\underbrace{R(x, w_1) \vee R(x, w_1 b_1) \vee R(x, w_1 b_1 w_2) \vee \dots}_{k \text{ раз}} .$$

### 1.3 Замкнутость относительно $\cap$ , $\cup$ и $(\cdot)^c$ .

**Лемма 1.2** Если  $L_1, L_2 \in PH$ , то  $L_1 \cap L_2, L_1 \cup L_2, L_1^c \in PH$ .

**Доказательство.** Случай  $\cap$ . Для языков  $L_1, L_2$  выберем распознающие их игры так, чтобы длины ходов в них были одинаковыми, а последний ход первой делал не тот игрок, который начинает вторую. Последовательность ходов для игры, распознающей  $L_1 \cap L_2$ , будет состоять из последовательности ходов первой игры  $\bar{w}_1$ , которая продолжается последовательностью ходов второй игры  $-\bar{w}_2$ . В качестве предиката выигрыша следует взять  $R_1(x, \bar{w}_1) \wedge R_2(x, \bar{w}_2)$ , где  $R_1, R_2$  – предикаты выигрыша первой и второй игр.

Случай  $\cup$ . Для языков  $L_1, L_2$  выберем распознающие их игры так, чтобы у них совпадали длины и количества ходов, причем в обеих играх первый ход делал (А). Количество ходов в игре для  $L_1 \cup L_2$  будет на единицу больше, а дополнительный первый ход  $z$  будет делать (М). Предикат выигрыша  $R(x, z, \bar{w})$  будет

$$( (z = 0 \dots 0) \wedge R_1(x, \bar{w}) ) \vee ( (z \neq 0 \dots 0) \wedge R_2(x, \bar{w}) ).$$

Случай дополнения. Достаточно передать право первого хода другому игроку, а в качестве предиката выигрыша взять  $\neg R_1(x, \bar{w})$ . ■

V. Krupski  
COMPLETELY  
Lecture Notes, draft

## Глава 2

# Полиномиальная иерархия.

### 2.1 Классы полиномиальной иерархии

Условимся писать  $\exists^p z$  вместо  $\exists z_{|z|=p(|x|)}$  и  $\forall^p z$  вместо  $\forall z_{|z|=p(|x|)}$ .

**Определение 2.1** Рассмотрим логическую структуру условия существования выигрышных стратегий для (М) в  $n$ -ходовой игре (т.е. условия “ $x \in L$ ”). Если в игре начинает (М), то оно имеет вид

$$\exists^p w_1 \forall^p b_1 \exists^p w_2 \dots R(x, w_1, b_1, w_2, \dots), \quad R \in P, \quad (2.1)$$

где количество кванторов равно  $n$ . Такие игры (полином  $p$  и предикат  $R \in P$  – любые) называются  $\Sigma_n^p$ -играми, а соответствующие языки образуют класс полиномиальной иерархии  $\Sigma_n^p$ . Для игры, в которой начинает (А), соответствующее условие будет

$$\forall^p w_1 \exists^p b_1 \forall^p w_2 \dots R(x, w_1, b_1, w_2, \dots), \quad R \in P \quad (2.2)$$

с кванторной приставкой длины  $n$ . Эти игры называются  $\Pi_n^p$ -играми, а соответствующие языки составляют класс  $\Pi_n^p$ .

**Замечание.** Само понятие игры в определении классов полиномиальной иерархии формально не является необходимым – класс  $\Sigma_n^p$  состоит из всех предикатов вида (2.1), а класс  $\Pi_n^p$  – вида (2.2), где  $n$  есть длина кванторной приставки. Заметим также, что верхний индекс  $p$  в обозначениях классов происходит от слова “polynomial” и не обозначает конкретный полином. Например, в класс  $\Sigma_n^p$  попадают все предикаты вида (2.1) для всевозможных полиномов  $p(n) = n^c$ ,  $c = 1, 2, \dots$

**Замечание.** Эквивалентные логические описания этих классов получатся также в следующих случаях:

- если вместо кванторов  $\exists^p, \forall^p$  использовать (всюду или в некоторых местах)  $\exists z_{|z| < p(|x|)}, \forall z_{|z| < p(|x|)}$ ;
- если разрешить в формулах (2.1), (2.2) несколько одноименных кванторов подряд (в этом случае параметр  $n$  есть количество чередований кванторов);
- и то и другое одновременно.

**Замечание.** Для доказательства принадлежности языка  $L$  классу полиномиальной иерархии достаточно представить предикат “ $x \in L$ ” в виде (2.1) или (2.2) не для всех, а только для достаточно длинных слов  $x$ ,  $|x| > n_0$ . Такое “неполное” представление легко переделать в “полное”, переопределив значение предиката  $R$  для коротких слов  $x$  следующим образом:

$$R(x, \dots) := \begin{cases} 1, & \text{если } |x| \leq n_0 \text{ и } x \in L, \\ 0, & \text{если } |x| \leq n_0 \text{ и } x \notin L. \end{cases}$$

## 2.2 Структурные свойства классов полиномиальной иерархии.

Структурные свойства классов  $\Sigma_n^p$  и  $\Pi_n^p$  собраны в следующей лемме и отражены на Рис.2.1

**Лемма 2.2** (*Структура РН.*)

- $\Pi_n^p = \text{co-}\Sigma_n^p$ ;
- $\Sigma_n^p \cup \Pi_n^p \subset \Sigma_{n+1}^p \cap \Pi_{n+1}^p$ ;
- $\Sigma_0^p = \Pi_0^p = P$ ;  $\Sigma_1^p = NP$ ;
- $\bigcup_n (\Sigma_n^p \cup \Pi_n^p) = \bigcup_n \Sigma_n^p = \bigcup_n \Pi_n^p = PH$ .

**Доказательство.** Непосредственное следствие определений. ■

**Замечание.** В лемме речь идет про нестрогое включение. Про любую пару классов полиномиальной иерархии (кроме  $\Sigma_0^p = \Pi_0^p$ ) неизвестно, совпадают они или нет. В настоящее время наиболее правдоподобной представляется гипотеза о невырожденности полиномиальной иерархии, утверждающая, что все эти классы различны. Из нее, в частности, вытекает, что  $P \neq NP$ .

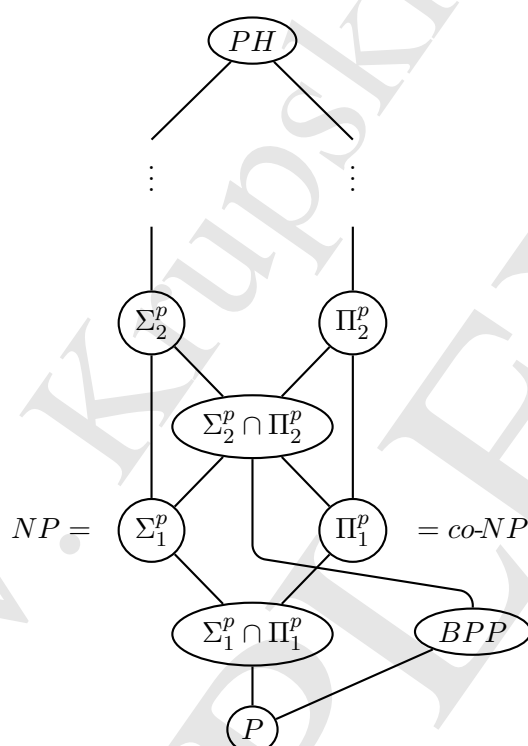
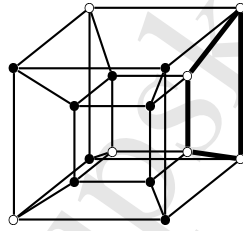


Рис. 2.1: Структура классов полиномиальной иерархии.

### 2.3 Пример.

Рассмотрим следующий пример использования полиномиальной иерархии для получения (верхней) оценки сложности конкретной задачи.

**Задача.** Вершины единичного  $2n$ -мерного куба раскрашены в два цвета – белый и черный. По данной раскраске выяснить, имеется ли в кубе  $n$ -мерная грань, все вершины которой – белые.



Координаты вершин единичного куба – числа 0 или 1. Для задания раскраски уместно использовать булеву формулу  $\varphi(x_1, \dots, x_{2n})$ , для которой

$$\varphi(x_1, \dots, x_{2n}) = 1 \Leftrightarrow \text{вершина } (x_1, \dots, x_{2n}) \text{ – белая.}$$

Будем дополнительно предполагать, что все  $2n$  переменных встречаются в формуле и ее размер не превосходит фиксированного полинома от  $n$ .

Рассмотрим язык  $L$ , состоящий из всех представленных таким образом раскрасок, для которых белая  $n$ -мерная грань существует. Покажем, что уточненная так задача попадает в класс  $\Sigma_2^p$ , т.е. что  $L \in \Sigma_2^p$ .

Двухходовая игра класса  $\Sigma_2^p$ , которая распознает язык  $L$ , такова. Своим первым ходом, (М) предлагает систему уравнений

$$\begin{cases} x_{i_1} = a_1, \\ \vdots \\ x_{i_n} = a_n, \end{cases} \quad (2.3)$$

которая, по его мнению, задает белую  $n$ -мерную грань. Ответный ход (А) состоит в выборе значений для всех остальных координат  $x_j$ ,  $j \neq i_1, \dots, i_n$ . После пары ходов вычисляется значение  $\varphi(x_1, \dots, x_{2n})$ . Если оно равно 1, то выигрывает (М); в противном случае выигрывает (А).

Для (М) единственный способ (стратегия) обеспечить себе гарантированный выигрыш состоит в указании системы уравнений, которая на

самом деле задает белую грань. Это возможно тогда и только тогда, когда такая грань существует. Таким образом, эта игра действительно распознает язык  $L$ .

**Замечание.** Следует иметь в виду, что другие уточнения постановки задачи могут оказаться неэквивалентными рассмотренной, а потому иметь совершенно другую сложность. Например, если раскраски подавать на вход распознающему алгоритму в виде списка всех черных вершин (вершины задаются своими координатами), то очевидным улучшением оценки будет класс  $NP = \Sigma_1^P$ . Недетерминированному полиномиально ограниченному по времени допускающему алгоритму достаточно “угадать” систему уравнений (2.3), задающую белую  $n$ -мерную грань, а затем (детерминировано) проверить, что ни одна из вершин списка не принадлежит этой грани. В случае двухходовой игры похожая проверка имитировалась ходом (А), когда фактический перебор осуществляло Провидение, а (А) лишь оглашал самый худший для (М) вариант. Здесь же перебор заменяется однократным просмотром исходных данных, поэтому проверку может осуществить алгоритм, затратив на это время, ограниченное полиномом от их длины. (При аккуратной организации хранения системы (2.3) можно добиться линейной оценки времени.)

## 2.4 Включение $BPP \subset \Sigma_2^P \cap \Pi_2^P$ .

**Теорема 2.3**  $BPP \subset \Sigma_2^P \cap \Pi_2^P$ .

**Сведение.** Учитывая дополняемость классов  $\Sigma_2^P$  и  $\Pi_2^P$  достаточно установить, что для каждого языка  $L \in BPP$  выполняется  $L \in \Sigma_2^P$  и  $L^c \in \Sigma_2^P$ . Но так как сам класс  $BPP$  замкнут относительно дополнений, то остается установить следующий факт:

**Теорема 2.4**  $BPP \subset \Sigma_2^P$ .

**Доказательство.** Пусть язык  $L \in BPP$  и  $(R, p, \alpha)$  – его частотный распознаватель – первый из построенных в лемме ??, т.е.

$$p(n) = n^{d+1}, \quad \alpha(n) = \lambda^n, \quad 0 < \lambda < 1.$$

Для достаточно длинных слов  $x$  надо представить условие “ $x \in L$ ” в виде эквивалентного, имеющего форму

$$\exists^q w \forall^q b Q(x, w, b),$$

где  $q$  – некоторый полином, а предикат  $Q$  принадлежит классу  $P$ . Вместо этого мы построим условие вида

$$\exists^p g_1 \dots \exists^p g_k \forall^p g Q'(x, g_1, \dots, g_k, g), \quad Q' \in P,$$

где величина  $k$  есть полином от длины  $x$ . Его легко свести к требуемому:  $q(n) := k \cdot p(n)$ ,  $w := g_1 \dots g_k$ ,  $b := g_0 \dots 0$ , а предикат  $Q$  “сам” вычисляет  $k$  и разрезает  $w$  и  $b$  на нужные куски.

Обозначим через  $G = G(x)$  множество всех двоичных слов длины  $p(|x|)$ . Тогда кванторы  $\exists^p$  и  $\forall^p$  есть просто кванторы по переменным, пробегающим  $G$ . Введем на  $G$  структуру абелевой группы с операцией  $+$  – поразрядным сложением по модулю 2. Обозначим

$$Y = Y(x) = \{y \in G \mid R(x, y)\}$$

множество всех “свидетелей”, используемых частотным распознавателем для проверки условия “ $x \in L$ ”. Мы покажем, что при подходящем выборе параметров искомое представление есть

$$x \in L \Leftrightarrow (\exists g_1 \in G) \dots (\exists g_k \in G) (\forall g \in G) \bigvee_{i=1}^k (g \in g_i + Y). \quad (2.4)$$

Сначала убедимся, что (при условии полиномиальности  $k$ ) в правой части (2.4) под кванторами стоит предикат из класса  $P$ :

$$Q'(x, g_1, \dots, g_k, g) \Leftrightarrow \bigvee_{i=1}^k (g \in g_i + Y) \Leftrightarrow \bigvee_{i=1}^k R(x, g - g_i).$$

Теперь займемся выбором параметров, обеспечивающим верность (2.4) и полиномиальность  $k$ . Правая часть (2.4) эквивалентна условию

$$\exists g_1, \dots, g_k \in G \left( \bigcup_{i=1}^k (g_i + Y) = G \right), \quad (2.5)$$

утверждающему, что с помощью  $k$  сдвигов множества  $Y$  можно покрыть всю группу  $G$ . Естественно ожидать, что это условие выполнено, если доля  $\delta = |Y|/|G|$  достаточно велика, и не выполнено, если мала.

**Лемма 2.5** *Если  $k \cdot \delta < 1$ , то условие (2.5) не выполнено. Если  $|G| \cdot (1 - \delta)^k < 1$ , то условие (2.5) выполнено.*

**Доказательство.** (Специфика выбора группы не используется.) Легко видеть, что

$$\left| \bigcup_{i=1}^k (g_i + Y) \right| \leq k \cdot \delta \cdot |G|$$

откуда следует первое утверждение.

Для доказательства второго утверждения достаточно установить, что (при указанных условиях) если в качестве  $g_i$  взять значения, полученные в результате независимых испытаний случайной величины, принимающей произвольные значения из  $G$  с равными вероятностями  $1/|G|$ , то

$$Prob\left\{ \bigcup_{i=1}^k (g_i + Y) = G \right\} > 0.$$

Оценим вероятность противоположного события, учитывая, что  $|g_i + Y| = |Y|$ , а события " $g \notin (g_i + Y)$ " для фиксированного  $g$  и различных  $i$  независимы:

$$Prob\left\{ \bigvee_{g \in G} \left( g \notin \bigcup_{i=1}^k (g_i + Y) \right) \right\} \leq |G| \cdot (1 - \delta)^k < 1.$$

В нашем случае величина  $\delta$  зависит от входного слова  $x$  и оценивается функцией от его длины  $n = |x|$ :

$$\begin{aligned} x \notin L &\Rightarrow \delta < \lambda^n &\Rightarrow k \cdot \delta < k \cdot \lambda^n \\ x \in L &\Rightarrow \delta > 1 - \lambda^n &\Rightarrow |G| \cdot (1 - \delta)^k < 2^{p(n)} \cdot \lambda^{k \cdot n} \end{aligned}$$

Положим  $k = p(n)$ . При достаточно больших  $n$  будет  $k \cdot \lambda^n < 1$  и  $2^{p(n)} \cdot \lambda^{k \cdot n} < 1$ . По лемме 2.5 отсюда следует, что эквивалентность (2.4) справедлива для всех достаточно длинных слов  $x$ .