

## Н1 теория сложности

### literature:

- Arora Barak "Complexity Modern Approach" (1st part)
- Garry Johnson "Трудно разрешенные задачи"
- site: compendium of NP-complete problems

### outline:

- NP-полнота
  - Концепция недетерминированных вычислений
- Сведения
  - Теорема Кука-Левина
- язык CNFSAT
  - Теорема  $CNFSAT \in NPC$
  - Теорема  $CNFSAT \rightarrow 3SAT$
- Теорема  $IND \in NPC$
- диагональный метод
  - теоремы об иерархии
    - Теорема о ёмкости иерархии
    - Теорема о временной иерархии
  - Теорема Бэйкера-Гилла-Соловья (BGS)
  - Теорема Ладнера
- coNP
- PSPACE и PSPACE полнота
  - $TQBF \in PSC$
  - Теорема  $NSPACE(f(n)) \subset DSPACE(f(n)^2)$ 
    - Следствие. Теорема Сэвитча

## Н2 NP-полнота

Характеристики сложности вычисления.

Есть распознаватели ( $\Sigma^* \rightarrow B$ ) и преобразователи ( $\Sigma^* \rightarrow \Sigma^*$ )

- время:  $T(n) = O(f(n))$
- память:  $S(n)$
- random:  $R(n)$

$DTIME(f) = \{L \mid \exists \text{ program } p :$

1.  $x \in L \implies p(X) = 1, x \notin L \implies p(x) = 0$

2.  $n = |x| \implies T(p, x) = O(f(n))\}$

$h = (01)^* \in DTIME(n)$

$\widetilde{DTIME}(f) = \{h \mid \dots\}$

палиндромы:  $Pal \in DTIME_{RAM}(n)$

$Pal \notin DTIME_{TM}(n)$

$P = \cup_{f-polynom} DTIME(f) = \cup_{i=0}^{\infty} DTIME(n^i)$

$$p(n)q(n) : p + q, p * q, p(q(n))$$

$$L_1 L_2 \in P : L_1 \cup L_2 \in P, L_1 \cap L_2 \in P, \overline{L_1} \in P, L_1 L_2 \in P, L_1^* \in P$$

### НЗ концепция недетерминированных вычислений

Допускается  $\iff \exists$  последовательность переходов, которая приводит к допуску  
 недетерминирования программа  $p(x)$  допускает  $\iff \exists$  последовательность  
 недетерминированных выборов, приводящая к допуску  
 $p(x)$  не допускает  $\iff \forall$  последовательности выборов не допуск

**def**  $NTIME(f) = \{L \mid \exists \text{ недетерминированная программа } p$   
 1)  $p(x) - acc \iff x \in L$ ; 2)  $T(p, x) = O(f(n))\}$

**ex** задача о гамильтоновом цикле

```
p(G)
vis[1..n]: arr of bool
s = 1
for i = 1..n
  u = ?{1..n}
  if (vis[u]) return false
  if (su not in EG) return false
  vis[u] = true
  s = u
if (s ≠ 1) return false
return true
```

**ex**  $isComposite(z)$ ,  $n = \lceil \log_B z \rceil$ , где  $B$  - это основание системы счисления

```
a = ?{2..z-1} // T = logn
if z % a = 0 // poly(logn)
  return true
return false
```

Нельзя свопнуть ветки и сделать проверку на простоту, потому что это `true` и `false` не симметричны в недетерминированных вычислениях (нельзя даже `isPrime(n): return !isComposite(n)`)

**def**  $NP = \cup_{f-polynome} NTIME(f)$ , *nondeterministic polynomial*  
**stat**  $P \subset NP$

?  $P = NP$

*неформально*: класс  $P$  - класс задач, которые можно решить за полином, класс  $NP$  - класс задач, решение которых можно проверить за полином

$\Sigma_1$  - класс языков, в которых можно формализовать класс решения, которое можно проверить за полином

$\Sigma_1 = \{L \mid \exists \text{ полином } p, \text{ работающая за полином программа } R(x, y) - \text{детерминированная}\}$

$x \in L \iff \exists y$  (называют *сертификат*):  $|y| \leq p(|x|)$  and  $R(x, y) = 1$   
 $x \notin L \implies \forall y (|y| \leq p(|x|)) R(x, y) = 0\}$

**ex** гамильтонов цикл  $Ham \in \Sigma_1$

```

R(G, y):
  y as arr[1..n] of int
  // we can add: y = ?arr[i..n] of {1..n} // O(n)
  vis = arr[1..n] of bool
  for i = 1..n
    if (y[i] y[i mod n+1] not in EG) return false
    if vis[y[i]] return false
    vis[y[i]] = true
  return true

```

Th  $NP = \Sigma_1$

$L \in NP, L \in \Sigma_1$

неформально: NP – определение на языке недетерминированных формат,  $\Sigma_1$  – определение на языке сертификатов

## Н2 сведения

**def сводим В к А по Тьюрингу:** А, В – языки, С – сложностный класс,  $B \in C^A$  (С с оракулом А). не считая вызова функции `isInA(x): Bool`, остальные ограничения класса С учитываются.

**def сведение по Куку-Левину** (Тьюрингу за полином)  $B \in P^A$

**def сведене по Карпу** (m-сведение): язык В сводится к А ( $B \leq A$ ), если  $\exists$  вычислимая за полином функция f такая, что  $x \in B \iff f(x) \in A$

**ex**  $IND = \{\langle G, k \rangle \mid \text{в } G \text{ независимое множество размера } k\}$

$CLIQUE = \{\langle G, k \rangle \mid \text{в } G \exists \text{ клика размера } k\}$

$IND \leq CLIQUE$

$f(\langle G, k \rangle) = \langle \bar{G}, k \rangle$  // за полином

в G и множестве размера k  $\iff$  в  $\bar{G}$   $\exists$  клика размера k

$VCOVER = \{\langle G, k \rangle \mid \text{в } G \exists \text{ вершинное покрытие размера } k\}$

$IND \leq VCOVER$

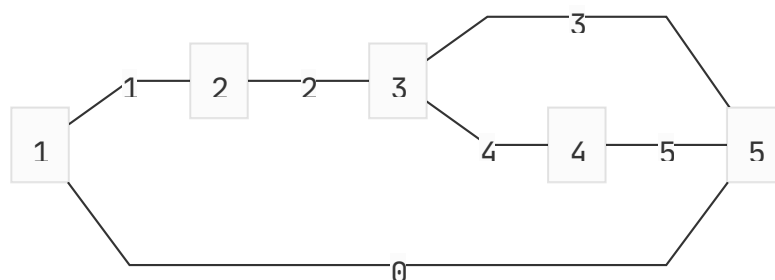
$f(\langle G, k \rangle) = \langle G, n - k \rangle$ , где n - число вершин G

**ex**  $SUBSETSUM = \{\langle [x_1, x_2, \dots, x_n], s \rangle \mid \exists I \subset \{1, 2, \dots, n\}, \sum_{i \in I} x_i = s, x_i \in \mathbb{N}\}$

`dp[i][w]` - можно ли первые i  $\Sigma = w // w - 2^{|s|}$

$VCOVER \leq SUBSETSUM$

пронумеруем вершины с единицы, рёбра – с нуля, битовыми масками каждой вершине сопоставляем рёбра



	6	5	4	3	2	1	0
$x_1$	1	0	0	0	0	1	1
$x_2$	1	0	0	0	1	1	0
$x_3$	1	0	1	1	1	0	0
$x_4$	1	1	1	0	0	0	0
$x_5$	1	1	0	1	0	0	1
s	3	2	2	2	2	2	2

$$x_6 = 1$$

$$x_7 = 10$$

$$x_8 = 100$$

$$x_9 = 1000$$

$$x_{10} = 10000$$

$$x_{11} = 100000$$

$f(\langle G, k \rangle)$ ,  $n$  - число вершин,  $m$  - число рёбер,  $s = k22\dots 2$ ,  $m$  двоек

$f$  сводит VCOVER к SUBSETSUM

$\Rightarrow$ : в  $G \exists$  вершинное покрытие размера  $k$

$\Leftarrow$ :  $[x_1 \dots x_{n+n}]$ ,  $s \exists$  решение  $\Rightarrow$  в  $G \exists$  вершинное покрытие размера  $k$

**def** язык называется **NP-hard** (NP-трудный), если выполнены следующие условия:

$$\forall B \in NP : B \leq A$$

**def**  $A$  называется **NP-complete** (NP-полный), если:

$$1) A \in NPH$$

$$2) A \in NP$$

$$// NPC = NPH \cap NP$$

**ex**  $BH_{1N}$  (bounded halting unary nondeterministic)

$BH_{1N} = \{ \langle m, x, 1^t \rangle \mid m - \text{недетерминированная машина Тьюринга, } x - \text{вход, } t -$

ограничение времени:  $\exists$  последовательность недетерминирования выборов машины

Тьюринга  $m$ , что она допускается за  $t$  шагов:  $\{ m(x) = 1 \}$

**Th**  $BH_{1N} \leq NPC$

$$1. BH_{1N} \in NPH$$

$$A \in NP$$

// def по Карпу

$m_A$  - недетерминированная машина Тьюринга, решающая  $A$  за полином  $p(n) = cn^k$

$$f(x) = \langle m_A, x, q^{p(|x|)} \rangle$$

$$x \in A \iff \exists \text{ последовательность выборов } m_A(x) = 1 \text{ (за } p(|x|) \text{)}$$

$$2. BH_{1N} \in NP$$

$$L A \leq^k B, B \leq^k C \implies A \leq^k C$$

$$x \xrightarrow{t} f(x) \xrightarrow{t} g(f(x))$$

$$\text{con } A \in NPH, A \leq B \implies B \in NPH$$

**stat** если  $B \leq A, A \in NPH$

$$NP \xrightarrow{t} BH_{1N} \xrightarrow{t} SAT$$

$$\text{def } SAT = \{ \phi(x_1 \dots x_n) \mid \exists x_1 \dots x_n \phi(x_1 \dots x_n) = 1, \phi - \text{б ф} \}$$

$$SAT \in NPC$$

$$BH_{1N} \leq SAT$$

$$\langle m, x, 1^t \rangle \xrightarrow{f} \phi$$

$\phi$  удовлетворяет  $\iff \exists$  последовательность недетерминированных выборов

$$m(x) = 1, \text{ за время } t$$

больше  $t$  шагов не будет, есть мгновенные описания машины  $\alpha \#_q \beta$

дополним описания до длины  $t + 1$

$$q_0 \vdash q_1 \vdash \dots \vdash q_t$$

табло вычислений: первая строка - стартовое состояние,  $i \rightarrow i + 1, q_i \vdash q_{i+1}$ , допуск:

последовательность до  $\#_{acc}$

$$\langle m, x, 1^t \rangle \in BH_{1N} \iff \exists \text{ допускающее табло вычислений}$$

количество состояний  $|Q| = z$ , множество ленточного алфавита  $|PT| = y, z + y = k$

заведём  $(t + 1)^2 k$  переменных,  $x_{ijc}$  – верно ли, что в табло в  $i$ -й  $j$ -й ячейке записан символ 'с'

$$\phi(x_{ijc}) = C \wedge S \wedge T \wedge N$$

$$C = \bigwedge i, j = 0..t \vee_C ((\bigwedge \neg X_{ij\alpha}) \wedge X_{ijc})$$

$$S = X_{00\#s} \wedge X_{01x_1} \wedge X_{02x_2} \wedge \dots \wedge X_{0nx_n} \wedge X_{0(n+1)B} \wedge \dots$$

$$T = X_{t0\#x} \vee X_{t1\#y} \vee \dots \vee X_{tt\#y}$$

$$N = (\bigwedge_{i,j} \bigwedge_{c_1 c_2 c_3 c_4 \notin Q} X_{i-1,j-1,c_1} \wedge X_{i-1,j,c_2} \wedge X_{i,j+1,c_3} \wedge X_{i,j,c_4} \rightarrow c_1 = c_4) \wedge_{ijx} \bigwedge_{c_1 \dots c_6 \dots}$$

допустимы

qed  $\square$

## Н2 язык CNFSAT

def **CNFSAT** =  $\{\phi \mid \phi \text{ в КНФ}, \phi \in SAT\}$

$$(x_i \vee \neg x_j \dots) \wedge (\vee \vee \vee) \wedge (\vee)$$

clause (клез)

ex 2-SAT (ровно две) HornSAT (не более одной без отрицания)

## Н3 Th CNFSAT in NPC

$$1. CNFSAT \in NP$$

$$2. CNFSAT \in NPH$$

$$SAT \leq CNFSAT$$

$$\phi \xrightarrow{f \text{ (polynomial time)}} \psi$$

$$\phi \in SAT \iff \psi = f(\psi) \in CNFSAT$$

базис:  $\wedge, \vee, \neg$

строим дерево разбора нашей формулы  $\phi$ :

- если у neg сын neg, то можем удалить
- neg  $\rightarrow$  and/or  $\Rightarrow$  neg  $\leftarrow$  and/or  $\rightarrow$  neg neg

каждому поддереву соответствует преобразованная подформула  $\phi_i(x_{i_1} \dots x_{i_k})$ ,

хотим построить следующее:  $\psi_i(x_{i_1} \dots x_{i_k}, y_1 \dots y_{i_t})$

$$\phi(\bar{X}) = 1 \implies \exists \bar{y} \psi(\bar{x}, \bar{y}) = 1$$

$$\phi(\bar{X}) = 0 \implies \forall \bar{y} \psi(\bar{x}, \bar{y}) = 0$$

вершина	brand new $\psi$
X	$\phi = X, \psi = X$
neg X	$\phi = \neg X, \psi = \neg X$
and	$\phi_1 \wedge \phi_2, \psi_1 \wedge \psi_2$
or	$\psi_1 \vee \psi_2$ не можем написать, потому что это не будет в КНФ новая переменная z: $(\psi_1 \vee z) \wedge (\psi_2 \vee \neg z)$

получается, что число клозов равно числу листьев

внутри каждого клоза число вхождений равно числу переменных + или

#clauses = #leaves

#entries = #vars + #or

poly

□ *qed*

### НЗ Th CNFSAT to 3SAT

$3SAT = CNFSAT \wedge 3CNF$

1.  $3SAT \in NP$

2.  $3SAT \in NPH$

$CNFSAT \leq 3SAT$

$\psi$	$X$
$(x \vee y \vee u) \wedge (x \vee y \vee \neg u)$	$x \vee y$
ok	$x \vee y$
вспомогательные переменные k - 3 новые перменные: $(x_1 \vee x_2 \vee t_1) \wedge (\neg t_1 \vee x_3 \vee t_2) \wedge (\neg t_2 \vee x_2 \vee t_3) \wedge \dots \wedge (\neg t_{k-3} \vee x_{k-1} \vee x_k)$	$x_1 \vee x$

□ *qed*

3SAT - superstar

### Н2 Th IND in NPC

дана формула  $\phi$  в 3КНФ, мы хотим вывести граф G и число k, такие что  $\phi$  удовлетворима тогда и только тогда, когда в графе есть независимое множество размера k

$\phi \in 3SAT \iff \langle G, k \rangle \in IND$

в  $\phi$  k clauses, граф построим из k triangles

в вершинах переменные, соответствующие claus'am

соединим переменные с их отрицанием

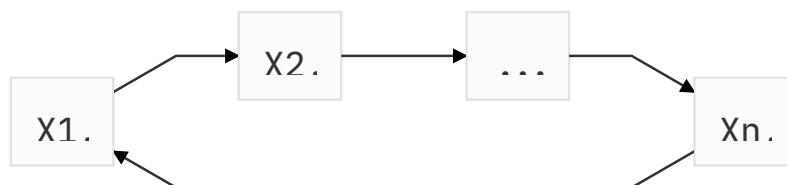
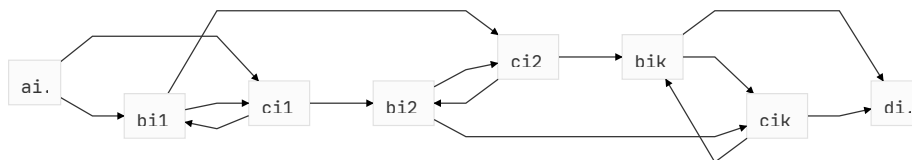
$HAM = \{G \mid G \text{ — ориентированный граф, содержит Гамильтонов цикл}\}$

$HAM \in NP$

$HAM \in NPH$

$\phi(x_1 x_2 \dots x_n)$  k clauses

$x_i \rightarrow 2k + 2$  вершины



где X - это компонента предыдущего вида

## Н2 диагональный метод

## Н3 теоремы об иерархии

$$DSPACE(f) = \{L \mid \exists \text{ программа } p: x \in L \implies p(x) = 1 \text{ } S(p, x) = O(f(n))\}$$

$$x \notin L \implies p(x) = 0$$

$$PSACE = \cup_{p-polynom} DSPACE(p)$$

**Th NP subset PS subset EXP**

**thesis** если p запускает q, q использует  $O(f)$  памяти, то p может тоже для этого использовать  $O(f)$  памяти

## Н4 Th о ёмкости иерархии

$$\frac{f}{g} \rightarrow 0 \text{ тогда } \exists L : L \in DSPACE(g) \setminus DSPACE(f)$$

$$h = \sqrt{fg}, \quad \frac{h}{g} \rightarrow 0, \quad \frac{f}{h} \rightarrow 0$$

$$n = |\langle p, x \rangle|$$

$$L = \{\langle p, x \rangle \mid \text{неверно, что } (p(\langle p, x \rangle) = 1, \text{ использовав } h(n) \text{ памяти})\}$$

$$L \in DSPACE(g)$$

Пусть  $L \notin DSPACE(f)$ , q - разрешает L, используя  $\leq cf(n)$ , рассмотрим

$$n_0 : h(n_0) > cf(n_0), n_0 > |q|$$

рассмотрим  $x : |\langle q, x \rangle| = n_0$

$$q(\langle q, x \rangle) = ?$$

$$q(\langle q, x \rangle) = q \implies \langle q, x \rangle \in L \implies !(q(\langle q, x \rangle) = 1 \text{ and } S(q, \langle q, x \rangle) \leq cf(n) \langle h(n_0) \rangle) \implies q(\langle q, x \rangle) = 0$$

$$q(\langle q, x \rangle) = 0 \implies \langle q, x \rangle \notin L \implies q(\langle q, x \rangle) = 1$$

## Н4 Th о временной иерархии

DSPACE -> DTIME, память -> время

ломается немного первая часть, так что новое условие:

$\frac{f}{g} \rightarrow 0, \exists h : \frac{f}{h} \rightarrow 0, \frac{sim(h)}{g} \rightarrow 0. (sim(h) = O(g))$  (где  $sim(f)$  - за сколько можно просимулировать программу, работающую за  $f$ ) тогда

$\exists L : L \in DTIME(g) \setminus DTIME(f)$

$h = \sqrt{fg}, \frac{h}{g} \rightarrow 0, \frac{f}{h} \rightarrow 0$

$n = |\langle p, x \rangle|$

$L = \{ \langle p, x \rangle \mid \text{неверно, что } (p(\langle p, x \rangle) = 1, \text{ используя } h(n) \text{ времени}) \}$

$L \in DTIME(g)$

Пусть  $L \notin DTIME(f)$ ,  $q$  - разрешает  $L$ , используя  $\leq cf(n)$ , рассмотрим

$n_0 : h(n_0) > cf(n_0), n_0 > |q|$

рассмотрим  $x : |\langle q, x \rangle| = n_0$

**Implies  $P \neq EXP$**

$f = n^{\log_2 n} = 2^{(\log_2 n)^2}$

$g = 2^n$

$\frac{f}{g} \rightarrow 0 \implies \exists L \in DTIME(g) \setminus DTIME(f)$  (первая часть  $\implies L \in EXP$ , вторая –

$\implies L \notin P$ )

### ИЗ Th (Бейкер, Гилл, Соловэй) BGS

$u = \{ \langle p, x \rangle \mid p(x) = 1 \}$

$uni(p, x) \rightarrow$  останавливается ли  $p$  на  $x$

Вычисления с оракулом  $p^A$  –  $p$  с оракулом  $A$

$\exists$  оракул  $A : p^A = NP^A$

$\exists$  оракул  $B : p^B \neq NP^B$

// **релятивизируется**, если доказательство остаётся верным, если всему фиксированному в программе добавить оракул

рассмотрим  $A \in PSC$

$p^A \stackrel{1}{\subset} NP^A \stackrel{2}{\subset} PS^A \stackrel{3}{\subset} PS \stackrel{4}{\subset} P^A$ :

1. любая недетерминированная программа частный случай детерминированной
2. релятивизируется
3. можем заменить вызов оракула на процедуру проверки
4. потому что взяли  $PSpace$  полный, любой сводится за полином и спросим у оракула

$\forall U_B = \{ x \mid \exists y \in B \mid |x| = |y| \}$

$\forall B \ U_B \in NP^B$

Придумаем  $B : U_B \notin P^B$

Теперь рассмотрим часть  $\exists$  оракул  $B : p^B \neq NP^B$ :

Построим последовательность программ  $q_1, q_2, q_3, \dots$

$T(q_i)$  - полином

$\forall L \in P : \exists i : q_i$  разрешает  $L$

Рассмотрим все коды исходных программ, упорядочим их лексикографически и запустим



// n – это длина входа

	$n$	$2n^2$	$3n^3$	...	$kn^k$	...
$p_1$						
$p_2$						
...						
$p_m$					$p_m \mid TL = kn^k$	
...						

каждая из этих программ работает за полином

нумеруем эту табличку по диагонали

получим счётное множество пронумерованных программ

если программа не успела завершиться за  $TL$ , то говорим, что  $q_i$  возвращает 0

так же можем занумеровать все программы с оракулами:  $q_1^*, q_2^*, \dots, q_n^*, \dots$

должны сделать  $B : p^B \neq NP^B$

рассмотрим  $B : U_B = \{x \mid \exists y : |x| = |y|, y \in B\}$

$\mathbf{L} \forall B : U_B \in NP^B$

```

ub(x)
  y ← недетерминированно Sigma^{|x|}
  return check(y)

```

Построить  $B : U_B \notin p^B$  (если построим такое  $B$ , то теорема БГС доказана)

$B_1 : q_1^{B_1}$  не распознавала  $U_{B_1}$

запустим  $q_1$  с оракулом и будем выступать в роли оракула

$q_1^*(x_1)$  : спрашивает оракула  $?y_1 \rightarrow NO$  (пишем в тар наши ответы)

$?y_2 \rightarrow NO \dots ?y_k \rightarrow NO$

// выберем  $x_1 : T(q_1, x_1) < 2^{|x_1|}$

если результат программы  $YES : \forall z \mid |z| = |x_1| : z \notin B_1$

$NO : \exists z_1 : q_1^*(x_1)$  не задала вопрос про  $z_1, |z_1| = |x_1|; z_1 \in B_1$

$B_1 \rightarrow B_2$   $q_1^{B_2}$  не распознаёт  $U_{B_2}$ ,  $q_2^{B_2}$  не распознаёт  $U_{B_2}$

$T(q_2^*, x_2) < 2^{|x_2|}, |x_2| >$  максимальной длины, для которого известно принадлежность  $B_1$

теперь запускаем  $q_2(x_2)$ : спрашивает у нас: если спрашивали уже про это слово, то я то же самое и отвечаю, если нет, отвечаю  $NO$  и записываю

$B_k \forall i \leq k : q_i^{B_k}$  не распознаёт  $U_{B_k}$

опять находим  $x_k$  и запускаем

тот же самый подход, что и выше, при запуске

этот процесс продолжается до бесконечности

для ответа БГС возьмём  $B = \bigcup_{k=1}^{\infty} B_k$

// релятивизация – это барьер доказательства  $P \neq NP$

$$P \neq NP \implies \exists L : L \notin P, L \notin NPC, L \in NP$$

иллюстрация, **не** доказательство

*Blowing Holes in SAT*

координатная ось с итерированным логарифмом

$$1 \rightarrow 10 \rightarrow 10^{10} \rightarrow 10^{10^{10}}$$

выбираем нечётные промежутки

$$SAT_0 = SAT \cap EVEN$$

$$EVEN = \{x \mid \log_{10}^* |x| \text{ чётен}\}$$

к нему сводится  $SAT$ :

$$\exists f : x \in SAT \iff f(x) \in SAT_0$$

так же, как в теореме БГС, у нас есть последовательность  $q_1, q_2, \dots, q_n, \dots$ , так же запускаем программу  $p_i$  с таймером  $jn^j$  и так же занумеровали программу по диагонали:  $f_1 \dots f_i \dots$

все  $f_i$  работают за полином

$$L = SAT \cap EVEN (SAT \cap \{\phi \mid |\phi| \text{ в "чёрном" куске}\})$$

рассмотрели первый чёрный кусок, префикса которого достаточно, чтобы программа  $q_1$  не разрешала  $L$  за полином

теперь рассмотрим некий белый кусок: добьёмся того, чтобы сведение  $f_1$  неправильно сводило  $SAT$  к нашему языку

занумеруем формулы по возрастанию длины и дальше лексикографически:

$$\phi_1, \phi_2, \dots$$

$$\phi_1 \xrightarrow{f_1} z_1$$

$$\phi_2 \rightarrow z_2$$

...

$$\text{найдётся формула } \phi_x \xrightarrow{f_1} z_x : \phi_x \in SAT \neq z_x = f_1(\phi_x) \in L$$

найдётся такая  $\phi_x$  потому, что если бы не нашлось, то получили бы противоречие в том, что  $SAT$  сводится за полиномиальное время под действием  $f_1$  к конечному языку

$z_x$  лежит либо в первом чёрном отрезке, либо во втором белом

$$n_2 = \max(n_1 + 1, |z_x|)$$

**Lemma**  $L \in NPC, F$  – конечный,  $L \setminus F \in NPC$

$$L \leq L \setminus F$$

```
f(x):
  if x in F
    if x in L return YesWord
    else return NoWord
  else return x
```

построим  $BLACK$ :

1.  $x \in BLACK$  – зависит только от  $|X|$
2.  $BLACK \in P$

3.  $L \notin NPC, L \notin P$

разрешитель *BLACK*: (верно ли, что слова длины  $n$  принадлежат нашему языку, пусть работает за  $n$ )

```
black(x: String)
  a = black(|x|)
  return x in BLACK // основываясь на данных из массива a

black(n): List<Int>
// [n1, n2, ..., nk] - список всех границ, которые не превышают n
// ограничение по времени  $n^{(большое\ число, \text{ пусть } 100)}$ 
  if n = 0 return []
  a = black(n - 1)
  // black(n - 1) отработала за  $T \leq (n - 1)^{100}$ ,  $T_{left} \geq n^{99}$ 
  set Timer on  $n^{99}$ , if triggered return a
  if len(a) чётна:
    i = len(a) / 2 + 1
    for (phi - формула, |phi| ≤ n):
      if (phi in SAT intersect BLACK ≠ q_i(phi))
        return a ++ [n]
  else // len(a) нечётна
    i = (len(a) - 1) / 2 + 1
    for (phi - формула, |f_i(phi)| ≤ n):
      if (phi in SAT ≠ f_i(phi) in SAT intersect BLACK):
        return a ++ [n]
  return a
```

## Н2 coNP

def  $coNP = L \mid \bar{L} \in NP$

ex  $SAT \in NP$ ,  
 $\overline{SAT} \in coNP$

есть все слова  $\Sigma^*$ , среди них есть булевы формулы и давайте рассматривать только булевы формулы, они делятся на  $SAT$  и на  $\overline{SAT}$ , а на небулевы формулы забудем

$\overline{SAT} = \{\phi \mid \forall \vec{x}: \phi(\vec{x}) = 0\}$

ex  $FACTORIZATION = \{\langle n, x \rangle \mid \exists \text{ простой делитель } \leq x\} \in NP \cap coNP$   
(P candidate)

## Н2 PSpace и PSpace полнота

def  $PS = \cup_{p-polynom} DSPACE(p)$

$P \subset NP \subset PS \subset EXP$

def  $L \in PSH: \forall A \in PS: A \leq L$  ( $f$  — за полином  $x \in A \iff f(x) \in L$ )

def  $L \in PSC: 1) L \in PSH$   
2)  $L \in PS$

**ex** булевы формулы с квантора (матлог референс)

$TQBF$  (True Quantified Boolean Formula) =  $\{\phi \mid \phi - \text{булева формула с кванторами,}$

$Free(\phi) = \emptyset \quad val(\phi) = 1\}$

### НЗ **TQBF in PSC**

1.  $TQBF \in PS$

построим дерево разбора и храним множество значений текущих свободных переменных

2.  $TQBF \in PSH$

рассмотрим  $L \in PS$ ,  $L \leq TQBF$

$m$  - машина Тьюринга, разрешающая  $L$ , детерминированная,  $S(m, x) \leq p(n) //$

$n = |x|$

$m(x) \quad q_0 \vdash q_1 \vdash q_2 \vdash \dots \vdash q_t$

$f : x \rightarrow \phi$

$\phi - \text{истина} \iff m(x) = 1$

$X_{ijc}$  - ячейка  $(i, j)$  содержит символ  $c$

$Q_i = [X_{i0c_1}, X_{i1c_1}, \dots, X_{ip(n)c_1}, X_{i0c_2}, \dots, X_{ip(n)c_2}]$

$S(Q_0) \cap T(Q_t) \cap C \cap N$

введём синтаксический сахар:  $\exists(\forall)Q_i := \exists(\forall)X_{i0c_1}, \exists(\forall)\dots$

$Q_i \vdash Q_{i+1}$

$\exists Q_0 \exists Q_1 \dots \exists Q_t \quad S(Q_0) \wedge T(Q_t) \wedge C \wedge Q_0 \vdash Q_1 \wedge Q_1 \vdash Q_2 \wedge \dots \wedge Q_{t-1} \vdash Q_t$

выведенная формула плоха её длиной:  $Q(Q_0)$ ,  $T(Q_t)$ ,  $Q_0 \vdash Q_1$  имеют длину  $p(n)$ , но последних кусков  $t$ , таким образом вся формула имеет длину  $p(n)2^{q(n)}$ , а это не полиномиальное сведение

$Q \vdash R$

$\vdash$  - булева формула от  $2(p(n) + 1)z$  аргументов

$Q \vdash R := Q \vdash \underbrace{U_1 \vdash U_2 \dots \vdash U_{2^m-1}}_{2^m} \vdash R$

$\vdash_m = \vdash^{2^m}$

$Q \vdash_m R = \exists T (Q \vdash_{m-1} T \wedge T \vdash_{m-1} R)$

$Q \vdash_m R = \exists T \forall A \forall B (\neg(A \vdash_{m-1} B) \rightarrow (Q \neq A \vee B \neq T) \wedge (T \neq A \vee B \neq R))$

$len(m) = O(p(n)) + len(m-1) \implies len(m) = O(p(n) m)$

□

---

// PS proof template:  $PS \rightarrow TQBF \rightarrow L$

### НЗ **Th $NSPACE(f(n))$ subset $DSPACE(f(n)^2)$**

$f(n) \geq \log(n)$

$NSPACE(f(n)) \subset DSPACE(f(n)^2)$

Доказательство:

Пусть  $L \in NSPACE(f(n)) \exists$  недетерминированная машина Тьюринга  $x \in L \iff \exists$  последовательность недетерминированных выборов,  $m(x) = 1$

$S(m, x) \leq f(n)$ ,  $n = len(x)$

вход — лента машины Тьюринга со словом  $x$

рабочая — лента машины Тьюринга с  $f(n)$

конфигурация машины Тьюринга кодируется:  $(pos, work)$ , где  $work = \alpha \#_p \beta$ , длина  $pos = \log(n)$ , а длина  $work = f(n) + 1$ , и тогда вся длина пары —  $O(f(n))$

Существует ли последовательность переходов длиной  $2^{c f(n)}$ , которая  $q_0$  переводит в допускающую конфигурацию  $q_t$

заведём функцию (можно ли достичь):  $Reach(q_s, q_t, k)$  (можно ли из  $q_s$  перейти за  $2^k$  шагов до  $q_t$  ( $q_s \vdash^{2^k} q_t$ ))

```
Reach(qs, qt, k):
  if (k = 0):
    return qs  $\vdash$  qt
  for (qm - конфигурация машины Тьюринга m):
    if Reach(qs, qm, k - 1) and Reach(qm, qt, k - 1):
      return True
  return False
```

локальные переменные функции `Reach` занимают  $f(n)$ , суммарно памяти нам понадобится  $O(k f(n))$

```
inL(x):
  qs - стартовая конфигурация m
  for (qt - допускающая конфигурация m):
    if Reach(qs, qt, c * f(|x|)):
      return 1
  return 0
```

$q_s$  требует  $f(n)$  памяти

вызов `Reach` требует  $f(n)^2$  памяти

локальная переменная  $q_t$  требует  $f(n)$  памяти

#### Н4 Следствие Th (Сэвитча)

$$PS = NPS$$