1 Билет 1

Утверждение 1.1. *Пусть M – одноленточная МТ, которая распознает язык бинарных палиндромов. Тогда существует константа $C : \exists n_o : \forall n > n_0$ существует вход длины n, на котором $M(x)$ делает $\geq Cn^2$ шагов.*

Доказательство. В начале очевиден принцип несжимаемости, нельзя инъективно перевести строки из $\{0,1\}^n$ в $\{0,1\}^*$ так, чтобы все образы по длине были меньше чем n.
Будем доказывать для входов, длина которых кратна 3. По $x \in \{0,1\}^n$ строим вход $x0^n x^{rev}$ и скармливаем МТ все такие входы. Возьмем все перегородки после нулей, их всего n, существует перегородку, через которую МТ прошла $\leq \dfrac{T(x)}{n}$ раз.
Теперь строим отображение $f : \{0,1\}^n \to \{0,1\}^*$, переводим строку x в протокол работы МТ на строке $x0^n rev(x)$. Для этого выпишем набор состояний, в которые переодила МТ, переходя через "хорошую"перегородку и номер этой перегородки.
Утверждается, что такая f – инъекция, чтобы доказать, предположите обратное и рассмотрите работу на строке $x0^n y^{rev}$.
Пусть $|x| = n$, тогда $f(x) \leq \log n + \dfrac{T(x)}{n} C$, но при этом существует $|y| = n$, такой что $f(y) \geq n$. Получаем, что

$$n \leq logn + \frac{T(x)}{n} C$$

$T(x) = \omega(n^2)$ на таких входах.
Для некратных 3 входов делаем также, но по-середине пишем вместо n нулей, на один ноль больше или меньше – это не влияет на оценки. ∎

2 Билет 2

Определение 2.1. *k – ленточная машина Тьюринга. (Добавляется куча лент и функция перехода теперь действует по всем лентам).*

Утверждение 2.1. *Для любой k – ленточной МТ, которая на входе x работает время $T(x)$, существует 1 ленточная МТ, которая работает $O(T(x)^2)$.*

Доказательство. Будем хранить в одном символе МТ символы всех лент (а также спец символы, помеченные головкой). На каждом шаге будем идти вправо и делать все изменения, которые нужны на лентах. ∎

Определение 2.2. *Универсальная МТ – эмулирует МТ по описанию.*

Утверждение 2.2. *Для любой k-ленточной МТ существует универсальная k-ленточная МТ с линейным замедлением.*

Доказательство. Понятно как получить квадратичное замедление, нужно положить описание в начало, например, первой ленты. Далее постоянно возвращаться, чтобы узнать, какой шаг сделать. Если же хотим линейного – давайте возить описание с собой, это будет давать $O(1)$ действий из-за его константного размера, при этом эмуляция будет работать за линейное время. ∎

Утверждение 2.3. *k ленточную МТ можно эмулировать на 2-ленточной с логарифмическим замедлением.*

Доказательство. TODO ∎

3 Билет 3

Основная модель вычислений – многоленточная МТ.

Определение 3.1. *$f : \mathbb{N} \to R_+$, тогда $L \in DTime[f(n)]$, если существует многоленточная МТ, такая что*

1. *$\forall x \in L \Rightarrow M(x) = 1$.*
2. *$\forall x \notin L \Rightarrow M(x) = 0$.*
3. *$\forall x$ МТ работает $O(f(|x|)$ шагов.*

Определение 3.2. *$P = \cup_{i>0} DTime[n^i]$.*

Определение 3.3. *Про семейство схем, распознающих язык.*

Определение 3.4. *$L \in Size[f(n)]$, если есть последовательность схем, распознающих L и для достаточно больших n выполнено $|C_n| \leq f(n)$.*

Определение 3.5. *$P/Poly = \cup_{i>0} Size[n^i]$.*

Пример 3.1. *Неразрешимый язык может лежат в $P/Poly$. Например $1^H = \{1^n | n \in H\}$, для некоторого языка тоже является разрешимым и лежит в $P/Poly$, так как на каждую длину мы можем предоставить схему.*

Утверждение 3.1. *Существует такой алгоритм A, который получает на вход T, n, m и*

1. *A работает $poly(n + T + |m|)$ шагов.*
2. *Если МТ m на всех входах из $\{0,1\}^*$ выдает ответ за $\leq T$ шагов, то алгоритм A выдает схему C, которая имеет n входов и 1 выход и распознает на входах длины n также как m.*

Доказательство. Будем возвращать схему размера $T \times T \cdot O(1)$.
На уровне i будет T ячеек, в каждой из которых будет вычисляться некоторая информация: сивмол, написанный в этой ячейке, есть ли тут головка в момент i, а также, если есть головка, то состояние, в которой МТ сейчас находится. Понятно, что для пересчета этих параметров нужно обратиться к нескольким соседним ячейкам предыдущей строки. Для того, чтобы узнать ответ, посмотрим, принималось ли где-нибудь состояние q_{yes}.

■

Утверждение 3.2. *$P \subseteq P/Poly$.*

Замечание 3.1. *Таким образом хотели доказывать, что $P \neq NP$, взять, к примеру, SAT и показать, что он не лежит в $P/Poly$, однако доказывать нижние оценки на схемы пока что не научились.*

4 Билет 4

Определение 4.1. *$L \subseteq \Sigma^*$, система доказательств для языка L – это такой алгоритм Π, который обладает следующими свойствами:*

1. *(Полнота) $\forall x \in L \Rightarrow \exists w : \Pi(x, w) = 1$*
2. *(Корректность) $\forall x \notin L \Rightarrow \forall w \Pi(x, w) \neq 1$ (Ну или равно 0).*

3. Π *всегда останавливается*

Определение 4.2. *Система доказательств* Π *называется эффективной, если* $\Pi(x, w)$ *работает за* $\leq poly(|x| + |w|)$ *шагов.*

Замечание 4.1. *Системы доказательств существуют для перечислимых языков, как мы знаем из предыдущий главы курса. Также можно доказать, что для всех перечислимых языков существует и эффективная система доказательств* ($TCS12$)*, искуственно увеличивая подсказку.*

Определение 4.3. *класс* NP *состоит языков, для которых существует эффективная система доказательств* Π*, а также полином* q*, такой что* $\forall x \in L \exists w, |w| \leq q(|x|), \Pi(x, w) = 1$*, то есть, существует полиномиальная подсказка.*

Пример 4.1. *Примеры языков из* NP*.*

1. SAT *– множество выполнимых пропозициональных формул.*
   $SAT \in NP$*, но не выяснено,* $UNSAT \notin / \in NP$*.*

2. $Hampath$ *– язык графов, в которых есть гамильтонов путь, также лежит в* NP*.*

3. $CLIQUE$ *– язык пар (граф, число) такой, что в графе есть клина на числе вершин. Лежит в* NP*.*

4. $Composite$ *– язык составных натуральных чисел, лежит в* NP*, а также, известно, что* $Primes \in NP$ *(*$TCS11$*) и, более того, человечество умеет показывать* $Primes \in P$*.*

Определение 4.4. *Недетерминированные МТ. Вместо одной функции перехода теперь две и машина сама выбирает, в какую идти. (:)).*
Говорят, что недетерминированная M *принимает слово, если существует последовательность корректных переходов, при которых она придет в состояние* q_{yes} *на входе этом слове.*
Время работы НМ – максимум по всем возможным применениям функции перехода.

Определение 4.5. $NTime[f(n)]$ *– множество языков, которые принимаются многоленточными НМТ за* $O(f(n))$ *шагов, где* n *– длина входа.*

Определение 4.6 (Второе определение NP)**.** $NP = \cup_{c>0} NTime[n^c]$.

Определение 4.7 (МТ с подсказкой)**.** *. К обычной МТ добавляется лента подсказки, на которую записывается некоторая строка, к которой может обращаться МТ во время работы. Машина принимает слово, если существует подсказка, для которой она придет в состояние* q_{yes}*. Время работы такой машины – максимум по всем подсказкам.*

Теорема 4.1. *Следующие условия эквивалентны:*

1. $L \in NP$ *(на языке систем доказательств)*

2. L *распознается машиной Тьюринга с подсказкой за полиномиальное время.*

3. $L \in \cup_{c>0} Ntime[n^c]$*.*

Доказательство. (1) $\Rightarrow$ (2): построим МТ с подсказкой. Пусть наша МТ при обращениях к подсказке будет теперь обращаться на ленту с подсказкой вместо ленты входа. Тогда понятно, что подсказки аналогичны друг другу.
(2) $\Rightarrow$ (3) : пусть МТ порождает подсказку, а далее действует детерминированно. Подсказка более чем полиномилаьного размера не нужна, так как наш алгоритм не успеет ее обработать из-за своего времени работы.
(3) $\Rightarrow$ (1) : подсказка – какую функцию перехода выбирать на каждом шагу. ■

5 Билет 5

Определение 5.1. *Язык A сводится по Карпу к языку B, если существует полиномиально вычислимая $f \colon \forall x, x \in A \iff f(x) \in B$. Обозначается $A \leq_p B$.*

Утверждение 5.1. *Свойства сведения*

1. $A \leq_p B$, $B \in P \Rightarrow A \in P$.
2. $A \leq_p B, B \leq_p C \Rightarrow A \leq_p C$.

Определение 5.2. *Язык A называется NP-трудным, если $\forall L \in NP, L \leq_p A$. Язык A NP-полный, если $A \in NP$ и A – NP-трудный.*

Определение 5.3. $BH = \{(M, x, 1^t) | \ \exists y, M(x,y)$ *выдает 1 за* $\leq t$ *шагов* $\}$.

Теорема 5.1. *BH – NP-полный.*

Доказательство. Проверим, что $BH \in NP$. Подсказка как раз и будет этот y. Запускаем на t шагов и проверяем, приняло или нет. Если тройка лежит в языке, то по определению найдется такая подсказка, что выдаст *yes*. Иначе – нет.
Возьмем $L \in NP$, хотим проверить $L \leq_p BH$. У L есть нмт эффективная система доказательств П, работающая за $q(|x| + |y|)$, при этом также для лежащих в языке слов есть маленькая подсказка, длины $\leq p(x)$. Давайте сделаем следующее отображение $f(x) = (M, x, 1^{q(|x|+p(|x|))})$. Очевидно, что это корректное сведение.

∎

6 Билет 6

7 Билет 7

8 Билет 8

Теорема 8.1 (Ладнер)**.** *Если $P \neq NP$, то существует $L \in NP$, такой, что, $L \notin P$ и L – не полный в NP.*

Доказательство. Интуиция следующая: хотим немного ослабить язык SAT. Давайте рассматривать

$$SAT_H = \{\varphi 0 1^{n^{H(n)}} | \varphi \in SAT, |\varphi| = n\}$$

Поймем, как определить H. Возьмем нумерацию всех машин Тьюринга $M_1, M_2, M_3, \ldots$.
$H(n) = i$, если i – такое минимальное число, что $i < \lfloor \log \log n \rfloor$, что M_i решает SAT_H на всех входах $|x| \leq \log n$ за время $i|x|^i$, или, если такого числа нет, то $\lfloor \log \log n \rfloor$.
Заметим, что $H(n)$ определяется через SAT_H и наоборот.
Но заметим, что чтобы определить $H(n)$ нам не потребуется значений $H(x)$ при $x > \log n$, так как $H(n)$ нужно для дописывания только к строкам длины $n + 1$.

Утверждение 8.1. *$H(n)$ вычисляется по значениям $H(1), \ldots, H(n-1)$ за poly(n) действий.*

Будем вычислять по определению. Переберем

1. $i \leq \log \log n$
2. $x, |x| \leq \log n$, это $2^{\log n} = O(n)$ действий

3. запустим машину i на $(\log\log i)(\log i)^{\log\log i} =$ действий.

4. Сравним результат с реальным: мы можем понять, лежит или нет, так как знаем предыдущие значения H, SAT будем решать перебором за $O(n)$.

Таким образом вычислим $H(n)$ за $O(n^3)$ по предыдущим значениям. Тогда мы сможем вычислить $H(n)$ за $O(n^4)$, вычисляя по очереди все значения.

Утверждение 8.2. *$H(n)$ не убывает.*

Действительно, если $H(n) = i$, то такое i подошло и для всех предыдущих, либо i – верхняя граница для $H(n)$, но тогда оно точно больше предыдущих.

Утверждение 8.3. *$SAT_H \in P \Longleftrightarrow H(n) \le C$.*

$\Rightarrow$: есть МТ, решающая SAT_H за $p(|x|)$. Но эта МТ встречается в нумерации бесконечное число раз. Она там также встречается как M_i при $i|x|^i > p(|x|)$. Но тогда $H(n) \le i$, так как такая машина нам все решит и мы ее запустим на такое число шагов.
$\Leftarrow$: Ограниченная неубывающая функция это такая функция, что с некоторого момента она равна j. Но заметим, что тогда M_j решит нам язык за $j|x|^j$, что есть полином. $\blacksquare$

Утверждение 8.4. *$SAT_H \in NP$*

Действительно, давайте просто давать как подсказку решение для внутренней SAT, при этом определить, верное ли кол-во единиц мы можем, вычислив H.

Утверждение 8.5. *$SAT_H \notin P$.*

Пусть это не так. Тогда $H(n)$ ограничена. Тогда $SAT \le_p SAT_H$ и $P = NP$.

Утверждение 8.6. *SAT не сводится к SAT_H полиномиально.*

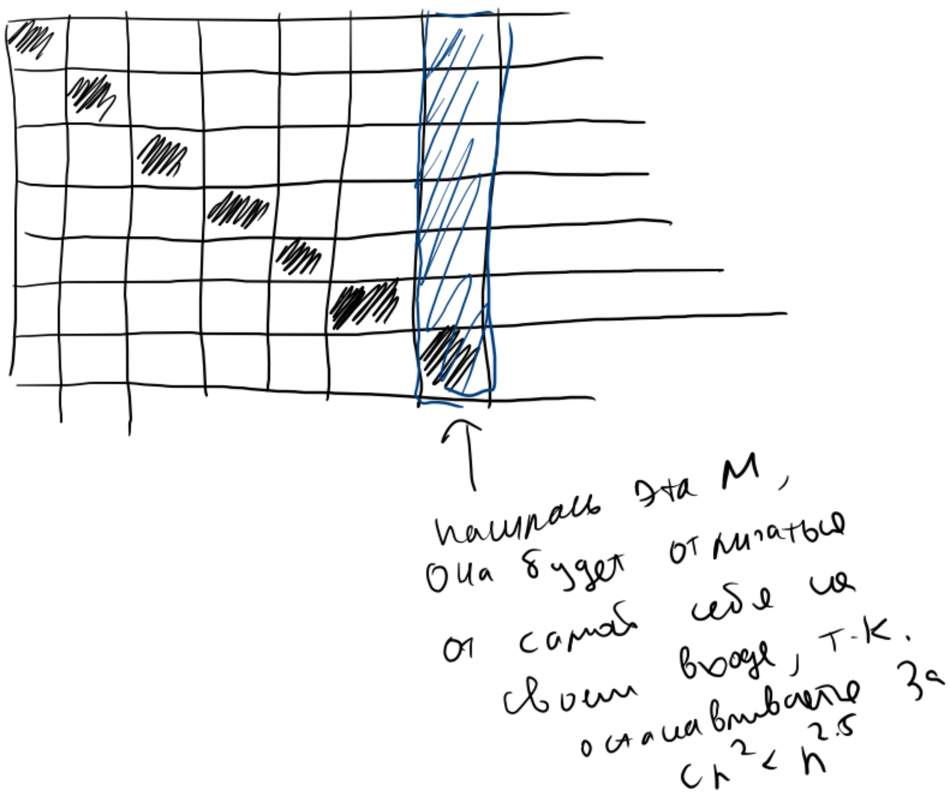
Предположим обратное. Покажем тогда, что мы сможем решить SAT за полином. Пусть сведение работает за n^c. Тогда величина формулы, которую выдаст сведение $\le n^c$, где n – длины формулы, поступившей нам на вход.
$H(n)$ не ограничена, возьмем n_0 такое, что $H(n) > 3c$ при $n > n_0$. Тогда пусть сведение выдало битовую строку, длина которой $m > n_0$ (иначе сделаем полный перебор, который займет $O(1)$) и у которой правильное число единиц на конце (иначе мы это легко проверим за полином).
Тогда $m \ge |\varphi| + 1 + |\varphi|^{3c}$, отсюда $|\varphi| \le m^{1/3c}$, при этом $m \le n^c$. Получается, $|\varphi| \le n^{1/3}$. При больших n такое значение хотя бы в 2 раза меньше n. Значит, мы свелись к формуле меньшего размера в 2 раза. Таким образом мы сделаем не более log полиномиальных сведений и выдадим верный ответ. Отсюда $P = NP$ и противоречие.

9 Билет 9

Пример 9.1. *$DTime[n^2] \subsetneq DTime[n^3]$.*

Доказательство. Давайте диагонализировать.

$L(n) = 1 - \langle n \rangle(n)$, если $\langle n \rangle(n)$ ост. за $n^{2.5}$ шагов



Рассмотрим язык $L = \{M | M$ отвергает M за не более $|M|^{2.5}$ $\}$. Пусть он решается квадратичной M за Cn^2 шагов. Тогда давайте найдем эквивалентную ей M' такую, что $|M'|^{2.5} > C|M'|^2$. Тогда получится, что M' не равна себе же в строчке, соответствующей M'.
Как показать, что $L \in DTime[n^3]$? Давайте эмулировать МТ, которая работает $O(n^{2.5})$ с логарифмической задержкой.
Тогда получили нужный язык L. ■

Определение 9.1. *$h : \mathbb{N} \to \mathbb{N}$ – конструктивная по времени, если $n \to h(n)$ можно вычислить за $O(h(n))$ шагов на ДМТ.*

Теорема 9.1 (Об иерархии по времени для детерменированных вычислений)**.** *$f, g, h : \mathbb{N} \to \mathbb{N}$, h – конструктивная по времени.*
$f(n) = o(h(n)), h(n) \log h(n) = o(g(n))$, тогда $DTime[f(n)] \subsetneq Dtime[g(n)]$.

Доказательство. Конструкция такая же, как и в предыдущем утверждении. Конструктивная функция нужна для будильника в МТ. log – для эмуляции. ■

Утверждение 9.1. *$P \subsetneq EXP$.*

Доказательство. $P \subseteq DTime[2^n] \subsetneq DTime[2^{n^2}] \subseteq EXP$. ■

Замечание 9.1. *Доказательство не работает для НМТ, потому что мы не можем реверснуть ответ за такое же время.*

Пример 9.2. $NTime[n^2] \subsetneq NTime[n^3]$.

Доказательство. Определим для M_i из перечисления всех НМТ отрезок $[n_i, n_i^*]$ так, что $n_i^* = 2^{n_i^{2.5}}$. $n_{i+1} = n_i^* + 1$.
$[n_1, n_1^*], [n_2, n_2^*], [n_3, n_3^*], \ldots$.
Определим язык L так:

1. $L(n_i^*) = 1 - M_i(0^{n_i})$, если $M_i(0^{n_i})$ завершилось за менее чем $n_i^{2.5}$ шагов и 0 иначе.

2. $L(n) = M_i(0^{n+1})$ для $n_i \le n < n_i^*$

Утверждение 9.2. $L \notin NTime[n^2]$.

Пусть это не так, тогда есть M, решающая L за Cn^2. Возьмем такое ее вхождение, что $\forall n \in [n_i, n_i^*], n^{2.5} > Cn^2$. Тогда:

$$M(0^{n_i}) = L(0^{n_i}) = M(0^{n_i+1}) = L(0^{n_i+1}) = \ldots = L(0^{n_i^*}) \neq M(0^{n_i})$$

так как все машины успеют отработать.

Утверждение 9.3. $L \in NTime[n^3]$. *Разбираем случаи, если надо проэмулировать, то эмулируем, иначе нам нужно реверснуть выход недетерменированной машины. Мы можем сделать это за* $2^{n_i^{2.5}} \cdot n_i^{2.5} < ((n_i)^*)^3$.

■

Теорема 9.2 (Об иерархии по времени для недетерменированных вычислений)**.** . $f, g, h : \mathbb{N} \to \mathbb{N}$, h – *конструктивная по времени.* $f(n) = o(h(n))$, $h(n+1) = o(g(n))$, *тогда* $NTime[f(n)] \subsetneq NTime[g(n)]$.

Утверждение 9.4. $NP \notin NEXP$.

Доказательство. аналогично детерменированному случаю. ■

10 Билет 10

Определение 10.1 (Модель вычислений с ограничением по памяти.)**.** *МТ с дополнительной лентой входа, она read-only, также по ней нельзя уйти правее первого пробела. Также есть лента выхода, по которой нельзя двигаться влево, а только выдавать очередной символ ответа. Затраченная память – максимальный уход вправо на какой-то из рабочих лент.*

Определение 10.2. $DSpace[f(n)]$ – *класс языков, которые принимают ДМТ, использующие* $O(f(n))$ *памяти.* $NSpace[f(n)]$ – *то же самое, но для НМТ.*

Определение 10.3. $PSPACE = \cup_{c>0} DTime[n^c]$, $NPSPACE = \cup_{c>0} NTime[n^c]$.

Определение 10.4. $L = LOGSPACE = DSPACE[\log(n)]$, $NL = NSPACE[\log n]$.

Утверждение 10.1. $\forall s(n) : DTime[s(n)] \subseteq DSpace[s(n)] \subseteq NSpace[s(n)]$. *Причем первое включение работает лишь для некоторых моделей вычислений. В частности, для МТ.*

Определение 10.5. $S : \mathbb{N} \to \mathbb{N}$ – *конструктивная по памяти, если* $1^{S(n)}$ *можно вывести за* $O(S(n))$ *памяти.*

Теорема 10.1. $s(n)$ – *конструктивная по памяти и* $s(n) \ge \log n$. *Тогда*

$$NSpace[s(n)] \subseteq DTime[2^{O(s(n))}] = \cup_{c>0} DTime[2^{cs(n)}]$$

.

Доказательство. Для доказательства используется идея с графом конфигураций. Конфигурация МТ это набор параметров:

1. Положение головок на всех лентах
2. Содержание рабочих лент
3. Состояние

Проблема выяснения того, принимает ли МТ вход x может быть рассмотрена как проблема выяснения существования пути в графе конфигураций. Поймем, сколько есть конфигураций:
$n \cdot (Cs(n))^k$ – положение головок, $2^{c's(n)k}$ – содержимое рабочих лент. Если $s(n) \geq \log n$, то это число есть $2^{O(s(n))}$. В таком случае мы можем сгенерировать такой граф и проверять наличие пути полиномиальным алгоритмом. Получится время работы $poly(2^{O(s(n))} = 2^{O(s(n))}$.
Зачем пользовались конструктивностью функции по времени? Для того чтобы сгенерировать конфигурацию нужно отмерить максимальную длину конфигурации и дальше уже перебирать все возможные строчки. Поэтому хочется уметь отмерить за нормальную память.
То есть итоговый алгоиртм такой: по M, x строим граф конфигураций и ищем путь из K_0 в K_{accept}. Можно сделать одно состояние K_{accept}, попросив МТ стирать все рабочие ленты перед тем как завершиться. Так мы унифицируем конечные состояние, но, очевидно, не изменим вычислительную мощь (:)). ■

Утверждение 10.2. .

1. *$NSPACE \subseteq EXP$*
2. *$NP \subseteq EXP$*

Теорема 10.2 (Савич). *$s(n)$ – конструктивная по времени и $s(n) \geq \log n$, тогда $NSpace[s(n)] \subset DSpace[s(n)^2]$.*

Доказательство. Все сводится к тому, чтобы по графу понять, есть ли в нем путь от вершины до другой за память $s(n)^2$, где $s(n)$ – память рассматриваемой НМТ. Воспользуемся предикатом $PATH(u, v, i)$ – есть ли путь от u до v длины не более 2^i и рекурсивным перебором. ■

Утверждение 10.3. $NPSPACE = PSPACE$

$$P \subseteq NP \subseteq NPSPACE = PSPACE \subseteq EXP \subseteq NEXP$$

11 Билет 11

Рассмотрим кванторные пропозициональные формулы: $Q_1 x_1, \ldots, Q_n x_n \varphi(x_1, \ldots, x_n)$. Язык таких истинных формул это $TQBF$. Понятно, что $SAT \leq_p TQBF$ просто дописываниям ко всем переменным квантора существования.

Утверждение 11.1. *$TQBF$ лежит в $PSPACE$*

Действительно, можно просто сделать перебор рекурсивно и проверить выполнимость.

Утверждение 11.2. *$TQBF$ – полный в классе $PSPACE$.*

Доказательство. Берем граф конфигураций. Хотим записать формулу $PATH(K_0, K_{accept}, cp(n))$. (время работы не более $2^{cp(n)}$, так как иначе все зациклится).
Будем записывать при помощи следующего выражения:
$PATH(u, v, i) = \exists z \forall A, B((A = u, B = z) \vee (A = z, B = v)) \to PATH(A, B, i - 1)$, тогда мы сможем записать всё за полиномиальное количество бит. Остаются детали, как записать в виде формулы утверждения вида $PATH(u, v, 0)$. Мы можем с помощью полиномиального алгоритма проверить такой предикат.
Еще нужно подумать о том, как записать равенство конфигураций, это можно сделать просто побитово. ■

Замечание 11.1. *Выяснение вопроса детерменированной победы в конечных играх – задача из $PSPACE$, так как ее можно записать в $TQBF$ в виде $\exists step_{1,1} \forall step_{2,1} \exists step_{1,2} \dots Q(..)$, что означает, что есть ход первого игрока, что при любом ходе второго есть ход первого и тд, что первый игрок выиграл.*