# Алгоритмическая теория игр
# Лекция 4

М.Н. Вялый

Лекции в Computer Science club (Санкт-Петербург), 2019

# Двоичные клеточные автоматы и игры вычитания

# Основная цель

Представить эволюцию двоичного КА $C = (\{0, 1\}, r, \delta)$, начинающуюся в конфигурации $\ldots 11011 \ldots$, с помощью функции оценки позиций в некоторой многомерной игре вычитания.

**Координаты в пространственно-временно́й диаграмме**

Время КА течёт вдоль $(1, 1)$

Ячейки располагаются вдоль $(1, -1)$.

Моменту $t$ отвечают точки на прямой $x_1 + x_2 = 2Nt$.

Номер ячейки равен $(x_1 - x_2)/2$.

$N$ — подходящая константа (заведомо $N \geqslant r$).

# Основная цель

Представить эволюцию двоичного КА $C = (\{0,1\}, r, \delta)$, начинающуюся в конфигурации $\ldots 11011 \ldots$, с помощью функции оценки позиций в некоторой многомерной игре вычитания.

**Координаты в пространственно-временно́й диаграмме**

Время КА течёт вдоль $(1,1)$

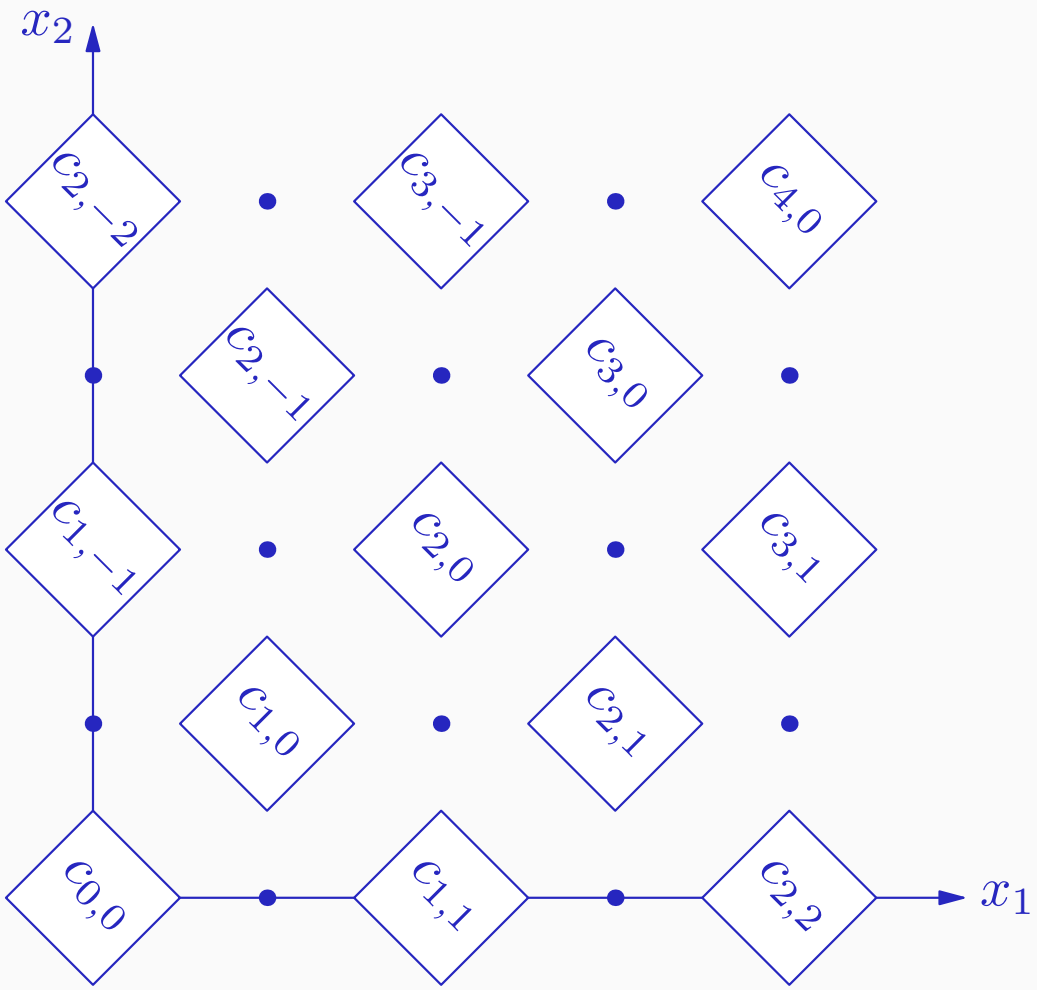
Ячейки располагаются вдоль $(1,-1)$.

Моменту $t$ отвечают точки на прямой $x_1 + x_2 = 2Nt$.

Номер ячейки равен $(x_1 - x_2)/2$.

$N$ — подходящая константа (заведомо $N \geqslant r$).

# Иллюстрация для $N = 1$



# Условие соответствия между КА и игрой

Стремимся к выполнению соотношения

$$c(t,i) = \mathrm{val}(Nt+i, Nt-i) \quad \text{при } |i| \leqslant Nt,$$

при остальных значениях $t$, $i$ полагаем $\mathrm{val}(Nt+i, Nt-i) = 1$.

**Пример**

КА: $r = 1$ и $\delta(x,y,z) = \neg x \vee \neg y \vee \neg z$. Этому КА соответствует игра с $\mathcal{D} = \{(2,0),\ (1,1),\ (0,2)\}$ (при $N = 1$).

Рекуррента для оценки позиции:

$$\mathrm{val}(i,j) = \neg\,\mathrm{val}(i-2,j) \vee \neg\,\mathrm{val}(i-1,j-1) \neg\,\mathrm{val}(i,j-2).$$

Если на прямой $x_1 + x_2 = 2t$ соответствие есть, то оно есть и на прямой $x_1 + x_2 = 2(t+1)$. На прямой $x_1 + x_2 = 0$ соответствие есть. Значит, оно есть всегда.

# Условие соответствия между КА и игрой

Стремимся к выполнению соотношения

$$c(t,i) = \mathrm{val}(Nt+i, Nt-i) \quad \text{при } |i| \leqslant Nt,$$

при остальных значениях $t$, $i$ полагаем $\mathrm{val}(Nt+i, Nt-i) = 1$.

**Пример**

КА: $r = 1$ и $\delta(x,y,z) = \neg x \vee \neg y \vee \neg z$. Этому КА соответствует игра с $\mathcal{D} = \{(2,0),\ (1,1),\ (0,2)\}$ (при $N = 1$).

Рекуррента для оценки позиции:

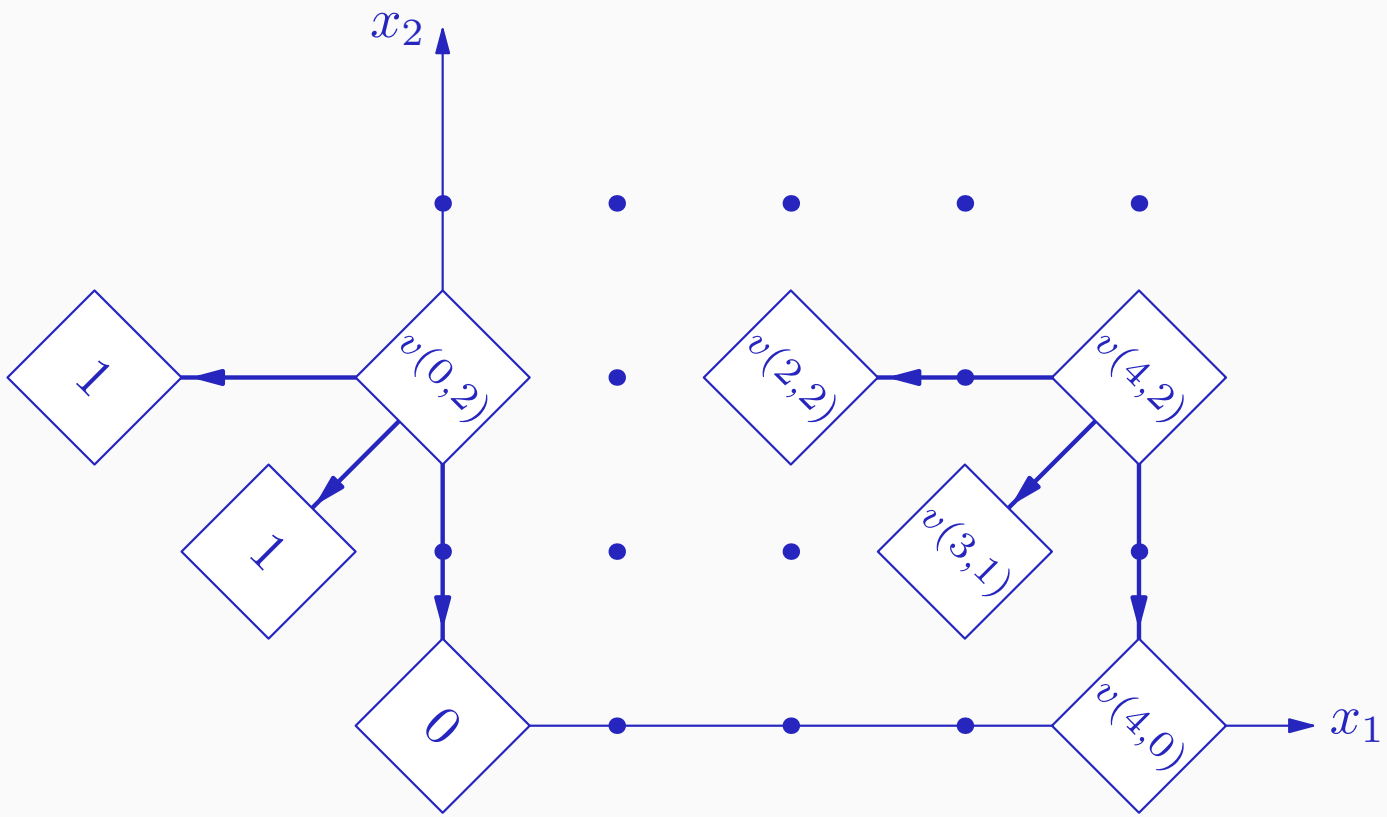
$$\mathrm{val}(i,j) = \neg\,\mathrm{val}(i-2,j) \vee \neg\,\mathrm{val}(i-1,j-1) \neg\,\mathrm{val}(i,j-2).$$

Если на прямой $x_1 + x_2 = 2t$ соответствие есть, то оно есть и на прямой $x_1 + x_2 = 2(t+1)$. На прямой $x_1 + x_2 = 0$ соответствие есть. Значит, оно есть всегда.

# Учёт краевых эффектов

Свойство функции оценки $[a_1, \ldots, a_k] = \bigvee_{i=1}^{k} \neg a_i$:

$$[a_1, \ldots, a_k, 1, \ldots, 1] = [a_1, \ldots, a_k]$$



# Модулярные игры

**Определение**

Задано несколько множеств $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{k-1}$.

Ход в позиции $x \in \mathbb{N}^d$ возможен в позиции $x - a^{(j)}$, где $a^{(j)} \in \mathcal{D}_j$, а $j$ — остаток от деления суммы координат вектора $x$ на $k$.

# Свойство функций оценки позиции

**Утверждение**

Функции $[\ldots]$ образуют полный базис.

**Доказательство**

Выразим функции стандартного полного базиса:

$$\neg x = [x],$$

$$x \vee y = \neg\neg x \vee \neg\neg y = [[x], [y]],$$

$$x \wedge y = \neg(\neg x \vee \neg y) = [[x, y]].$$

# Свойство функций оценки позиции

**Утверждение**

Функции $[\dots]$ образуют полный базис.

**Доказательство**

Выразим функции стандартного полного базиса:

$$\neg x = [x],$$
$$x \vee y = \neg\neg x \vee \neg\neg y = [[x], [y]],$$
$$x \wedge y = \neg(\neg x \vee \neg y) = [[x, y]].$$

# Схема для функции переходов

Схема вычисления функции переходов $\delta(x_{-r}, \ldots, x_{-1}, x_0, x_1, \ldots, x_r)$ двоичного КА:

$$s_1 := [\text{аргументы}_1]$$
$$s_2 := [\text{аргументы}_2]$$
$$\ldots \quad \ldots$$
$$s_N := [\text{аргументы}_N]$$

Без ограничения общности считаем, что $N \geqslant r$ (всегда можно добавить фиктивные присваивания).

Тогда размер этой схемы и есть искомый параметр $N$.

# Схема для функции переходов

Схема вычисления функции переходов $\delta(x_{-r}, \ldots, x_{-1}, x_0, x_1, \ldots, x_r)$ двоичного КА:

$$
\begin{aligned}
s_1 &:= [\text{аргументы}_1] \\
s_2 &:= [\text{аргументы}_2] \\
\ldots &\quad \ldots \\
s_N &:= [\text{аргументы}_N]
\end{aligned}
$$

Без ограничения общности считаем, что $N \geqslant r$ (всегда можно добавить фиктивные присваивания).

Тогда размер этой схемы и есть искомый параметр $N$.

# Построение соответствующей модулярной игры

Модуль равен $2N$.

Условия соответствия

$$v(Nt + i, Nt - i) = c(t, i),$$

$$v(Nt + i + j, Nt - i + j) = s_j, \quad 1 \leqslant j < N,$$

где $s_j$ — значение $j$-го присваивания при вычислении

$$\delta(c(t, i - r), \ldots, c(t, i), \ldots, c(t, i + r))$$

по заданной схеме.

Аргументам $s_j$ соответствуют точки на диаграмме кучек камней с фиксированными сдвигами относительно места $s_j$. Они и дадут множество $\mathcal{D}_{2j}$.

Множества $\mathcal{D}_{2j+1}$ неважны.

# Пример для автомата Паскаля

Функция переходов $\delta(x_{-1}, x_0, x_1) = 1 \oplus x_{-1} \oplus x_0$.

Схема её вычисления размера $N = 5$:

$$
\begin{aligned}
s_1 &:= [x_0] && (s_1 = \neg x_0),\\
s_2 &:= [x_{-1}] && (s_2 = \neg x_{-1}),\\
s_3 &:= [s_1, s_2] && (s_3 = x_{-1} \vee x_0)\\
s_4 &:= [x_{-1}, x_0] && (s_4 = \neg x_{-1} \vee \neg x_0),\\
s_5 &:= [s_3, s_4] && \big(s_5 = (\neg x_{-1} \wedge \neg x_0) \vee (x_{-1} \wedge x_0) = 1 \oplus x_{-1} \oplus x_0\big).
\end{aligned}
$$

# Пример для автомата Паскаля

Функция переходов $\delta(x_{-1}, x_0, x_1) = 1 \oplus x_{-1} \oplus x_0$.

Разностные множества:

$$\mathcal{D}_2 = \{(1,1)\}, \qquad (s_1 := [x_0]),$$

$$\mathcal{D}_4 = \{(3,1)\}, \qquad (s_2 := [x_{-1}]),$$

# Пример для автомата Паскаля

Функция переходов $\delta(x_{-1}, x_0, x_1) = 1 \oplus x_{-1} \oplus x_0$.

Разностные множества:

$$\mathcal{D}_2 = \{(1,1)\}, \qquad (s_1 := [x_0]),$$
$$\mathcal{D}_4 = \{(3,1)\}, \qquad (s_2 := [x_{-1}]),$$
$$\mathcal{D}_6 = \{(1,1),(2,2)\}, \quad (s_3 := [s_1, s_2]),$$

# Пример для автомата Паскаля

Функция переходов $\delta(x_{-1}, x_0, x_1) = 1 \oplus x_{-1} \oplus x_0$.

Разностные множества:

$$\begin{aligned}
\mathcal{D}_2 &= \{(1,1)\}, & (s_1 &:= [x_0]),\\
\mathcal{D}_4 &= \{(3,1)\}, & (s_2 &:= [x_{-1}]),\\
\mathcal{D}_6 &= \{(1,1),(2,2)\}, & (s_3 &:= [s_1, s_2]),\\
\mathcal{D}_8 &= \{(4,4),(5,3)\}, & (s_4 &:= [x_{-1}, x_0]),
\end{aligned}$$

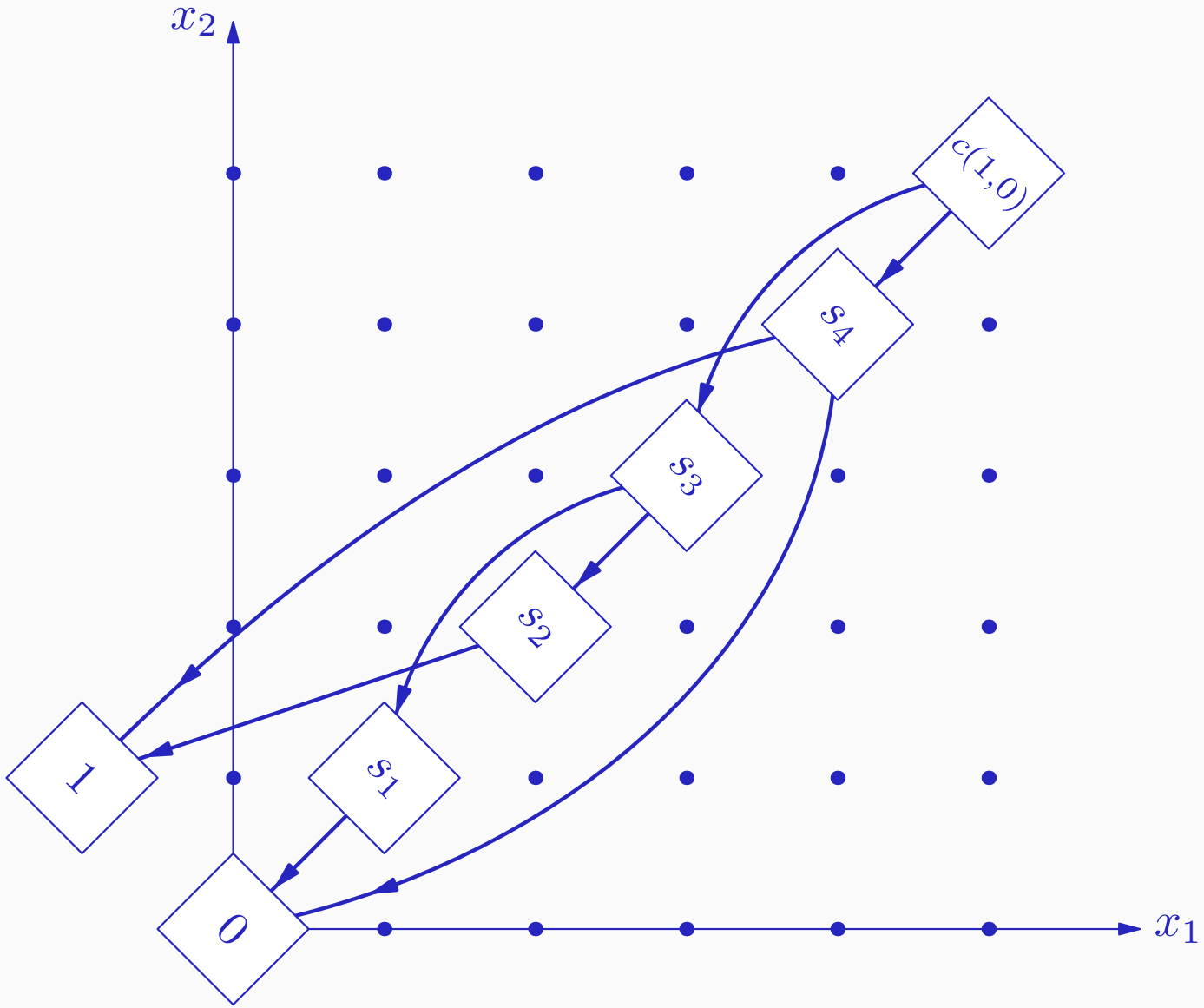
# Пример для автомата Паскаля

Функция переходов $\delta(x_{-1}, x_0, x_1) = 1 \oplus x_{-1} \oplus x_0$.

Разностные множества:

$$\begin{aligned}
\mathcal{D}_2 &= \{(1,1)\}, & (s_1 &:= [x_0]),\\
\mathcal{D}_4 &= \{(3,1)\}, & (s_2 &:= [x_{-1}]),\\
\mathcal{D}_6 &= \{(1,1),(2,2)\}, & (s_3 &:= [s_1, s_2]),\\
\mathcal{D}_8 &= \{(4,4),(5,3)\}, & (s_4 &:= [x_{-1}, x_0]),\\
\mathcal{D}_0 &= \{(1,1),(2,2)\}, & (s_5 &:= [s_3, s_4]).
\end{aligned}$$

# Пример для автомата Паскаля (иллюстрация)



# Идея избавления от остатков

Заводим $2N$ дополнительных кучек камней.

**Условие контроля**

Ограничимся только такими позициями, в которых ровно в одной из дополнительных кучек камней лежит ровно один камень.

Чтобы контролировать остатки, расширим векторы из разностных множеств координатами

$$0, \ldots, 0, 1, 0, \ldots, 0, -1, 0, \ldots, 0.$$

Если условие контроля выполнено, то возможны лишь те ходы, которые вынимают камень из кучки, отвечающей координате $1$ (и добавляют камень в следующую кучку, отвечающую координате $-1$).

# Идея избавления от остатков

Заводим $2N$ дополнительных кучек камней.

**Условие контроля**

Ограничимся только такими позициями, в которых ровно в одной из дополнительных кучек камней лежит ровно один камень.

Чтобы контролировать остатки, расширим векторы из разностных множеств координатами

$$0, \ldots, 0, 1, 0, \ldots, 0, -1, 0, \ldots, 0.$$

Если условие контроля выполнено, то возможны лишь те ходы, которые вынимают камень из кучки, отвечающей координате $1$ (и добавляют камень в следующую кучку, отвечающую координате $-1$).

# Построение обычной игры по модулярной

Из двумерной игры по модулю $2N$ с множествами разрешённых разностей $\mathcal{D}_j$, $0 \leqslant j < 2N$, построим $(2+2N)$-мерную игру с множеством разностей

$$\mathcal{D} = \left\{(a, 0^{2N}) + (0, 0, e_j - e_k) : a \in \mathcal{D}_j\right\},$$

где $e_j$ — $j$-й координатный вектор, а $k = j - a_1 - a_2 \pmod{2N}$.

# Соответствие между двоичными КА и играми

**Теорема**

Пусть по двоичному КА $C = (\{0,1\}, r, \delta)$ построена игра на вычитание с множеством разрешённых разностей $\mathcal{D}$ как описано выше (сначала модулярная, а из модулярной обычная).

Тогда при $|i| \leqslant Nt$ выполняется условие соответствия

$$c(t,i) = \mathrm{val}(Nt + i, Nt - i, \underbrace{0,0,\ldots,0,1}_{2N \text{ координат}}),$$

где $c_{t,i}$ — символ в $i$-й ячейке КА в момент времени $t$ при начальной конфигурации $c(0,0) = 0$, $c(0,i) = 1$ при $i \neq 0$; $\mathrm{val}(x)$ — оценка позиции $x$ в игре.

# Доказательство теоремы

Индукция по времени. Соотношения соответствия

$$\mathrm{val}((Nt + i, Nt - i, 0^{2N}) + (0, 0, e_{2N})) = c(t, i),$$

$$\mathrm{val}((Nt + i + j, Nt - i + j, 0^{2N}) + (0, 0, e_{2j}) = s_j, \quad 1 \leqslant j < N.$$

База индукции очевидна: в начальной конфигурации есть ровно один $0$, он соответствует оценке позиции $(0, \ldots, 0, 1)$.

Индуктивный переход: ходы из позиции

$$(Nt + i + j, Nt - i + j, 0^{2N}) + (0, 0, e_{2j})$$

ведут лишь в позиции такого же вида, но с меньшим значением времени.

# Трудность многомерных игр на вычитание

# Основной результат

**Теорема**

Для некоторой константы $d$ существует такая игра на вычитание с конечным множеством разрешённых разностей $\mathcal{D} \subset \mathbb{N}^d$, что любой алгоритм решения этой игры, принимающий на вход координаты позиции в двоичной записи, работает дольше $2^{n/11}$, где $n$ — длина входа.

**Замечание**

Тривиальный алгоритм решения игры на вычитание (построить граф позиций и применить рекурсивную оценку) работает за время $2^{O(dn)}$.

Теорема говорит, что существенно быстрее решать эти игры невозможно.

# Основной результат

**Теорема**

Для некоторой константы $d$ существует такая игра на вычитание с конечным множеством разрешённых разностей $\mathcal{D} \subset \mathbb{N}^d$, что любой алгоритм решения этой игры, принимающий на вход координаты позиции в двоичной записи, работает дольше $2^{n/11}$, где $n$ — длина входа.

**Замечание**

Тривиальный алгоритм решения игры на вычитание (построить граф позиций и применить рекурсивную оценку) работает за время $2^{O(dn)}$.

Теорема говорит, что существенно быстрее решать эти игры невозможно.

# Нижние оценки времени работы алгоритмов

Для доказательств алгоритмической неразрешимости применяется диагональный метод (далёкое обобщение парадокса лжеца).

Диагональный метод возможно обобщить на случай ограниченных по времени вычислений.

**Теорема об иерархии по времени, частный случай**

$\mathrm{DTIME}(2^{n/2}) \subset \mathrm{DTIME}(2^n)$ (включение строгое).

# Нижние оценки времени работы алгоритмов

Для доказательств алгоритмической неразрешимости применяется диагональный метод (далёкое обобщение парадокса лжеца).

Диагональный метод возможно обобщить на случай ограниченных по времени вычислений.

**Теорема об иерархии по времени, частный случай** $\mathrm{DTIME}(2^{n/2}) \subset \mathrm{DTIME}(2^n)$ (включение строгое).

# От теоремы об иерархии к трудности игр на вычитание

Общий план доказательства.

1. Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
2. Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (без существенной потери во времени работы).
3. По машине $U$ построим КА и соответствующую игру вычитания. Размерность игры $O(1)$.
4. Результат работы $M$ на входе $w$ определяется оценкой позиции игры на вычитание.
5. По выбору $L$ эту оценку невозможно вычислить слишком быстро.

# От теоремы об иерархии к трудности игр на вычитание

Общий план доказательства.

1. Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
2. Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (без существенной потери во времени работы).
3. По машине $U$ построим КА и соответствующую игру вычитания. Размерность игры $O(1)$.
4. Результат работы $M$ на входе $w$ определяется оценкой позиции игры на вычитание.
5. По выбору $L$ эту оценку невозможно вычислить слишком быстро.

# От теоремы об иерархии к трудности игр на вычитание

Общий план доказательства.

1. Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
2. Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (без существенной потери во времени работы).
3. По машине $U$ построим КА и соответствующую игру вычитания. Размерность игры $O(1)$.
4. Результат работы $M$ на входе $w$ определяется оценкой позиции игры на вычитание.
5. По выбору $L$ эту оценку невозможно вычислить слишком быстро.

# От теоремы об иерархии к трудности игр на вычитание

Общий план доказательства.

1. Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
2. Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (без существенной потери во времени работы).
3. По машине $U$ построим КА и соответствующую игру вычитания. Размерность игры $O(1)$.
4. Результат работы $M$ на входе $w$ определяется оценкой позиции игры на вычитание.
5. По выбору $L$ эту оценку невозможно вычислить слишком быстро.

# От теоремы об иерархии к трудности игр на вычитание

Общий план доказательства.

1. Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
2. Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (без существенной потери во времени работы).
3. По машине $U$ построим КА и соответствующую игру вычитания. Размерность игры $O(1)$.
4. Результат работы $M$ на входе $w$ определяется оценкой позиции игры на вычитание.
5. По выбору $L$ эту оценку невозможно вычислить слишком быстро.

# МТ, которая работает на всех входах сразу

По МТ $M$, работающей за время $T(n) = C \cdot 2^n$, $C$ — константа, строим МТ $U$, которая работает на пустом входе следующим образом.

Работа машины $U$ разбивается на этапы, а состояние её ленты — на зоны.

# МТ, которая работает на всех входах сразу

По МТ $M$, работающей за время $T(n) = C \cdot 2^n$, $C$ — константа, строим МТ $U$, которая работает на пустом входе следующим образом.

Работа машины $U$ разбивается на этапы, а состояние её ленты — на зоны.

Вид состояния зоны перед началом этапа:

# МТ, которая работает на всех входах сразу

По МТ $M$, работающей за время $T(n) = C \cdot 2^n$, $C$ — константа, строим МТ $U$, которая работает на пустом входе следующим образом.

Работа машины $U$ разбивается на этапы, а состояние её ленты — на зоны.

Вид состояния зоны перед началом этапа:


Алфавит $U$ содержит $(Q \cup \{\Lambda\}) \times A$, где $Q$ — множество состояний $M$, $A$ — алфавит $M$.

# МТ, которая работает на всех входах сразу

По МТ $M$, работающей за время $T(n) = C \cdot 2^n$, $C$ — константа, строим МТ $U$, которая работает на пустом входе следующим образом.

Работа машины $U$ разбивается на этапы, а состояние её ленты — на зоны.

Перед началом этапа $k$ на ленте машины $U$ записаны $k-1$ зон, отвечающих первым входам $w_1$, $w_2$, $\ldots$, $w_{k-1}$.
На зоне $i$ записана конфигурация работы $M$ на входе $w_i$ после $k-1-i$ тактов (в последней зоне записана начальная конфигурация $M$).

# МТ, которая работает на всех входах сразу (этап $k$)

1. $U$ перемещается по зонам и в каждой выполняет один такт работы $M$.
2. В конец дописывается свежая зона.

**Наблюдение**

В первом блоке зоны $p$ после $p + T(|w_p|)$ этапов записан результат работы $M$ на входе $w_p$.

# МТ, которая работает на всех входах сразу (этап $k$)

1. $U$ перемещается по зонам и в каждой выполняет один такт работы $M$.
2. В конец дописывается свежая зона.

| $1$ | $n$ | $n + 2T(n)$ |
|---|---|---|
| $0$ | $w_k$ | $(\Lambda, \Lambda) \dots (\Lambda, \Lambda)(q_0, w_{k,1})(\Lambda, w_{k,2}) \dots (\Lambda, w_{k,n})(\Lambda, \Lambda) \dots (\Lambda, \Lambda)$ |

**Наблюдение**

В первом блоке зоны $p$ после $p + T(|w_p|)$ этапов записан результат работы $M$ на входе $w_p$.

# МТ, которая работает на всех входах сразу (этап $k$)

1. $U$ перемещается по зонам и в каждой выполняет один такт работы $M$.
2. В конец дописывается свежая зона.

| 1 | $n$ | $n + 2T(n)$ |
|---|---|---|
| $0$ | $w_k$ | $(\Lambda, \Lambda) \dots (\Lambda, \Lambda)(q_0, w_{k,1})(\Lambda, w_{k,2}) \dots (\Lambda, w_{k,n})(\Lambda, \Lambda) \dots (\Lambda, \Lambda)$ |

**Наблюдение**

В первом блоке зоны $p$ после $p + T(|w_p|)$ этапов записан результат работы $M$ на входе $w_p$.

# Оценки времени исполнения этапов

Можно построить $U$ так, чтобы выполнялись следующие утверждения.

**Утверждение 1**

Обновление конфигурации $M$ занимает время $O(L)$, где $L$ — размер зоны.

**Утверждение 2**

Добавление свежей зоны на этапе $k$ занимает время $O(nT(n))$, где $n$ — длина входа $w_k$.

# Оценки времени исполнения этапов

Можно построить $U$ так, чтобы выполнялись следующие утверждения.

**Утверждение 1**

Обновление конфигурации $M$ занимает время $O(L)$, где $L$ — размер зоны.

**Утверждение 2**

Добавление свежей зоны на этапе $k$ занимает время $O(nT(n))$, где $n$ — длина входа $w_k$.

# Доказательство утверждения 2

1. Запись блока результата требует времени $O(1)$.
2. Запись следующего слова $w_k$ во второй блок (копирование предыдущего входа и прибавление 1 в двоичной записи): время $O(nT(n))$.
3. Отметить пространство $n + 2T(n)$: время $O(nT(n))$.
4. Поместить в центр этого пространства слово $w_k$: время $O(nT(n))$.

---

# Доказательство утверждения 2

1. Запись блока результата требует времени $O(1)$.
2. Запись следующего слова $w_k$ во второй блок (копирование предыдущего входа и прибавление $1$ в двоичной записи): время $O(nT(n))$.
3. Отметить пространство $n + 2T(n)$: время $O(nT(n))$.
4. Поместить в центр этого пространства слово $w_k$: время $O(nT(n))$.

---

К п. 2: $n$ символов, каждый нужно перенести на расстояние $2n + 2T(n) = O(T(n))$.

# Доказательство утверждения 2

1. Запись блока результата требует времени $O(1)$.
2. Запись следующего слова $w_k$ во второй блок (копирование предыдущего входа и прибавление $1$ в двоичной записи): время $O(nT(n))$.
3. Отметить пространство $n + 2T(n)$: время $O(nT(n))$.
4. Поместить в центр этого пространства слово $w_k$: время $O(nT(n))$.

---

К п. 3: счётчик с начальным значением $n + 2T(n)$, двигается вправо, уменьшаясь каждый раз на $1$. Станет $0$ в конце зоны.
Длина счётчика $\leqslant 1 + \log(n + T(n)) = O(n)$.
Вычисление начального значения: время $O(n \log n) = o(nT(n))$.
Сдвиг и обновление: время $O(n)$.

# Доказательство утверждения 2

1. Запись блока результата требует времени $O(1)$.
2. Запись следующего слова $w_k$ во второй блок (копирование предыдущего входа и прибавление $1$ в двоичной записи): время $O(nT(n))$.
3. Отметить пространство $n + 2T(n)$: время $O(nT(n))$.
4. Поместить в центр этого пространства слово $w_k$: время $O(nT(n))$.

---

К п. 4: $n$ символов, каждый нужно перенести на расстояние $n + T(n) = O(T(n))$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 3**

Пусть $s = |w_k|$. Время исполнения этапа $k$ не превосходит $O(sT(s)^2)$.

**Доказательство**

Нужно обновить существующие зоны, по утв. 1 требует времени

$$O(k(s + T(s))),$$

и записать свежую зону, по утв. 2 требует времени

$$O(sT(s)).$$

Количество зон не больше $2^{|k|} = O(T(s))$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 3**

Пусть $s = |w_k|$. Время исполнения этапа $k$ не превосходит $O(sT(s)^2)$.

**Доказательство**

Нужно обновить существующие зоны, по утв. 1 требует времени

$$O(k(s + T(s))),$$

и записать свежую зону, по утв. 2 требует времени

$$O(sT(s)).$$

Количество зон не больше $2^{s+1} = O(T(s))$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 3**

Пусть $s = |w_k|$. Время исполнения этапа $k$ не превосходит $O(sT(s)^2)$.

**Доказательство**

Нужно обновить существующие зоны, по утв. 1 требует времени

$$O(k(s + T(s))),$$

и записать свежую зону, по утв. 2 требует времени

$$O(sT(s)).$$

Количество зон не больше $2^{s+1} = O(T(s))$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 3**

Пусть $s = |w_k|$. Время исполнения этапа $k$ не превосходит $O(sT(s)^2)$.

**Доказательство**

Нужно обновить существующие зоны, по утв. 1 требует времени

$$O(k(s + T(s))),$$

и записать свежую зону, по утв. 2 требует времени

$$O(sT(s)).$$

Количество зон не больше $2^{s+1} = O(T(s))$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 4**

При достаточно больших $n$ результат работы МТ $M$ на входе $w$ длины $n$ появляется на ленте машины $U$ не позднее, чем после $2^{4n}$ тактов работы.

**Доказательство**

Если считать время в этапах, то результат работы $M$ на входе $w$ появится не позже этапа $2^{n+1} + T(n)$.

Пересчитывая на такты с помощью утв. 3, получаем искомую оценку времени появления результата

$$O((2^{n+1} + T(n))nT(n)^3) = O(n2^{3n}) < 2^{4n}$$

при достаточно больших $n$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 4**

При достаточно больших $n$ результат работы МТ $M$ на входе $w$ длины $n$ появляется на ленте машины $U$ не позднее, чем после $2^{4n}$ тактов работы.

**Доказательство**

Если считать время в этапах, то результат работы $M$ на входе $w$ появится не позже этапа $2^{n+1} + T(n)$.

Пересчитывая на такты с помощью утв. 3, получаем искомую оценку времени появления результата

$$O\big((2^{n+1} + T(n))nT(n)^2\big) = O(n2^{3n}) < 2^{4n}$$

при достаточно больших $n$.

# Оценки времени исполнения этапов (продолжение)

**Утверждение 4**

При достаточно больших $n$ результат работы МТ $M$ на входе $w$ длины $n$ появляется на ленте машины $U$ не позднее, чем после $2^{4n}$ тактов работы.

**Доказательство**

Если считать время в этапах, то результат работы $M$ на входе $w$ появится не позже этапа $2^{n+1} + T(n)$.

Пересчитывая на такты с помощью утв. 3, получаем искомую оценку времени появления результата

$$O\big((2^{n+1} + T(n))nT(n)^2\big) = O(n2^{3n}) < 2^{4n}$$

при достаточно больших $n$.

# Доказательство трудности игр вычитания, п. 3

- $\surd$ Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
- $\surd$ Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (результат работы $M$ на входе $w$ длины $n$ появляется не позже $2^{4n}$ тактов работы).

3. По машине $U$ построим КА $C_U$. Размер окрестности $O(1)$. Помимо условий $(\Lambda, \Lambda) \mapsto 1^{\ell+2}$; $(q_0, \Lambda) \mapsto 1^\ell 01$ потребуем ещё

$$(\Lambda, 1) \mapsto 110^{\ell-1}1.$$

Это гарантирует, что в блоке результата третий бит равен 0 лишь тогда, когда над ним нет головки $U$ и машина $M$ принимает вход из этой зоны.

...

# Доказательство трудности игр вычитания, п. 3

- $\surd$ Выберем $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$ и такую МТ $M$, что $L(M) = L$, причём $M$ работает за время $O(2^n)$.
- $\surd$ Построим другую машину $U$, которая моделирует работу $M$ параллельно на всех входах (результат работы $M$ на входе $w$ длины $n$ появляется не позже $2^{4n}$ тактов работы).
- 3. По машине $U$ построим КА $C_U$. Размер окрестности $O(1)$. Помимо условий $(\Lambda, \Lambda) \mapsto 1^{\ell+2}$; $(q_0, \Lambda) \mapsto 1^\ell 01$ потребуем ещё

$$(\Lambda, 1) \mapsto 110^{\ell-1}1.$$

Это гарантирует, что в блоке результата третий бит равен 0 лишь тогда, когда над ним нет головки $U$ и машина $M$ принимает вход из этой зоны.

...

# Игра из автомата

По машине $U$ построили КА $C_U$. Размер окрестности $O(1)$.

По $C_U$ строим $d$-мерную игру вычитания с множеством разрешённых разностей $\mathcal{D}_U$. Здесь $d = O(1)$, так как параметр $N$ в конструкции игры зависит от функции переходов $\delta$, у которой $O(1)$ аргументов.

Условие согласования игры и автомата позволяет по $(t, i)$ выразить символ $c(t, i)$ на ленте $C_U$ через оценку позиции

$$\mathrm{val}(Nt + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

# Игра из автомата

По машине $U$ построили КА $C_U$. Размер окрестности $O(1)$.

По $C_U$ строим $d$-мерную игру вычитания с множеством разрешённых разностей $\mathcal{D}_U$. Здесь $d = O(1)$, так как параметр $N$ в конструкции игры зависит от функции переходов $\delta$, у которой $O(1)$ аргументов.

Условие согласования игры и автомата позволяет по $(t, i)$ выразить символ $c(t, i)$ на ленте $C_U$ через оценку позиции

$$\mathrm{val}(Nt + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.

1.1. $k = 2^n + \mathrm{bin}(w)$, $\mathrm{bin}(w)$ — число с двоичной записью $w$.

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.

1.1. $k = 2^n + \mathrm{bin}(w)$, $\mathrm{bin}(w)$ — число с двоичной записью $w$.

1.2.

$$i = L \sum_{\ell=0}^{n-1} 2^\ell (1 + 2\ell + 2^{\ell+1}) + L\,\mathrm{bin}(w)(1 + 2n + 2^{n+1}) + 3.$$

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.
2. Для $t_1 = 2^{4n}$, $t_2 = 2^{4n} + 1$, $t_3 = 2^{4n} + 2$ применить процедуру решения игры $\mathcal{D}_U$ для оценки позиций

$$\mathrm{val}(Nt_0 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$

$$\mathrm{val}(Nt_1 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$

$$\mathrm{val}(Nt_2 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.
2. Для $t_1 = 2^{4n}$, $t_2 = 2^{4n} + 1$, $t_3 = 2^{4n} + 2$ применить процедуру решения игры $\mathcal{D}_U$ для оценки позиций

$$\mathrm{val}(Nt_0 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$

$$\mathrm{val}(Nt_1 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$

$$\mathrm{val}(Nt_2 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

3. Если $\geqslant 2$ из этих оценок дают $0$, принять слово $w$.

# Алгоритм разрешения для языка $L$

1. Вычислить $i$: номер бита результата в зоне $k$, в которой $U$ записывает результаты работы $M$ на слове $w$.
3. Если $\geqslant 2$ из этих оценок дают $0$, принять слово $w$.

Замечание. Три момента $t_1 = 2^{4n}$, $t_2 = 2^{4n} + 1$, $t_3 = 2^{4n} + 2$ выбраны, чтобы избежать чрезмерной конкретизации машины $U$. Мы лишь предполагаем, что она возвращается в блок результата не раньше, чем через три такта работы (это её замедлит разве что втрое).

# Завершение доказательства

Размер описания позиций игры $\mathcal{D}_U$, для которых вызывается алгоритм решения игры, оценивается как $4n + o(n)$, то есть не больше $5n$.

Если время работы алгоритма решения игры $S(n)$, где $n$ — длина входа, то время работы описанного выше алгоритма разрешения языка $L$ оценивается как $\mathrm{poly}(n) + 3S(5n)$.

Если $S(n) < 2^{n/11}$, то получаем алгоритм разрешения языка $L$, работающий за время

$$< \mathrm{poly}(n) + 2^{5n/11} < 2^{n/2},$$

что противоречит выбору языка $L$.

# Завершение доказательства

Размер описания позиций игры $\mathcal{D}_U$, для которых вызывается алгоритм решения игры, оценивается как $4n + o(n)$, то есть не больше $5n$.

Если время работы алгоритма решения игры $S(n)$, где $n$ — длина входа, то время работы описанного выше алгоритма разрешения языка $L$ оценивается как $\mathrm{poly}(n) + 3S(5n)$.

Если $S(n) < 2^{n/11}$, то получаем алгоритм разрешения языка $L$, работающий за время

$$< \mathrm{poly}(n) + 2^{5n/11} < 2^{n/2},$$

что противоречит выбору языка $L$.

# Завершение доказательства

Размер описания позиций игры $\mathcal{D}_U$, для которых вызывается алгоритм решения игры, оценивается как $4n + o(n)$, то есть не больше $5n$.

Если время работы алгоритма решения игры $S(n)$, где $n$ — длина входа, то время работы описанного выше алгоритма разрешения языка $L$ оценивается как $\mathrm{poly}(n) + 3S(5n)$.

Если $S(n) < 2^{n/11}$, то получаем алгоритм разрешения языка $L$, работающий за время

$$< \mathrm{poly}(n) + 2^{5n/11} < 2^{n/2},$$

что противоречит выбору языка $L$.

# Отступление: о детерминированности бесконечных игр

# Детерминированность игры

**Определение**

Игра на выигрыш (результатов два: выиграл один из игроков) называется детерминированной, если у одного из игроков есть выигрывающая стратегия.

**Смелое предположение**

Каждая игра детерминирована.

«Обоснование»: если у первого игрока нет выигрывающей стратегии, значит, второй всегда ему может помешать выиграть. Но это и есть выигрывающая стратегия для второго игрока?

# Детерминированность игры

**Определение**

Игра на выигрыш (результатов два: выиграл один из игроков) называется детерминированной, если у одного из игроков есть выигрывающая стратегия.

**Смелое предположение**

Каждая игра детерминирована.

«Обоснование»: если у первого игрока нет выигрывающей стратегии, значит, второй всегда ему может помешать выиграть. Но это и есть выигрывающая стратегия для второго игрока?

# Детерминированность игры

**Определение**

Игра на выигрыш (результатов два: выиграл один из игроков) называется детерминированной, если у одного из игроков есть выигрывающая стратегия.

**Смелое предположение**

Каждая игра детерминирована.

«Обоснование»: если у первого игрока нет выигрывающей стратегии, значит, второй всегда ему может помешать выиграть. Но это и есть выигрывающая стратегия для второго игрока?

# Игра в закрашивание натурального ряда

## Правила игры

Игроки $0$ и $1$ по очереди красят натуральные числа в два цвета; игрок $0$ красит в красный, а игрок $1$ — в синий. Если уже закрашены числа от $0$ до $N$, то на следующем ходу игрок может закрасить в свой цвет любой конечный отрезок из оставшихся чисел: от $N+1$ до некоторого числа $N'$. Хотя бы одно число закрасить нужно.

## Результат партии

Итог партии: разбиение множества натуральных чисел на красные ($S_0$) и синие ($S_1$). Игрок $0$ выигрывает в партии, если множество $S_0$ принадлежит некоторому, заранее заданному, семейству $\mathrm{Win}_0$ подмножеств натурального ряда. Иначе выигрывает игрок $1$.

# «Большие» и «маленькие» подмножества

Называем множества из $\mathrm{Win}_0$ «большими», а остальные — маленькими.

Потребуем выполнения свойств:

1. любое конечное множество маленькое;
2. $X$ маленькое тогда и только тогда, когда $\mathbb{N} \setminus X$ большое;
3. если $X \subset Y$ и $X$ большое, то и $Y$ большое;
4. пересечение больших множеств большое.

**Утверждение**

Если $X \oplus Y$ конечно ($\oplus$ — симметрическая разность множеств), то $X$ и $Y$ одной величины (либо оба большие, либо оба маленькие).

# «Большие» и «маленькие» подмножества

Называем множества из $\mathrm{Win}_0$ «большими», а остальные — маленькими.

Потребуем выполнения свойств:

1. любое конечное множество маленькое;
2. $X$ маленькое тогда и только тогда, когда $\mathbb{N} \setminus X$ большое;
3. если $X \subset Y$ и $X$ большое, то и $Y$ большое;
4. пересечение больших множеств большое.

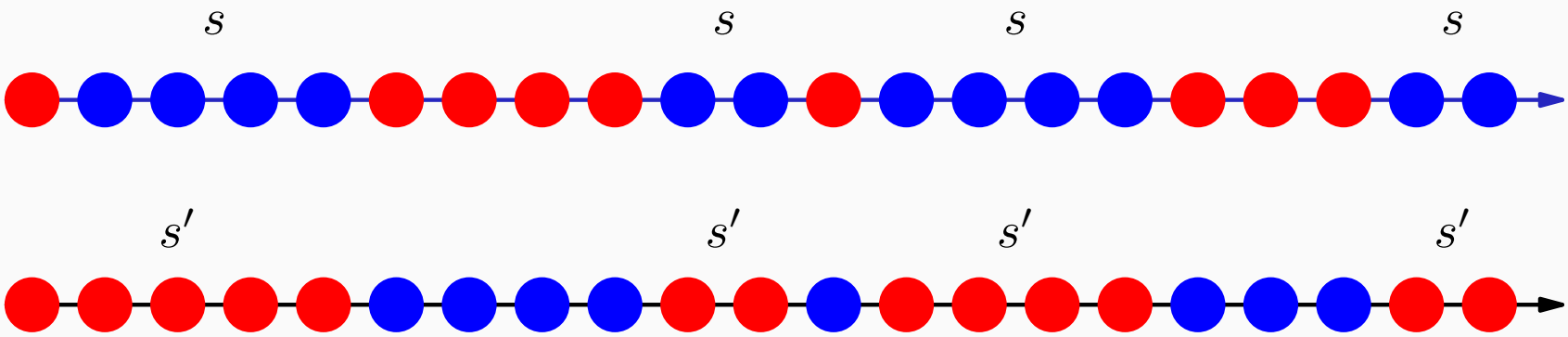
**Утверждение**

Если $X \oplus Y$ конечно ($\oplus$ — симметрическая разность множеств), то $X$ и $Y$ одной величины (либо оба большие, либо оба маленькие).

# Перехватывающая стратегия Красного

Пусть $s$ — выигрывающая стратегия Синего.

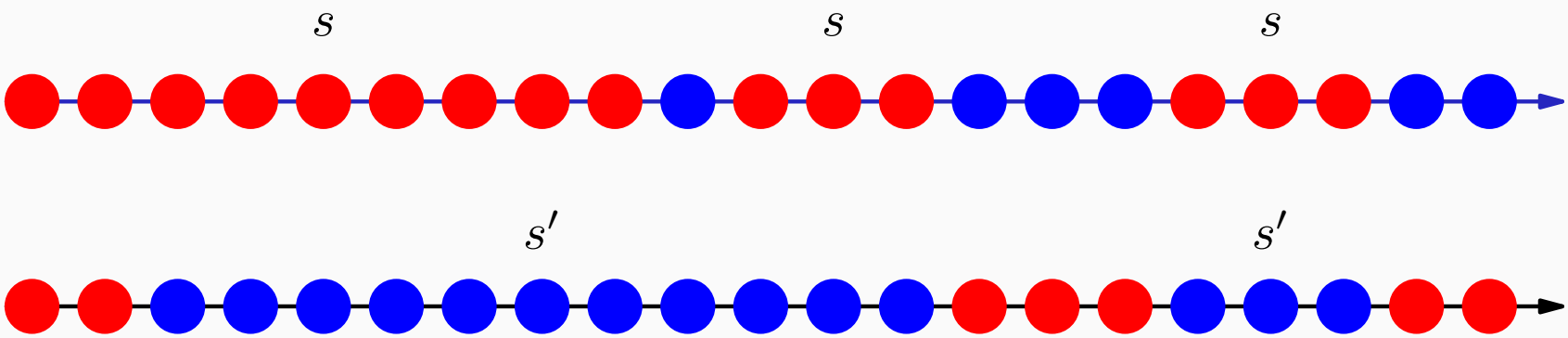
Стратегия $s'$ Красного: посмотреть, какой ход предписывает Синему стратегия $s$ в ответ на первый ход Красного, закрашивающий только 0. Пусть Синий должен закрасить $N_0$ чисел. Сделать ход, который закрашивает числа от 0 до $N_0$. Далее следовать стратегии $s$ Синего.



# Перехватывающая стратегия Синего

Пусть $s$ — стратегия Красного, которая первым ходом закрашивает числа от $0$ до $N_0$, а в ответ на ход Синего, закрашивающий отрезок $[N_0 + 1, N_1]$, закрашивает числа от $N_1 + 1$ до $N_2(N_1)$.

Стратегия $s'$ Синего: в ответ на ход $[0, X]$ Красного закрасить отрезок $[X + 1, N_2(X - N_0)]$ отрезок, если $X > N_0$, и отрезок $[X + 1, N_2(N_0 + 1)]$, если $X \leqslant N_0$. Далее следовать стратегии Красного.



# Недетерминированность

В обоих случаях перехватывающая стратегия гарантирует, что красное множество, возникающее в партии, в которой один игрок придерживается стратегии $s$, лишь на конечное множество отличается от синего множества, возникающего в партии, в которой другой игрок придерживается перехватывающей стратегии $s'$.

Если $s$ — выигрывающая стратегия для одного игрока, то перехватывающая стратегия $s'$ будет выигрывающей для другого.

У обоих игроков выигрывающей стратегии быть не может.

# Непростой вопрос

Существует ли разбиение на большие и маленькие множества, удовлетворяющее указанным выше свойствам? (Такие разбиения называются неглавными ультрафильтрами.)

# Непростой вопрос

Существует ли разбиение на большие и маленькие множества, удовлетворяющее указанным выше свойствам? (Такие разбиения называются неглавными ультрафильтрами.)

Ответ зависит от аксиоматики теории множеств.

# Непростой вопрос

Существует ли разбиение на большие и маленькие множества, удовлетворяющее указанным выше свойствам? (Такие разбиения называются неглавными ультрафильтрами.)

Если принять аксиому выбора и основанную на ней трансфинитную индукцию, то ответ положительный.

**Задача**

Используя трансфинитную индукцию, докажите существование неглавного ультрафильтра.

# Непростой вопрос

Существует ли разбиение на большие и маленькие множества, удовлетворяющее указанным выше свойствам? (Такие разбиения называются неглавными ультрафильтрами.)

**Аксиома детерминированности**

Всякая игра на закрашивание натурального ряда детерминирована.

Эта аксиома противоречит аксиоме выбора. Но на её основе можно построить интересную теорию множеств: без неглавных ультрафильтров, но с измеримостью любого множества действительных чисел.

# Непростой вопрос

Существует ли разбиение на большие и маленькие множества, удовлетворяющее указанным выше свойствам? (Такие разбиения называются неглавными ультрафильтрами.)

Далее нам эти тонкости не понадобятся. В играх, которые мы будем изучать, всегда есть выигрышная стратегия у одного из игроков. Более того, эта стратегия может быть выбрана позиционной (memoryless) или стратегией с конечной памятью (выбор хода зависит лишь от конечной информации и начале партии, размер этой информации не зависит от длины начального отрезка партии).