ЛИКБЕЗ
Лекция 1: Машины Тьюринга. Основы теории вычислимости. Булевы функции и пропозициональные формулы

Дмитрий Ицыксон

ПОМИ РАН

21 сентября 2008

План

- O-символика и ассимптотические классы функций
- Машины Тьюринга
- Элементы теории вычислимости: разрешимые и перечислимые языки
- Булевы функции и пропозициональные формулы

Литература

1. Н. К. Верещагин, А. Шень. Вычислимые функции.
2. Т. Кормен, Ч. Лейзерсон, Р. Ривест. Алгоритмы. Построение и анализ.

O, o, Ω, Θ

Обозначения

Функции $f, g : \mathbb{N} \to \mathbb{R}_+$

- $f(n) = O(g(n))$, если $\exists c > 0 \exists n_0 \forall n > n_0, f(n) < cg(n)$;
- $f(n) = o(g(n))$, если $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = 0$;
- $f(n) = \Omega(g(n))$, если $\exists c > 0 \exists n_0 \forall n > n_0, f(n) > cg(n)$;
- $f(n) = \Theta(g(n))$, если
  $\exists c_1, c_2 > 0 \exists n_0 \forall n > n_0, c_1 g(n) < f(n) < c_2 g(n)$;

$$f(n) = \Theta(g(n)) \iff \begin{cases} f(n) = O(g(n)) \\ g(n) = O(f(n)) \end{cases}$$

Полезно знать

- $f(n), g(n)$ — многочлены, $\deg f = \deg g \implies f = \Theta(g)$;
- $f(n), g(n)$ — многочлены, $\deg f < \deg g \implies f = o(g)$;
- $\log_2 n = o(n^k)$;
- $\log_2 n = \Theta(\log_a n) = \Theta($ количество цифр в числе $n)$;
- $n^k = o(2^n)$;
- $n^k = o(n^{\log n})$;
- $O(1)$ — это константа;
- $o(1)$ — это бесконечно малая функция.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Классы функций

- *poly* — класс функций не более, чем полиномиального роста.
- $f \in poly$, если существует $k \in \mathbb{N}$, что $f = O(n^k)$.
- Класс *poly* замкнут относительно суммы, произведения и композиции функций.
- *exp* — класс функций не более, чем экспоненциального роста.
- $f \in exp$, если существует $g \in poly$, что $f = O(2^{g(n)})$.
- Класс *exp* замкнут оносительно суммы, произведения и композицией с элементами класса *poly*.
- *log* — класс функций не более, чем логарифмического роста.
- $f \in log$, если $f = O(logn)$.

Модели вычислений

Зачем они нужны?

- Математическое определение понятия алгоритм;
- Строгое определение сложности алгоритма;
- Возможность что-то доказывать про все алгоритмы. Доказывать невозможность алгоритма.

Какие Вы знаете модели вычисления?

λ-исчисление, машина Тьюринга, PAM-машина, машина Поста, нормальные алгоритмы Маркова...
Почти любой язык программирования может выступать в роли модели вычисления, если есть возможность использовать неограниченное количество памяти.

Модели вычислений

Зачем они нужны?

- Математическое определение понятия алгоритм;
- Строгое определение сложности алгоритма;
- Возможность что-то доказывать про все алгоритмы. Доказывать невозможность алгоритма.

Какие Вы знаете модели вычисления?

λ-исчисление, машина Тьюринга, PAM-машина, машина Поста, нормальные алгоритмы Маркова...
Почти любой язык программирования может выступать в роли модели вычисления, если есть возможность использовать неограниченное количество памяти.

Алфавит

Алфавит — это некоторое конечное множество символов.
Стандартное обозначение: Σ.

Слова в алфавите

Слово — это конечная последовательность символов. Если Σ — алфавит, то Σ^* — множество всех слов в алфавите.
Например: $\Sigma = \{a, b\}$, то $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \dots\}$, где λ — пустое слово.

Языки и функции

Язык

Языком над алфавитом Σ называется подмножество Σ^*.

Примеры языков

Язык простых чисел, язык четных чисел, язык двудольных графов, язык выполнимых формул в КНФ и т.д.

Соглашение

Считаем, что любой алгоритм либо проверяет принадлежность входного слова некоторому языку, либо вычисляет значение функции $\Sigma^* \to \Sigma^*$.

Машина Тьюринга

- Бесконечная в одну сторону лента, разделенная на ячейки. В самой левой ячейке написан символ $\triangleright$.
- Q — конечное множество состояний. $q_0 \in Q$ — начальное состояние. $q_f \in Q$ — конечное состояние.
- Σ — алфавит символов, которые могут быть записаны на ленте. $\triangleright$, " _ " $\in \Sigma$.
- Головка машины указывает на одну из ячеек ленты
- Правило перехода: $\delta : \Sigma \times Q \to \Sigma \times Q \times \{\rightarrow, \leftarrow, \bullet\}$
- Согласно правилу перехода машина по символу, на который указывает головка, и по текущему состоянию, пишет на это место новый символ, переходит в новое состояние и, возможно, сдвигает головку на 1 символ влево или вправо.
- Начинает работу в состоянии q_0, головка указывает на первый символ. Заканчивает в состоянии q_f.

Пример

Заменить число в двоичной системе счисления на его остаток при делении на 2.

- $(q_0, \frac{0}{1}) \mapsto (q_0, \frac{0}{1}, \rightarrow)$;
- $(q_0, \_) \mapsto (q_1, \_, \leftarrow)$;
- $(q_1, 0) \mapsto (q_2, \_, \leftarrow)$;
- $(q_1, 1) \mapsto (q_3, \_, \leftarrow)$;
- $(\frac{q_2}{q_3}, \frac{0}{1}) \mapsto (\frac{q_2}{q_3}, \_, \leftarrow)$;
- $(\frac{q_2}{q_3}, \triangleright) \mapsto (\frac{q_2}{q_3}, \triangleright, \rightarrow)$;
- $(q_2, \_) \mapsto (q_f, 0, \bullet)$;
- $(q_3, \_) \mapsto (q_f, 1, \bullet)$.

Пример

Делится ли число в 2-ой системе счисления на 3?

- $(q_0, 0) \mapsto (q_0, 0, \rightarrow)$;
- $(q_0, 1) \mapsto (q_1, 1, \rightarrow)$;
- $(q_1, 0) \mapsto (q_2, 0, \rightarrow)$;
- $(q_1, 1) \mapsto (q_0, 1, \rightarrow)$;
- $(q_2, 0) \mapsto (q_1, 0, \rightarrow)$;
- $(q_2, 1) \mapsto (q_2, 1, \rightarrow)$;
- $(q_0, \_) \mapsto (q_{yes}, \_, \bullet)$;
- $(q_1, \_) \mapsto (q_{no}, \_, \bullet)$;
- $(q_2, \_) \mapsto (q_{no}, \_, \bullet)$.

Машина Тьюринга

Вход машины Тьюринга — то, что записано на ленте. За входом следует бесконечное число пробелов.
Выход машины Тьюринга — то, что записано на ленте после того, как машина пришла в конечное состояние.
Если машина Тьюринга проверяет принадлежность языку, то удобно иметь два конечных состояния: q_{yes} и q_{no}.
Машина Тьюринга может:

- закончить работу;
- работать бесконечно.

Сложностные параметры

Время

- Временем работы машины Тьюринга на входе x называем количество шагов, которое машина делает, чтобы прийти в конечное состояние.
- Временной сложностью машины Тьюринга называем максимум по всем входам длины n времени работы машины на этих входах.

Память

- Сложностью по памяти работы машины Тьюринга на входе x называем количество ячеек, в которых побывала головка машины.
- Емкостной сложностью машины Тьюринга называем максимум по всем входам длины n сложности по памяти работы машины на этих входах.

Многоленточная машина Тьюринга

Есть k лент, головка есть на каждой ленте.
Правило перехода: $\delta : \Sigma^k \times Q \to \Sigma^k \times Q \times \{\rightarrow, \leftarrow, \bullet\}^k$

Факт

По любой многоленточной машине Тьюринга можно построить одноленточную машину Тьюринга, вычисляющую ту же функцию. Причем сложностные характеристики этой машины будут лишь в полином раз хуже.

Замечание

Иногда удобно считать, что машина снабжена специальной входной лентой, доступной только для чтения и выходной лентой, доступной для записи, но без исправлений.

Недетерминированная машина Тьюринга

- Правила перехода неоднозначны. Возможно, что существует несколько правил для одной пары (символ, состояние);
- НМТ принимает слово x, если существует последовательность легальных шагов, приводящих в состояние q_{yes};
- Определение не симметрично относительно ответов *yes* и *no*;
- Можно считать, что машина снабжена дополнительной лентой, доступной только для чтения слева направо. На этой ленте записана подсказка (какое из правил сейчас применять). Машина принимает x, если существует подсказка, с которой она попадет в состояние q_{yes};
- Можно доказать, что если некоторая НМТ проверяет принадлежность языку L, то существует и детерминированная МТ, проверяющая принадлежность языку L.

Недетерминированная машина Тьюринга

- Правила перехода неоднозначны. Возможно, что существует несколько правил для одной пары (символ, состояние);
- НМТ принимает слово x, если существует последовательность легальных шагов, приводящих в состояние q_{yes};
- Определение не симметрично относительно ответов *yes* и *no*;
- Можно считать, что машина снабжена дополнительной лентой, доступной только для чтения слева направо. На этой ленте записана подсказка (какое из правил сейчас применять). Машина принимает x, если существует подсказка, с которой она попадет в состояние q_{yes};
- Можно доказать, что если некоторая НМТ проверяет принадлежность языку L, то существует и детерминированная МТ, проверяющая принадлежность языку L.

Недетерминированная машина Тьюринга

- Правила перехода неоднозначны. Возможно, что существует несколько правил для одной пары (символ, состояние);
- НМТ принимает слово x, если существует последовательность легальных шагов, приводящих в состояние q_{yes};
- Определение не симметрично относительно ответов *yes* и *no*;
- Можно считать, что машина снабжена дополнительной лентой, доступной только для чтения слева направо. На этой ленте записана подсказка (какое из правил сейчас применять). Машина принимает x, если существует подсказка, с которой она попадет в состояние q_{yes};
- Можно доказать, что если некоторая НМТ проверяет принадлежность языку L, то существует и детерминированная МТ, проверяющая принадлежность языку L.

Недетерминированная машина Тьюринга

- Правила перехода неоднозначны. Возможно, что существует несколько правил для одной пары (символ, состояние);
- НМТ принимает слово x, если существует последовательность легальных шагов, приводящих в состояние q_{yes};
- Определение не симметрично относительно ответов *yes* и *no*;
- Можно считать, что машина снабжена дополнительной лентой, доступной только для чтения слева направо. На этой ленте записана подсказка (какое из правил сейчас применять). Машина принимает x, если существует подсказка, с которой она попадет в состояние q_{yes};
- Можно доказать, что если некоторая НМТ проверяет принадлежность языку L, то существует и детерминированная МТ, проверяющая принадлежность языку L.

Недетерминированная машина Тьюринга

- Правила перехода неоднозначны. Возможно, что существует несколько правил для одной пары (символ, состояние);
- НМТ принимает слово x, если существует последовательность легальных шагов, приводящих в состояние q_{yes};
- Определение не симметрично относительно ответов *yes* и *no*;
- Можно считать, что машина снабжена дополнительной лентой, доступной только для чтения слева направо. На этой ленте записана подсказка (какое из правил сейчас применять). Машина принимает x, если существует подсказка, с которой она попадет в состояние q_{yes};
- Можно доказать, что если некоторая НМТ проверяет принадлежность языку L, то существует и детерминированная МТ, проверяющая принадлежность языку L.

Сложностные параметры НМТ

Время и память НМТ считаются, как максимум по всем вариантам применения правила до прихода в конечное состояние. Если машина не останавливается при каком-то выборе правил, то время работы считается бесконечным.

Тезис Черча-Тьюринга

Любой алгоритм можно реализовать в виде машины Тьюринга.

Элементы теории вычислимости

- Машину Тьюринга можно записать: алфавит, состояния, правила... Каждой МТ соответствует строчка в некотором алфавите.
- Машин Тьюринга счетное число.
- Универсальная машина Тьюринга. Существует такая машина Тьюринга U, которая по записи машины Тьюринга M и входу x моделирует поведение машины M на входе x. При этом сложностные параметры машины U не более, чем в полином от записи M и $|x|$ хуже.

Элементы теории вычислимости

- Машину Тьюринга можно записать: алфавит, состояния, правила... Каждой МТ соответствует строчка в некотором алфавите.
- Машин Тьюринга счетное число.
- Универсальная машина Тьюринга. Существует такая машина Тьюринга U, которая по записи машины Тьюринга M и входу x моделирует поведение машины M на входе x. При этом сложностные параметры машины U не более, чем в полином от записи M и $|x|$ хуже.

Элементы теории вычислимости

- Машину Тьюринга можно записать: алфавит, состояния, правила... Каждой МТ соответствует строчка в некотором алфавите.
- Машин Тьюринга счетное число.
- Универсальная машина Тьюринга. Существует такая машина Тьюринга U, которая по записи машины Тьюринга M и входу x моделирует поведение машины M на входе x. При этом сложностные параметры машины U не более, чем в полином от записи M и $|x|$ хуже.

Разрешимые и перечислимые языки

Σ — алфавит, $L \subset \Sigma^*$ — язык.
Язык L называется алгоритмически разрешимым, если существует такая машина Тьюринга M, что

$$\begin{cases} x \in L \iff M(x) \text{ останавливается в состоянии } q_{yes} \\ x \notin L \iff M(x) \text{ останавливается в состоянии } q_{no} \end{cases}$$

Язык L называется перечислимым, если существует такая машина Тьюринга M, что

$$\begin{cases} x \in L \iff M(x) \text{ останавливается в состоянии } q_{yes} \\ x \notin L \iff M(x) \text{ не останавливается} \end{cases}$$

Разрешимые и перечислимые языки

Σ — алфавит, $L \subset \Sigma^*$ — язык.

Язык L называется алгоритмически разрешимым, если существует такая машина Тьюринга M, что

$$\begin{cases} x \in L \iff M(x) \text{ останавливается в состоянии } q_{yes} \\ x \notin L \iff M(x) \text{ останавливается в состоянии } q_{no} \end{cases}$$

Язык L называется перечислимым, если существует такая машина Тьюринга M, что

$$\begin{cases} x \in L \iff M(x) \text{ останавливается в состоянии } q_{yes} \\ x \notin L \iff M(x) \text{ не останавливается} \end{cases}$$

Простейшие свойства

Лемма 1

Любой разрешимый язык является перечислимым.

Достаточно состояние q_{no} заменить на q_∞ и добавить правила $(q_\infty, *) \mapsto (q_\infty, *, \rightarrow)$

Лемма 2

Язык L — перечислим $\iff$ существует МТ M', которая, работая на пустом входе, рано или поздно напечатает любой элемент языка L без повторений. (Машина M' может работать бесконечно долго).

$\Leftarrow$ Машина M будет ждать, пока M' напечатает слово x.

$\Rightarrow$ M' моделирует M: 1 шаг на первом входе, 2 шага на первом, 2 шага на втором, 3 шага на первом, втором, третьем, 4 шага...

Простейшие свойства

Лемма 1

Любой разрешимый язык является перечислимым.

Достаточно состояние q_{no} заменить на q_∞ и добавить правила $(q_\infty, *) \mapsto (q_\infty, *, \rightarrow)$

Лемма 2

Язык L — перечислим $\iff$ существует МТ M', которая, работая на пустом входе, рано или поздно напечатает любой элемент языка L без повторений. (Машина M' может работать бесконечно долго).

$\Leftarrow$ Машина M будет ждать, пока M' напечатает слово x.

$\Rightarrow$ M' моделирует M: 1 шаг на первом входе, 2 шага на первом, 2 шага на втором, 3 шага на первом, втором, третьем, 4 шага...

Простейшие свойства

Лемма 1

Любой разрешимый язык является перечислимым.

Достаточно состояние q_{no} заменить на q_∞ и добавить правила $(q_\infty, *) \mapsto (q_\infty, *, \rightarrow)$

Лемма 2

Язык L — перечислим $\iff$ существует МТ M', которая, работая на пустом входе, рано или поздно напечатает любой элемент языка L без повторений. (Машина M' может работать бесконечно долго).

$\Leftarrow$ Машина M будет ждать, пока M' напечатает слово x.

$\Rightarrow$ M' моделирует M: 1 шаг на первом входе, 2 шага на первом, 2 шага на втором, 3 шага на первом, втором, третьем, 4 шага...

Простейшие свойства

Лемма 1

Любой разрешимый язык является перечислимым.

Достаточно состояние q_{no} заменить на q_∞ и добавить правила $(q_\infty, *) \mapsto (q_\infty, *, \rightarrow)$

Лемма 2

Язык L — перечислим $\iff$ существует МТ M', которая, работая на пустом входе, рано или поздно напечатает любой элемент языка L без повторений. (Машина M' может работать бесконечно долго).

$\Leftarrow$ Машина M будет ждать, пока M' напечатает слово x.

$\Rightarrow$ M' моделирует M: 1 шаг на первом входе, 2 шага на первом, 2 шага на втором, 3 шага на первом, втором, третьем, 4 шага...

Простейшие свойства

Лемма 3

Если язык L и его дополнение $\overline{L} = \Sigma^* \setminus L$ перечислимы, то L алгоритмически разрешим.

Параллельно запускаем алгоритм для L и $\overline{L}$, один из них должен остановиться.

Простейшие свойства

Лемма 3

Если язык L и его дополнение $\overline{L} = \Sigma^* \setminus L$ перечислимы, то L алгоритмически разрешим.

Параллельно запускаем алгоритм для L и $\overline{L}$, один из них должен остановиться.

Вопросы

Существуют ли неперечислимые языки?

Существуют, так как машин Тьюринга счетно, а языков континуум.

Неконструктивное доказательство

Существуют ли алгоритмически неразрешимые, но перечислимые языки?

Да.

Вопросы

Существуют ли неперечислимые языки?

Существуют, так как машин Тьюринга счетно, а языков континуум.

Неконструктивное доказательство

Существуют ли алгоритмически неразрешимые, но перечислимые языки?

Да.

Вопросы

Существуют ли неперечислимые языки?

Существуют, так как машин Тьюринга счетно, а языков континуум.

Неконструктивное доказательство

Существуют ли алгоритмически неразрешимые, но перечислимые языки?

Да.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример неперечислимого языка

- Все записи машин Тьюринга можно перенумеровать с помощью алгоритма.
- Запись $< n >$ обозначает МТ с номером n.
- Рассмотрим язык
  $L = \{n | < n > \text{ не останавливается на входе } n\}$
- Пусть L перечислим алгоритмом с номером k.
- Если $k \in L$, то $< k >$ не останавливается на $k \implies < k >$ не перечисляет L.
- Если $k \notin L$, то $< k >$ останавливается на $k \implies < k >$ не перечисляет L.
- Противоречие. Значит L не перечисляется никаким алгоритмом.
- L — алгоритмически неразрешим.

Пример перечислимого, но не разрешимого языка

- $\overline{L} = \{n | < n > \text{останавливается на входе } n\}$;
- $\overline{L}$ неразрешим, так как иначе и язык L был бы разрешимым;
- $\overline{L}$ перечислим: моделируем машину $< n >$ на входе n и ждем, пока она остановится.

Пример перечислимого, но не разрешимого языка

- $\overline{L} = \{n | < n > \text{останавливается на входе } n\}$;
- $\overline{L}$ неразрешим, так как иначе и язык L был бы разрешимым;
- $\overline{L}$ перечислим: моделируем машину $< n >$ на входе n и ждем, пока она остановится.

Пример перечислимого, но не разрешимого языка

- $\overline{L} = \{n | < n > \text{останавливается на входе } n\}$;
- $\overline{L}$ неразрешим, так как иначе и язык L был бы разрешимым;
- $\overline{L}$ перечислим: моделируем машину $< n >$ на входе n и ждем, пока она остановится.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Комментарий

- Задача остановки МТ: по МТ и ее входу определить, остановится она или нет. Эта задача алгоритмически неразрешима;
- Метод доказательства: диагонализация;
- Все результаты об алгоритмической неразрешимости используют неразрешимость задачи остановки МТ.

Великая теорема Ферма

Если бы задача остановки была бы разрешима, то можно было бы доказать Великую теорему Ферма так:

1. Машина M на пустом входе перебирает все $x, y, z, n \in \mathbb{N}, n > 2$ и останавливается, если $x^n + y^n = z^n$.
2. Узнаем, останавливается ли машина M на пустом входе.

Язык пропозициональных формул

Γ — бесконечное множество пропозициональных переменных.
$\Gamma = \{x_1, x_2, x_3, \dots\}$.

- Пропозициональная переменная является пропозициональной формулой
- Если A — пропозициональная формула, то (A) — тоже пропозициональная формула.
- Если A — пропозициональная формула, то $\neg A$ — тоже пропозициональная формула.
- Если A, B — пропозициональные формулы, то $A \wedge B$ — тоже пропозициональная формула.
- Если A, B — пропозициональные формулы, то $A \vee B$ — тоже пропозициональная формула.
- Если A, B — пропозициональные формулы, то $A \to B$ — тоже пропозициональная формула.

Примеры пропозициональных формул

- x_1;
- $x_1 \wedge \neg x_1$;
- $x_1 \vee \neg x_1$;
- $(x_1 \vee x_2) \rightarrow x_3$;
- $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_2)$;
- $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (x_3 \wedge x_2)$;

Таблицы истинности

$\neg$	
a	$\neg a$
0	1
1	0

$\vee$		
a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

$\wedge$		
a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

$\rightarrow$		
a	b	$a \rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

Интерпретация

Пусть φ — пропозициональная формула с переменными $x_1, x_2, \ldots, x_n$.
Интерпретацией формулы φ называется отображение $\sigma : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$.
Значение формулы $I_\sigma(\varphi)$ при заданной интерпретации определяется индуктивно по построению формулы:

- $I_\sigma(x_i) = \sigma(x_i)$;
- $I_\sigma((A)) = I_\sigma(A)$;
- $I_\sigma(\neg A) = \neg I_\sigma(A)$;
- $I_\sigma(A \wedge B) = I_\sigma(A) \wedge I_\sigma(B)$;
- $I_\sigma(A \vee B) = I_\sigma(A) \vee I_\sigma(B)$;
- $I_\sigma(A \to B) = I_\sigma(A) \to I_\sigma(B)$.

Булевы функции

Определение. Булевой функцией мы называем функцию из $\{0,1\}^n$ в $\{0,1\}$.

Замечание. Каждая пропозициональная формула от n переменных задает булеву функцию из $\{0,1\}^n$ в $\{0,1\}$.

Примеры.

- Parity (четность): $f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n \bmod 2$
- Majority (большинство):

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 + x_2 + \cdots + x_n \geq \dfrac{n}{2} \\ 0, & \text{если } x_1 + x_2 + \cdots + x_n < \dfrac{n}{2} \end{cases}$$

- $f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } \overline{x_1 x_2 \ldots x_n} \text{ — простое число} \\ 0, & \text{иначе} \end{cases}$

Булевы функции

Определение. Булевой функцией мы называем функцию из $\{0,1\}^n$ в $\{0,1\}$.

Замечание. Каждая пропозициональная формула от n переменных задает булеву функцию из $\{0,1\}^n$ в $\{0,1\}$.

Примеры.

- Parity (четность): $f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n \bmod 2$
- Majority (большинство):

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 + x_2 + \cdots + x_n \geq \dfrac{n}{2} \\ 0, & \text{если } x_1 + x_2 + \cdots + x_n < \dfrac{n}{2} \end{cases}$$

- $f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } \overline{x_1 x_2 \ldots x_n} \text{ — простое число} \\ 0, & \text{иначе} \end{cases}$

Булевы функции

Определение. Булевой функцией мы называем функцию из $\{0,1\}^n$ в $\{0,1\}$.

Замечание. Каждая пропозициональная формула от n переменных задает булеву функцию из $\{0,1\}^n$ в $\{0,1\}$.

Примеры.

- Parity (четность): $f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n \bmod 2$
- Majority (большинство):

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 + x_2 + \cdots + x_n \geq \dfrac{n}{2} \\ 0, & \text{если } x_1 + x_2 + \cdots + x_n < \dfrac{n}{2} \end{cases}$$

- $f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } \overline{x_1 x_2 \ldots x_n} \text{ — простое число} \\ 0, & \text{иначе} \end{cases}$

Булевы функции

Определение. Булевой функцией мы называем функцию из $\{0,1\}^n$ в $\{0,1\}$.

Замечание. Каждая пропозициональная формула от n переменных задает булеву функцию из $\{0,1\}^n$ в $\{0,1\}$.

Примеры.

- Parity (четность): $f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n \bmod 2$
- Majority (большинство):

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 + x_2 + \cdots + x_n \geq \dfrac{n}{2} \\ 0, & \text{если } x_1 + x_2 + \cdots + x_n < \dfrac{n}{2} \end{cases}$$

- $f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{если } \overline{x_1 x_2 \ldots x_n} \text{ — простое число} \\ 0, & \text{иначе} \end{cases}$

КНФ и ДНФ

Определение. Литералом называется пропозициональная переменная или ее отрицание: x_i, $\neg x_i$.

Определение. Дизъюнктом или клозом называется дизъюнкция нескольких (возможно одного) литералов: $(x_1 \vee \neg x_2 \vee x_3)$.

Определение. Формулой в КНФ называется конъюнкция нескольких (возможно одного) дизъюнктов: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \neg x_1$.

Определение. Конъюнктом или мономом называется конъюнкция нескольких (возможно одного) литералов: $(x_1 \wedge \neg x_2 \wedge x_3)$.

Определение. Формулой в ДНФ называется дизъюнкция нескольких (возможно одного) конъюнктов: $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \neg x_1$.

КНФ и ДНФ

Определение. Литералом называется пропозициональная переменная или ее отрицание: x_i, $\neg x_i$.

Определение. Дизъюнктом или клозом называется дизъюнкция нескольких (возможно одного) литералов: $(x_1 \vee \neg x_2 \vee x_3)$.

Определение. Формулой в КНФ называется конъюнкция нескольких (возможно одного) дизъюнктов: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \neg x_1$.

Определение. Конъюнктом или мономом называется конъюнкция нескольких (возможно одного) литералов: $(x_1 \wedge \neg x_2 \wedge x_3)$.

Определение. Формулой в ДНФ называется дизъюнкция нескольких (возможно одного) конъюнктов: $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \neg x_1$.

КНФ и ДНФ

Определение. Литералом называется пропозициональная переменная или ее отрицание: x_i, $\neg x_i$.

Определение. Дизъюнктом или клозом называется дизъюнкция нескольких (возможно одного) литералов: $(x_1 \vee \neg x_2 \vee x_3)$.

Определение. Формулой в КНФ называется конъюнкция нескольких (возможно одного) дизъюнктов: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \neg x_1$.

Определение. Конъюнктом или мономом называется конъюнкция нескольких (возможно одного) литералов: $(x_1 \wedge \neg x_2 \wedge x_3)$.

Определение. Формулой в ДНФ называется дизъюнкция нескольких (возможно одного) конъюнктов: $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \neg x_1$.

КНФ и ДНФ

Определение. Литералом называется пропозициональная переменная или ее отрицание: x_i, $\neg x_i$.

Определение. Дизъюнктом или клозом называется дизъюнкция нескольких (возможно одного) литералов: $(x_1 \vee \neg x_2 \vee x_3)$.

Определение. Формулой в КНФ называется конъюнкция нескольких (возможно одного) дизъюнктов: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \neg x_1$.

Определение. Конъюнктом или мономом называется конъюнкция нескольких (возможно одного) литералов: $(x_1 \wedge \neg x_2 \wedge x_3)$.

Определение. Формулой в ДНФ называется дизъюнкция нескольких (возможно одного) конъюнктов: $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \neg x_1$.

КНФ и ДНФ

Определение. Литералом называется пропозициональная переменная или ее отрицание: x_i, $\neg x_i$.

Определение. Дизъюнктом или клозом называется дизъюнкция нескольких (возможно одного) литералов: $(x_1 \vee \neg x_2 \vee x_3)$.

Определение. Формулой в КНФ называется конъюнкция нескольких (возможно одного) дизъюнктов: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \neg x_1$.

Определение. Конъюнктом или мономом называется конъюнкция нескольких (возможно одного) литералов: $(x_1 \wedge \neg x_2 \wedge x_3)$.

Определение. Формулой в ДНФ называется дизъюнкция нескольких (возможно одного) конъюнктов: $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \neg x_1$.

Представление булевых функций формулами

Теорема. Любую булеву функцию можно записать в виде пропозициональной формулы в КНФ и ДНФ.

Иллюстрация.

$f(x_1, x_2, x_3)$			
x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Формула в ДНФ:

$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee$
$(\neg x_1 \wedge \neg x_2 \wedge x_3) \vee$
$(x_1 \wedge \neg x_2 \wedge \neg x_3)$

Формула в КНФ:

$(x_1 \vee \neg x_2 \vee x_3) \wedge$
$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

Представление булевых функций формулами

Теорема. Любую булеву функцию можно записать в виде пропозициональной формулы в КНФ и ДНФ.

Иллюстрация.

$f(x_1, x_2, x_3)$			
x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Формула в ДНФ:

$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee$
$(\neg x_1 \wedge \neg x_2 \wedge x_3) \vee$
$(x_1 \wedge \neg x_2 \wedge \neg x_3)$

Формула в КНФ:

$(x_1 \vee \neg x_2 \vee x_3) \wedge$
$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

Формулы де Моргана

- значение $\neg(A \vee B)$ совпадает со значением $(\neg A \wedge \neg B)$
- значение $\neg(A \wedge B)$ совпадает со значением $(\neg A \vee \neg B)$

Следствие. Отрицание формулы в ДНФ “есть” формула в КНФ

Доказательство.

$\neg((l_{1,1} \wedge l_{1,2} \wedge \cdots \wedge l_{1,n_1}) \vee (l_{2,1} \wedge l_{2,2} \wedge \cdots \wedge l_{2,n_2}) \vee \ldots$
$\vee(l_{k,1} \wedge l_{k,2} \wedge \cdots \wedge l_{k,n_k}))$

можно переписать в виде

$(\neg l_{1,1} \vee \neg l_{1,2} \vee \cdots \vee \neg l_{1,n_1}) \wedge (\neg l_{2,1} \vee \neg l_{2,2} \wedge \cdots \vee \neg l_{2,n_2}) \wedge \ldots$
$\wedge(\neg l_{k,1} \vee \neg l_{k,2} \vee \cdots \vee \neg l_{k,n_k})$

Формулы де Моргана

- значение $\neg(A \vee B)$ совпадает со значением $(\neg A \wedge \neg B)$
- значение $\neg(A \wedge B)$ совпадает со значением $(\neg A \vee \neg B)$

Следствие. Отрицание формулы в ДНФ “есть” формула в КНФ

Доказательство.

$\neg((l_{1,1} \wedge l_{1,2} \wedge \cdots \wedge l_{1,n_1}) \vee (l_{2,1} \wedge l_{2,2} \wedge \cdots \wedge l_{2,n_2}) \vee \ldots$
$\vee(l_{k,1} \wedge l_{k,2} \wedge \cdots \wedge l_{k,n_k}))$

можно переписать в виде

$(\neg l_{1,1} \vee \neg l_{1,2} \vee \cdots \vee \neg l_{1,n_1}) \wedge (\neg l_{2,1} \vee \neg l_{2,2} \wedge \cdots \vee \neg l_{2,n_2}) \wedge \ldots$
$\wedge(\neg l_{k,1} \vee \neg l_{k,2} \vee \cdots \vee \neg l_{k,n_k})$

Выполнимость, общезначимость, противоречивость

Определение. Пропозициональная формула называется выполнимой , если существует такая интерпретация, при которой значение формулы равняется 1.

Определение. Пропозициональная формула называется невыполнимой (или противоречивой) , если при всех интерпретациях значение формулы равняется 0.

Определение. Пропозициональная формула называется общезначимой (или тавтологией) , если при всех интерпретациях значение формулы равняется 1.

Определение. Пропозициональная формула называется необщезначимой , если при существует интерпретация, при которой значение формулы равняется 0.

Выполнимость, общезначимость, противоречивость

Определение. Пропозициональная формула называется выполнимой , если существует такая интерпретация, при которой значение формулы равняется 1.
Определение. Пропозициональная формула называется невыполнимой (или противоречивой) , если при всех интерпретациях значение формулы равняется 0.
Определение. Пропозициональная формула называется общезначимой (или тавтологией) , если при всех интерпретациях значение формулы равняется 1.
Определение. Пропозициональная формула называется необщезначимой , если при существует интерпретация, при которой значение формулы равняется 0.

Выполнимость, общезначимость, противоречивость

Определение. Пропозициональная формула называется выполнимой , если существует такая интерпретация, при которой значение формулы равняется 1.
Определение. Пропозициональная формула называется невыполнимой (или противоречивой) , если при всех интерпретациях значение формулы равняется 0.
Определение. Пропозициональная формула называется общезначимой (или тавтологией) , если при всех интерпретациях значение формулы равняется 1.
Определение. Пропозициональная формула называется необщезначимой , если при существует интерпретация, при которой значение формулы равняется 0.

О сложности...

- Задача определения по формуле в КНФ, выполнима ли она, является сложной (**NP**-трудной).
- Проверить формулу в КНФ, является ли она тавтологией, очень просто. Достаточно проверить, что в каждом дизъюнкте есть пара: переменная и ее отрицание.
- Задача определения по формуле в ДНФ, выполнима ли она, является простой: достаточно проверить, есть ли в ней конъюнкт, в который одновременно не входит переменная и ее отрицание.
- Проверить формулу в ДНФ, является ли она тавтологией, сложно (**NP**-трудно).

О сложности...

- Задача определения по формуле в КНФ, выполнима ли она, является сложной (**NP**-трудной).
- Проверить формулу в КНФ, является ли она тавтологией, очень просто. Достаточно проверить, что в каждом дизъюнкте есть пара: переменная и ее отрицание.
- Задача определения по формуле в ДНФ, выполнима ли она, является простой: достаточно проверить, есть ли в ней конъюнкт, в который одновременно не входит переменная и ее отрицание.
- Проверить формулу в ДНФ, является ли она тавтологией, сложно (**NP**-трудно).

О сложности...

- Задача определения по формуле в КНФ, выполнима ли она, является сложной (**NP**-трудной).
- Проверить формулу в КНФ, является ли она тавтологией, очень просто. Достаточно проверить, что в каждом дизъюнкте есть пара: переменная и ее отрицание.
- Задача определения по формуле в ДНФ, выполнима ли она, является простой: достаточно проверить, есть ли в ней конъюнкт, в который одновременно не входит переменная и ее отрицание.
- Проверить формулу в ДНФ, является ли она тавтологией, сложно (**NP**-трудно).

О сложности...

- Задача определения по формуле в КНФ, выполнима ли она, является сложной (**NP**-трудной).
- Проверить формулу в КНФ, является ли она тавтологией, очень просто. Достаточно проверить, что в каждом дизъюнкте есть пара: переменная и ее отрицание.
- Задача определения по формуле в ДНФ, выполнима ли она, является простой: достаточно проверить, есть ли в ней конъюнкт, в который одновременно не входит переменная и ее отрицание.
- Проверить формулу в ДНФ, является ли она тавтологией, сложно (**NP**-трудно).

Приведение формулы в ДНФ

- Избавляемся от импликации: $A \to B$ заменяем на $\neg A \vee B$.
- По правилам де Моргана проносим отрицания до переменных
- Пользуясь дистрибутивностью $A \wedge (C \vee D) = A \wedge C \vee A \wedge D$ раскрываем скобки.
- Упрощаем, выкидывая невыполнимые конъюнкты

Пример. $((A \vee C) \to B) \wedge (A \vee C)$

избавляемся от импликации: $(\neg(A \vee C) \vee B) \wedge (A \vee C)$

проносим отрицания: $((\neg A \wedge \neg C) \vee B) \wedge (A \vee C)$

раскрываем скобки:

$(\neg A \wedge \neg C \wedge A) \vee (\neg A \wedge \neg C \wedge C) \vee (B \wedge A) \vee (B \wedge C)$

упрощаем: $(B \wedge A) \vee (B \wedge C)$

Приведение формулы в ДНФ

- Избавляемся от импликации: $A \to B$ заменяем на $\neg A \vee B$.
- По правилам де Моргана проносим отрицания до переменных
- Пользуясь дистрибутивностью $A \wedge (C \vee D) = A \wedge C \vee A \wedge D$ раскрываем скобки.
- Упрощаем, выкидывая невыполнимые конъюнкты

Пример. $((A \vee C) \to B) \wedge (A \vee C)$
избавляемся от импликации: $(\neg(A \vee C) \vee B) \wedge (A \vee C)$
проносим отрицания: $((\neg A \wedge \neg C) \vee B) \wedge (A \vee C)$
раскрываем скобки:
$(\neg A \wedge \neg C \wedge A) \vee (\neg A \wedge \neg C \wedge C) \vee (B \wedge A) \vee (B \wedge C)$
упрощаем: $(B \wedge A) \vee (B \wedge C)$

Приведение формулы в ДНФ

- Избавляемся от импликации: $A \to B$ заменяем на $\neg A \vee B$.
- По правилам де Моргана проносим отрицания до переменных
- Пользуясь дистрибутивностью $A \wedge (C \vee D) = A \wedge C \vee A \wedge D$ раскрываем скобки.
- Упрощаем, выкидывая невыполнимые конъюнкты

Пример. $((A \vee C) \to B) \wedge (A \vee C)$
избавляемся от импликации: $(\neg(A \vee C) \vee B) \wedge (A \vee C)$
проносим отрицания: $((\neg A \wedge \neg C) \vee B) \wedge (A \vee C)$
раскрываем скобки:
$(\neg A \wedge \neg C \wedge A) \vee (\neg A \wedge \neg C \wedge C) \vee (B \wedge A) \vee (B \wedge C)$
упрощаем: $(B \wedge A) \vee (B \wedge C)$

Приведение формулы в ДНФ

- Избавляемся от импликации: $A \to B$ заменяем на $\neg A \vee B$.
- По правилам де Моргана проносим отрицания до переменных
- Пользуясь дистрибутивностью $A \wedge (C \vee D) = A \wedge C \vee A \wedge D$ раскрываем скобки.
- Упрощаем, выкидывая невыполнимые конъюнкты

Пример. $((A \vee C) \to B) \wedge (A \vee C)$
избавляемся от импликации: $(\neg(A \vee C) \vee B) \wedge (A \vee C)$
проносим отрицания: $((\neg A \wedge \neg C) \vee B) \wedge (A \vee C)$
раскрываем скобки:
$(\neg A \wedge \neg C \wedge A) \vee (\neg A \wedge \neg C \wedge C) \vee (B \wedge A) \vee (B \wedge C)$
упрощаем: $(B \wedge A) \vee (B \wedge C)$

Приведение формулы в ДНФ

- Избавляемся от импликации: $A \to B$ заменяем на $\neg A \vee B$.
- По правилам де Моргана проносим отрицания до переменных
- Пользуясь дистрибутивностью $A \wedge (C \vee D) = A \wedge C \vee A \wedge D$ раскрываем скобки.
- Упрощаем, выкидывая невыполнимые конъюнкты

Пример. $((A \vee C) \to B) \wedge (A \vee C)$
избавляемся от импликации: $(\neg(A \vee C) \vee B) \wedge (A \vee C)$
проносим отрицания: $((\neg A \wedge \neg C) \vee B) \wedge (A \vee C)$
раскрываем скобки:
$(\neg A \wedge \neg C \wedge A) \vee (\neg A \wedge \neg C \wedge C) \vee (B \wedge A) \vee (B \wedge C)$
упрощаем: $(B \wedge A) \vee (B \wedge C)$

Упражнения и задачи

1. Постройте машину Тьюринга, которая прибавляет 1 к написанному двоичному числу.
2. Постройте машину Тьюринга, которая проверит, является ли строка палиндромом.
3. Докажите, что нет алгоритма, который определял бы, закончит ли МТ работу на пустом входе.
4. Существует ли алгоритм, проверяющий, работает ли данная МТ полиномиальное время или нет?
5. Докажите, что не существует алгоритма, который определил бы по МТ M определил бы, является ли последовательность $M(1), M(2), M(3) \ldots$ периодической с некоторого места.