Лемма. Транзитивность сводимости. Если $M_1 \le_P M_2$ и $M_2 \le_P M_3$, то $M_1 \le_P M_3$.

Доказательство. Пусть $\varphi_1 : I_1 \to I_2$ и $\varphi_2 : I_2 \to I_3$ полиномиально вычислимые функции, задающие сводимость. Тогда функция $\varphi_2 \circ \varphi_1$ сводит задачу M_1 к задаче M_3. Осталось убедиться, что она полиномиально вычислима. Пусть трудоемкость вычисления функций φ_1 и φ_2 равна $O(l^k)$ и $O\ (l^m)$. Тогда для любой индивидуальной задачи $i \in I_1$ длина записи $\varphi_1(i)$ равна $O(|i|^k)$ и трудоёмкость вычисления суперпозиции $\varphi_2 \circ \varphi_1$ равна $O(|i|^k)+O((O(|i|^k))^m)=O(|i|^{km})$.

Теорема. Если $M_1 \in NPC$, $M_2 \in NP$ и $M_1 \le_P M_2$, то $M_2 \in NPC$.

Доказательство. По определению класса NPC для $\forall M \in NP\ M \le_P M_1$. По лемме о транзитивности сводимости $\forall M \in NP\ M \le_P M_2$.

Теорема позволяет расширять список *NP*-полных задач, сводя к новой задаче известную *NP*-полную задачу.

Первая *NP*-трудная задача.

Выполнимость схемы функциональных элементов.

Вход. n булевых переменных и схема функциональных переменных, содержащая m функциональных элементов с одним выходом, заданная с помощью ориентированного графа G. (Длина входа не превосходит $o((n+m)^2)$)

Вопрос. Существует ли набор значений истинности булевых переменных выполняющий схему функциональных элементов?

Утверждение. Задача выполнимость СФЭ принадлежит классу *NP*.

Доказательство. В качестве сертификата возьмём значения свободных переменных x_i и значения y_j полученные на выходе каждого функционального элемента. Тогда для любого функционального элемента легко проверить соответствуют ли водящие значения выходящим. Для выполнимой схемы достаточно взять значения x_i, выполняющие СФЭ и соответствующие им значения y_j. Для невыполнимой схемы любой сертификат отвергается.

Теорема. Задача выполнимость СФЭ *NP*-полна.

Для формального доказательства теоремы необходимо конкретизировать модель алгоритма, используемого для представления вычислений.
Мы ограничимся интуитивным представлением алгоритма, основанного на принципах работы аппаратного обеспечения компьютера.

Компьютерная программа хранится в памяти в виде последовательности инструкций. Типичная инструкция содержит код выполняемой операции, адреса операндов и адрес, куда помещается результат. Специальные ячейки памяти *счётчик команд* (program counter, PC) следит за номером инструкции, которая должна выполняться следующей. При извлечении очередной инструкции счётчик увеличивается на единицу. В результате выполнения некоторых инструкций счётчик может принимать заданное значение, что позволяет организовать циклы и условные операторы.

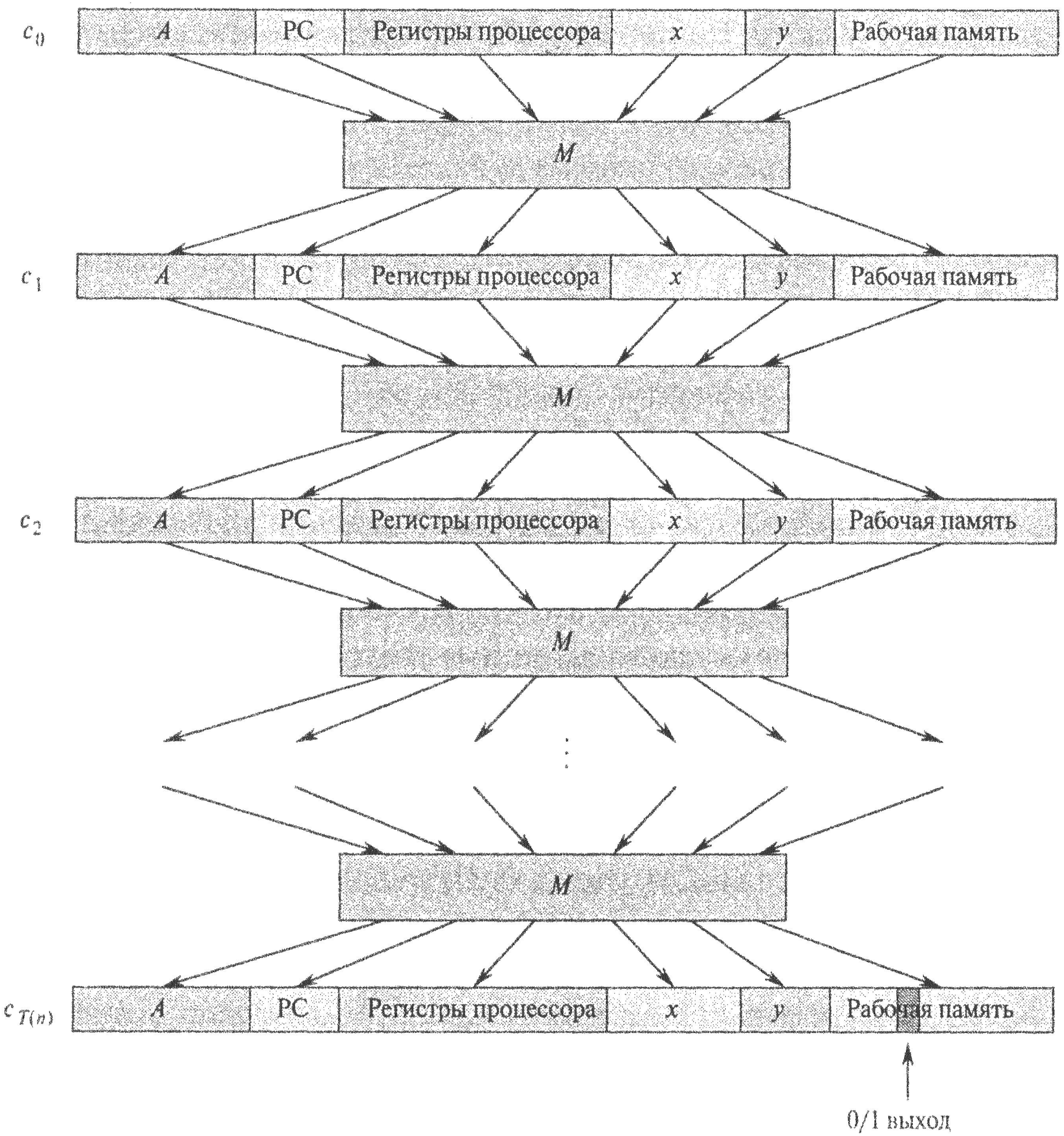
В любой момент времени выполнения программы состояние вычислений можно представить бинарной строкой содержания памяти компьютера, которое мы назовём *конфигурацией*. Выполнение команды можно рассматривать как отображение одной конфигурации на другую. Важно, что переход от одной конфигурации к другой организован как схема функциональных элементов, размер которой ограничен полиномом от длины конфигурации.

Доказательство теоремы об *NP*-полноте.

Пусть М произвольная задача из класса *NP*. Тогда для неё существует полиномиально вычислимый алгоритм верификации $A(x,y)$. Используем его для построения индивидуальной задачи выполнимости СФЭ для каждой индивидуальной задачи x из M.

В качестве множества переменных возьмём бинарную строку соответствующую бинарной записи сертификата y.

Представим выполнение алгоритма $A(x,y)$ как последовательность конфигураций. Пусть длина входа x равна n, а трудоёмкость алгоритма $T(n)=O(n^k)$. Тогда количество последовательных конфигураций не превосходит $T(n)$. И работу алгоритма можно изобразить следующей схемой.


c_0
A
PC
Регистры процессора
x
y
Рабочая память
M
c_1
c_2
$c_{T(n)}$
0/1 выход

Алгоритм приведения F склеивает все $T(n)$ копий СФЭ М в одну склеивая выходы конфигурации i со входами конфигурации i+1. После этого, реализаций с помощью функциональной схемы формул $q \wedge \overline{q}$ и $q \vee \overline{q}$, зафиксируем известные константы соответствующие коду алгоритма A, счетчику команд, регистрам процессора, индивидуальной задачи x и рабочим ячейкам памяти. И, наконец, отбросим все выходы полученной СФЭ, кроме ячейки, соответствующей ответу. Сконструированная таким образом схема вычисляет значение $C(y)=A(x,y)$.
Таким образом, индивидуальная задача верифицируется тогда и только тогда, когда построенная таким образом СФЭ C выполнима. Осталось только убедиться, что алгоритм приведения F полиномиален.

1. длина конфигурации полиномиально зависит от $n=|x|$.

A	PC	Регистры процессора	x	y	Рабочая память

Поскольку время работы алгоритма $T(n)$ то длина сертификата и рабочей памяти не превосходит $T(n)$. Длина кода алгоритма, регистра команд и регистра процессора константы не зависящие от n следовательно длина конфигурации $O(n^k)$.

2. Пусть сложность схемы M есть $O(l^m)$, где l длина конфигурации и m некоторая константа. Тогда сложность схемы C

$|C|=T(n)\cdot O(l^m)=O(n^k)O(n^{km})=O(n^s)$.

Основные *NP*-полные задачи.

Выполнимость булевой формулы.

Вход: n булевых переменных и булева формула, состоящая из m логических операций.

Вопрос. Существует ли набор значений истинности переменных, выполняющий булеву формулу?

3-выполнимость.

Существует ли набор значений истинности переменных, выполняющий 3-*CNF*?

Трехмерное сочетание (3-С).

Вход: Множества W, X и Y такие, что $|W|=|X|=|Y|=n$, и множество троек элементов $M \subset W \times X \times Y$, $|M|=m$.

Вопрос. Существует ли подмножество $M' \subset M$ такое, что $|M'|=n$ и никакие два разных элемента подмножества не имеют ни одной равной координаты?

Разбиение.

Вход. Множество A и весовая функция ω: $A \rightarrow N$.

Вопрос. Существует ли подмножество $A' \subset A$ такое, что

$$\sum_{a \in A'} \omega(a) = \sum_{a \in A \setminus A'} \omega(a)?$$

Вершинное покрытие (ВП).

Вход. Граф $G=(V, E)$ и натуральное число K.

Вопрос. Существует ли вершинное покрытие графа G мощности K?

Клика.

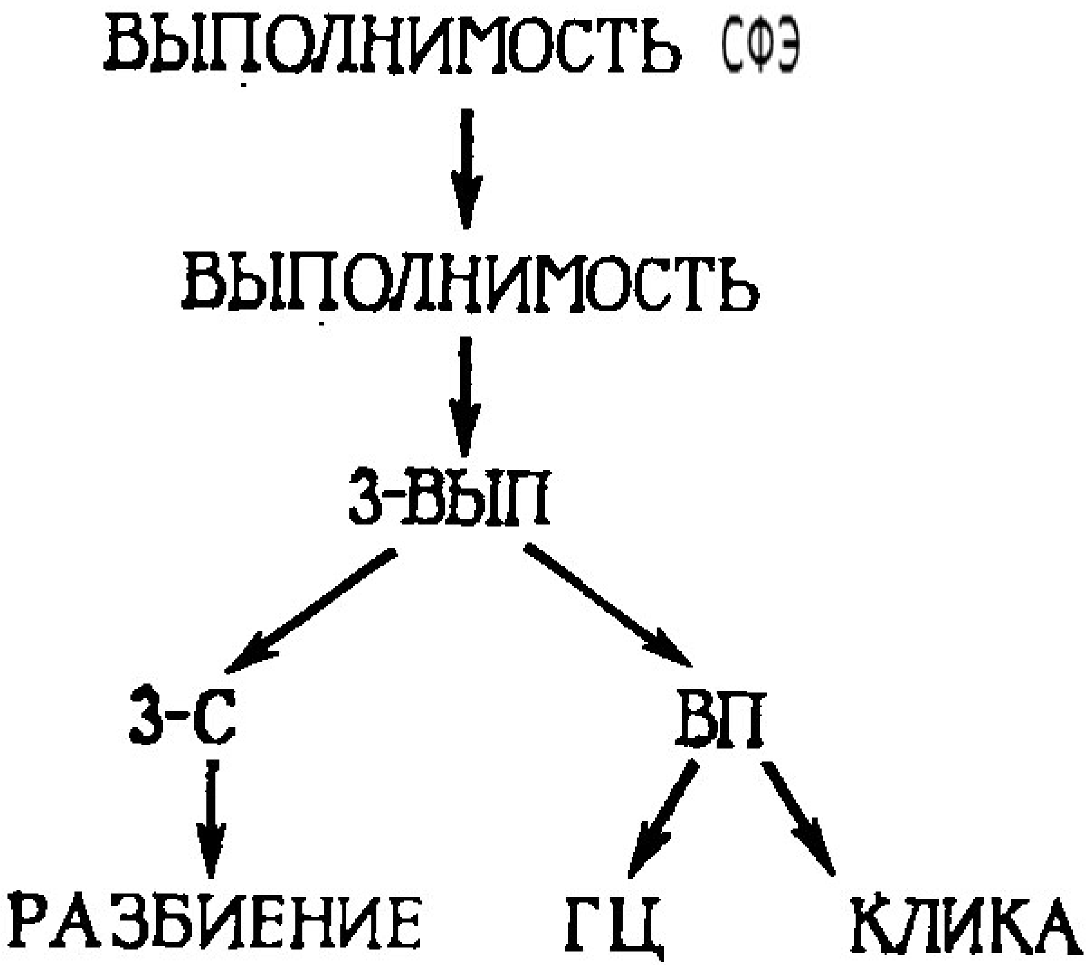
Вход. Граф $G=(V, E)$ и натуральное число K.
Вопрос. Существует ли клика в графе G мощности K?

Независимое множество.

Вход. Граф $G=(V, E)$ и натуральное число K.
Вопрос. Существует ли клика в графе G мощности K?

Гамильтонов цикл (ГЦ).

Вход. Граф $G=(V, E)$.
Вопрос. Является ли граф G гамильтоновым?


ВЫПОЛНИМОСТЬ СФЭ
ВЫПОЛНИМОСТЬ
3-ВЫП
3-С
ВП
РАЗБИЕНИЕ
ГЦ
КЛИКА

Теорема. Задача выполнимость *NP*-полна.

Доказательство.

I. Выполнимость принадлежит *NP*. Очевидно.

II. Выполнимость СФЭ $\leq_P$ Выполнимость.

Построим по индивидуальной задаче выполнимость СФЭ индивидуальную задачу выполнимость следующим образом. Сопоставим каждому выходу функционального элемента каждому входу СФЭ вспомогательную переменную y_j, $j=\{1,\ldots,m+n\}$. Отрицанию соответствует формула $y_i \Leftrightarrow \bar{y}_j$, где y_i и y_j вход и выход функционального элемента. Конъюнкции и дизъюнкции соответствуют формулы $(y_k \wedge y_l) \Leftrightarrow y_j$ и $(y_k \vee y_l) \Leftrightarrow y_j$. Для построения окончательной формулы возьмём конъюнкцию формул всех функциональных элементов. Построенная формула выполнима тогда и только тогда, когда выполнима исходная СФЭ. И наконец количество элементарных формул совпадает с количеством функциональных элементов, а длина каждой ограничена константой, следовательно, функция перехода полиномиально вычислима.

NP-полнота задачи выполнимость.

Определение. *Синтаксическое дерево* булевой функции — это бинарное дерево, листьями которого являются все вхождения литералов в формулу, а узлы соответствуют логическим операциям, причем уровень вершины соответствует приоритету выполняемой операции. Соответственно корню соответствует последняя выполняемая операция.

Упражнение. Построить бинарное дерево соответствующее формуле $\left(\left(x_1 \to x_2\right) \vee \neg\left(\left(\neg x_1 \leftrightarrow x_3\right) \vee x_4\right)\right) \wedge \neg x_2$.

Теорема. Задача 3-выполнимость принадлежит классу NPC.

Доказательство. Зададим функцию приведения F, сопоставляющую каждой индивидуальной задаче выполнимость некоторую задачу 3-выполнимость.

Вначале построим синтаксическое дерево и сопоставим каждому узлу вспомогательную переменную y_j. Также как в предыдущей теореме сопоставим каждому узлу формулу проверяющую соответствие входа и выхода и возьмем конъюнкцию формул всех узлов. При этом формула разборки выполнима тогда и только тогда, когда выполнима исходная формула.

Пример. Функции $\phi = ((x_1 \to x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$ соответствует формула разборки

$$\phi' = y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \wedge (y_3 \leftrightarrow (x_1 \to x_2)) \wedge$$
$$\wedge (y_4 \leftrightarrow \neg y_5) \wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3)).$$

Заметим, что формула записана в виде конъюнкции выражений содержащих не более 3 литералов.
Заменим формулы φ_i, содержащие два литерала, на выражение $(\phi_i \vee p) \wedge (\phi_i \vee \neg p)$, а формулу, содержащую единственный литерал, на выражение $(\phi_i \vee p \vee q) \wedge (\phi_i \vee \neg p \vee q) \wedge (\phi_i \vee p \vee \neg q) \wedge (\phi_i \vee \neg p \vee \neg q)$. После чего каждое выражение содержит ровно 3 литерала.
Пользуясь таблицей истинности можно записать каждое из них как конъюнкцию не более чем 7 дизъюнкций содержащих ровно по три литерала.
Для завершения доказательства осталось заметить, что построенная формула содержит менее чем $28m$ дизъюнкций, где m количество логических операторов в исходной формуле.

Замечание. Сводимость можно реализовать, построив не более чем $7m$ дизьюнкций.