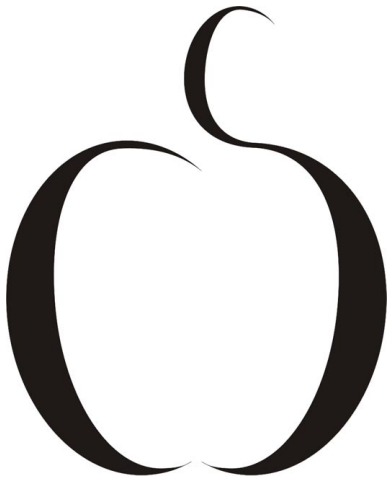
Алгоритмы для NP-трудных задач

Лекция 8: Алгоритмы расщепления

А. Куликов

Computer Science клуб при ПОМИ
http://logic.pdmi.ras.ru/~infclub/



План лекции

1. Комбинированные меры сложности

План лекции

1. Комбинированные меры сложности
2. Запоминание дизъюнктов

План лекции

Задача максимальной 2-выполнимости

Задача максимальной 2-выполнимости

- Задача максимальной 2-выполнимости (maximum 2-satisfiability) заключается в нахождении набора, выполняющего максимальное количество клозов данной формулы в 2-КНФ (то есть каждый клоз которой содержит не более двух литералов).

Задача максимальной 2-выполнимости

- Задача максимальной 2-выполнимости (maximum 2-satisfiability) заключается в нахождении набора, выполняющего максимальное количество клозов данной формулы в 2-КНФ (то есть каждый клоз которой содержит не более двух литералов).
- Является NP-трудной (сравните с 2-выполнимостью).

Задача максимальной 2-выполнимости

- Задача максимальной 2-выполнимости (maximum 2-satisfiability) заключается в нахождении набора, выполняющего максимальное количество клозов данной формулы в 2-КНФ (то есть каждый клоз которой содержит не более двух литералов).
- Является NP-трудной (сравните с 2-выполнимостью).
- В терминах максимальной 2-выполнимости легко формулируются многие задачи на графах.

Меры сложности

Стандартные меры сложности формул

- n — количество переменных
- K — количество клозов
- L — количество дизъюнктов

Стандартные меры сложности графов

- n — количество вершин
- m — количество ребер

Меры сложности

Стандартные меры сложности формул

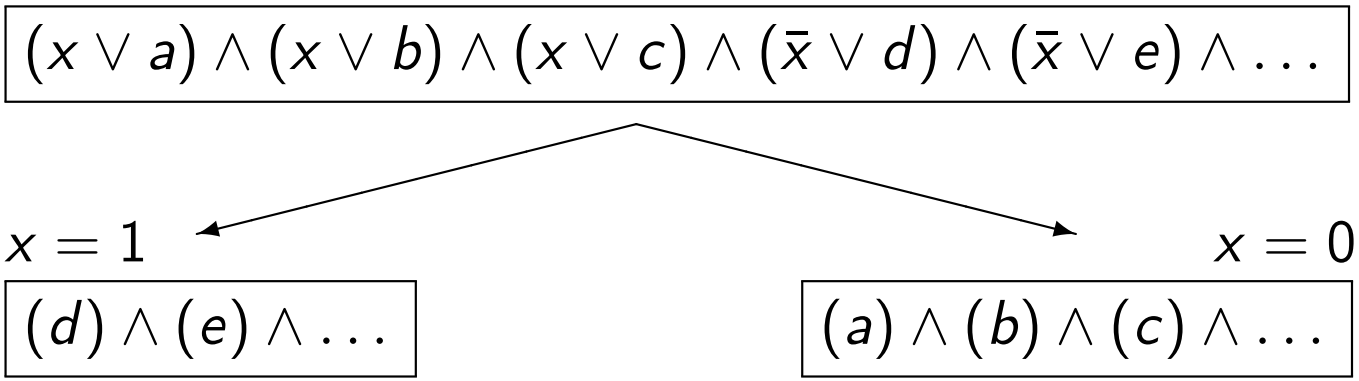
- n — количество переменных
- K — количество клозов
- L — количество дизъюнктов

Стандартные меры сложности графов

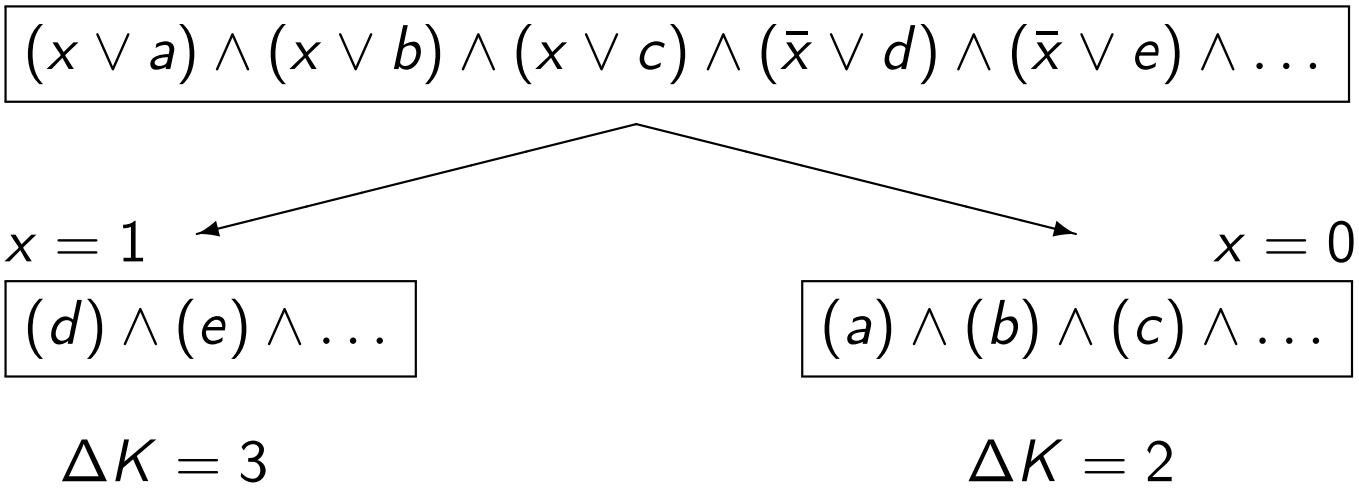
- n — количество вершин
- m — количество ребер

Как правило, именно относительно этих мер хочется получить верхнюю оценку на время работы алгоритма. Однако для доказательства оценки можно использовать и другую меру.

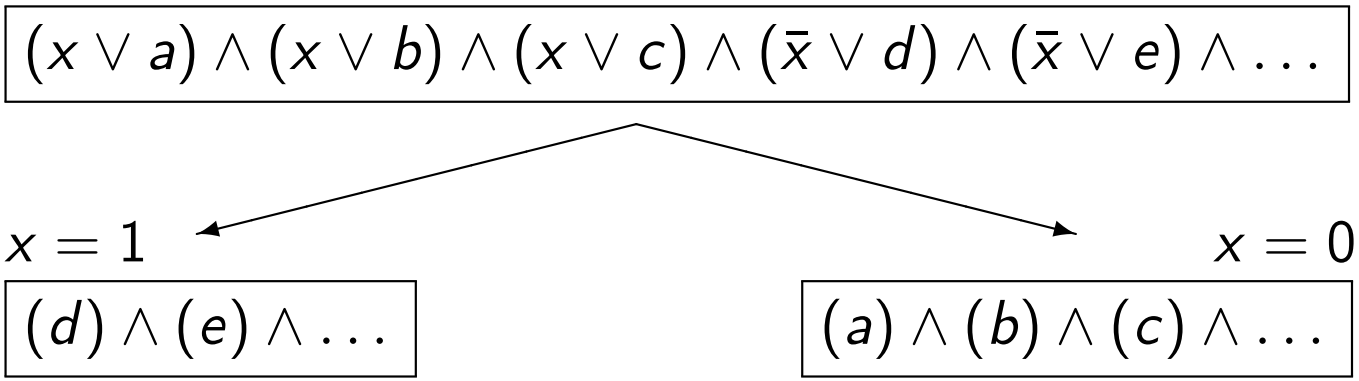
Пример: K и K_2


$(x \vee a) \wedge (x \vee b) \wedge (x \vee c) \wedge (\bar{x} \vee d) \wedge (\bar{x} \vee e) \wedge \ldots$
$x = 1$
$x = 0$
$(d) \wedge (e) \wedge \ldots$
$(a) \wedge (b) \wedge (c) \wedge \ldots$

Пример: K и K_2


$(x \vee a) \wedge (x \vee b) \wedge (x \vee c) \wedge (\bar{x} \vee d) \wedge (\bar{x} \vee e) \wedge \ldots$
$x = 1$
$x = 0$
$(d) \wedge (e) \wedge \ldots$
$(a) \wedge (b) \wedge (c) \wedge \ldots$
$\Delta K = 3$
$\Delta K = 2$

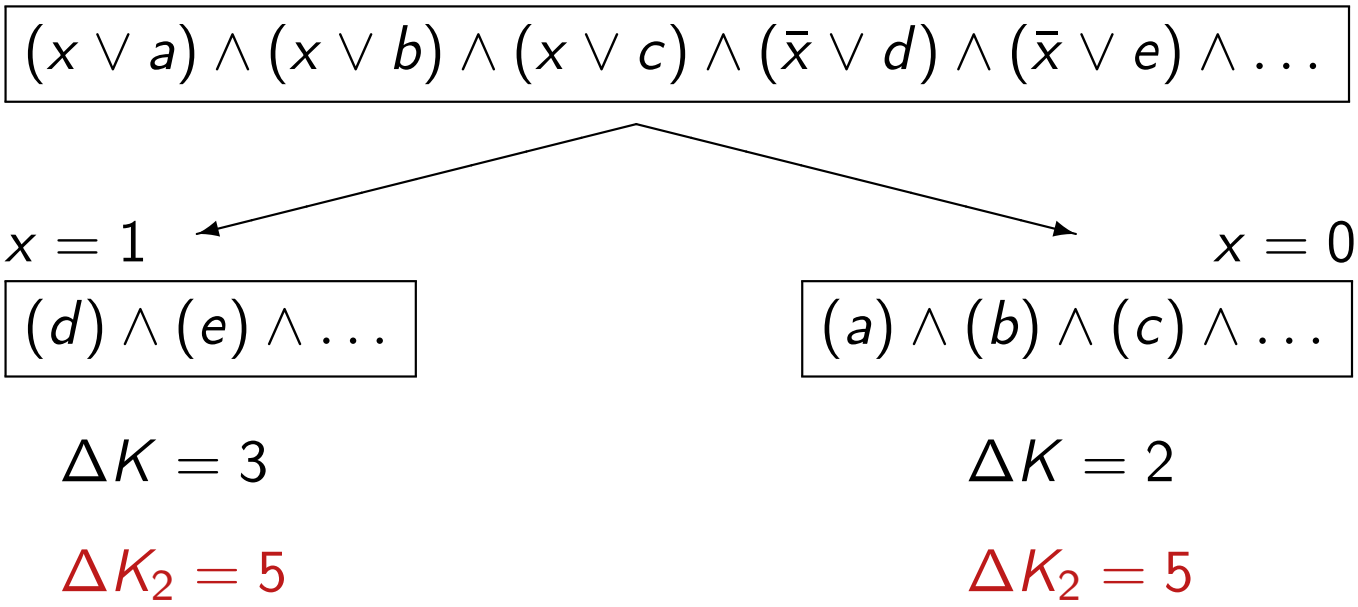
Пример: K и K_2



$\Delta K = 3$ $\quad$ $\Delta K = 2$

$\Delta K_2 = 5$ $\quad$ $\Delta K_2 = 5$

Пример: K и K_2



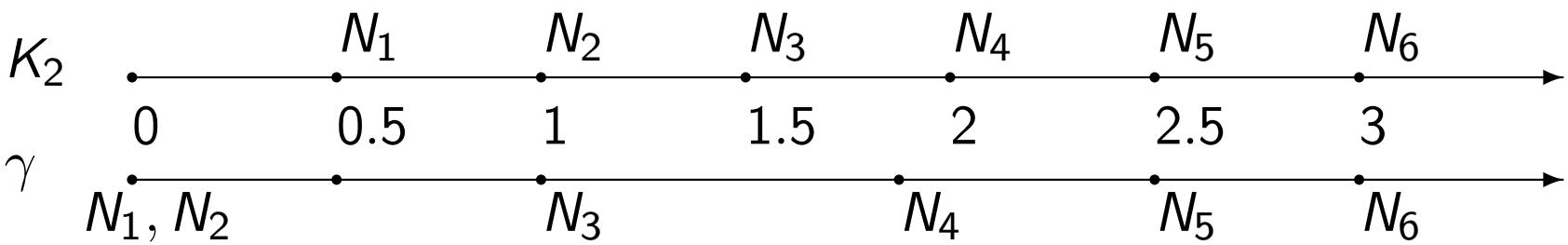
- $K_2 \leq K$
- K_2 — честная мера сложности: если $K_2(F) = 0$, от решение для F может быть найдено за линейное время

Комбинированная мера сложности

Обозначим через N_i количество переменных, входящих в ровно i 2-клоза.

количество 2-клозов: $$K_2 = \sum_{i=1}^{N} \frac{i}{2} N_i$$

комбинированная мера: $$\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i$$



Верхняя оценка $2^{K/5.5}$

Теорема

MAX-2-SAT может быть решена за время $\mathrm{poly}(K) \cdot 2^{K/5.5}$, где K — количество клозов формулы F.

Верхняя оценка $2^{K/5.5}$

Теорема

MAX-2-SAT может быть решена за время $\mathrm{poly}(K) \cdot 2^{K/5.5}$, где K — количество клозов формулы F.

Доказательство

Верхняя оценка $2^{K/5.5}$

Теорема

MAX-2-SAT может быть решена за время $\mathrm{poly}(K) \cdot 2^{K/5.5}$, где K — количество клозов формулы F.

Доказательство

- докажем оценку $2^{\gamma/5.5}$; этого достаточно, поскольку $\gamma \leq K_2 \leq K$

Верхняя оценка $2^{K/5.5}$

Теорема

MAX-2-SAT может быть решена за время $\mathrm{poly}(K) \cdot 2^{K/5.5}$, где K — количество клозов формулы F.

Доказательство

- докажем оценку $2^{\gamma/5.5}$; этого достаточно, поскольку $\gamma \leq K_2 \leq K$
- для этого покажем, что присваивание значения переменной формулы уменьшает γ хотя бы на 5.5

Доказательство

- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза

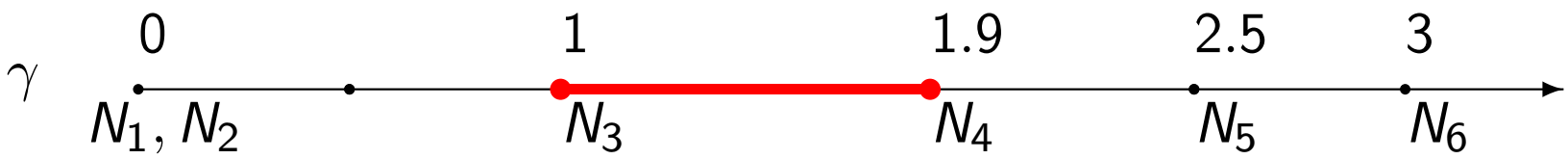
Доказательство

- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-клозов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$

Доказательство

- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-клозов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$
- рассмотрим переменную входящую в ровно пять 2-клозов: $\Delta\gamma \geq 2.5 + 5 \cdot 0.6 = 5.5$

Доказательство



- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-клозов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$
- рассмотрим переменную входящую в ровно пять 2-клозов: $\Delta\gamma \geq 2.5 + 5 \cdot 0.6 = 5.5$
- рассмотрим переменную входящую в ровно четыре 2-клозов: $\Delta\gamma \geq 1.9 + 4 \cdot 0.9 = 5.5$

$(n, 3)$-MAX-2-SAT

$(n, 3)$-MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза

$(n, 3)$-MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза
- если F упрощена, то все ее переменные входят ровно в три 2-клоза; значит, количество переменных четно

$(n, 3)$-MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза
- если F упрощена, то все ее переменные входят ровно в три 2-клоза; значит, количество переменных четно
- $\gamma(F) = N(F)$

$(n, 3)$-MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза
- если F упрощена, то все ее переменные входят ровно в три 2-клоза; значит, количество переменных четно
- $\gamma(F) = N(F)$
- необходимо показать, что после присваивания значения переменной еще хотя бы четыре другие переменные будут удалены правилами упрощения

$(n, 3)$-MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза
- если F упрощена, то все ее переменные входят ровно в три 2-клоза; значит, количество переменных четно
- $\gamma(F) = N(F)$
- необходимо показать, что после присваивания значения переменной еще хотя бы четыре другие переменные будут удалены правилами упрощения
- доказывается небольшим перебором случаев (есть и автоматическое доказательство)

План лекции

Запоминание дизъюнктов для задачи выполнимости

Запоминание дизъюнктов для задачи выполнимости

В алгоритмах, решающих задачу выполнимости, запоминание дизъюнктов означает, что алгоритм хранит частичные наборы переменных, которые не могут быть расширены до выполняющего.

Запоминание дизъюнктов для задачи выполнимости

Запоминание дизъюнктов для задачи выполнимости

В алгоритмах, решающих задачу выполнимости, запоминание дизъюнктов означает, что алгоритм хранит частичные наборы переменных, которые не могут быть расширены до выполняющего.

Пример

$\{y = 0\}$ является таким набором для формулы

$$(x \vee y \vee z) \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge (y \vee \neg z).$$

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

- Выбрать произвольный клоз $(x_1 \vee x_2 \vee \cdots \vee x_t)$ данной формулы F в k-КНФ и произвести рекурсивные вызовы для
  $F[x_1 = 1]$,
  $F[x_1 = 0, x_2 = 1]$,
  $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
  $\ldots$
  $F[x_1 = 0, x_2 = 0, \ldots, x_{t-1} = 0, x_t = 1]$.

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

- Выбрать произвольный клоз $(x_1 \vee x_2 \vee \cdots \vee x_t)$ данной формулы F в k-КНФ и произвести рекурсивные вызовы для
  $F[x_1 = 1]$,
  $F[x_1 = 0, x_2 = 1]$,
  $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
  $\ldots$
  $F[x_1 = 0, x_2 = 0, \ldots, x_{t-1} = 0, x_t = 1]$.

Анализ

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

- Выбрать произвольный клоз $(x_1 \vee x_2 \vee \cdots \vee x_t)$ данной формулы F в k-КНФ и произвести рекурсивные вызовы для
  $F[x_1 = 1]$,
  $F[x_1 = 0, x_2 = 1]$,
  $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
  $\ldots$
  $F[x_1 = 0, x_2 = 0, \ldots, x_{t-1} = 0, x_t = 1]$.

Анализ

- Расщепление выше является $(1, 2, \ldots, t)$-расщеплением относительно количества переменных n.

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

- Выбрать произвольный клоз $(x_1 \vee x_2 \vee \cdots \vee x_t)$ данной формулы F в k-КНФ и произвести рекурсивные вызовы для
  $F[x_1 = 1]$,
  $F[x_1 = 0, x_2 = 1]$,
  $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
  $\ldots$
  $F[x_1 = 0, x_2 = 0, \ldots, x_{t-1} = 0, x_t = 1]$.

Анализ

- Расщепление выше является $(1, 2, \ldots, t)$-расщеплением относительно количества переменных n.
- Поскольку $t \leq k$, $\tau(1, 2, \ldots, t) \leq \tau(1, 2, \ldots, k) = c_k < 2$.

Как решить k-SAT за менее чем 2^n шагов?

Алгоритм

- Выбрать произвольный клоз $(x_1 \vee x_2 \vee \cdots \vee x_t)$ данной формулы F в k-КНФ и произвести рекурсивные вызовы для
  $F[x_1 = 1]$,
  $F[x_1 = 0, x_2 = 1]$,
  $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
  $\ldots$
  $F[x_1 = 0, x_2 = 0, \ldots, x_{t-1} = 0, x_t = 1]$.

Анализ

- Расщепление выше является $(1, 2, \ldots, t)$-расщеплением относительно количества переменных n.
- Поскольку $t \leq k$, $\tau(1, 2, \ldots, t) \leq \tau(1, 2, \ldots, k) = c_k < 2$.
- Таким образом, время работы не превосходит c_k^n.

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0},$

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \cdots \wedge D_q}_{F_1}$

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \cdots \wedge D_q}_{F_1}$
- Допустим, что F_0 сравнительно мала, так что её выполнимость можно проверить за полиномиальное время.

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \cdots \wedge D_q}_{F_1}$
- Допустим, что F_0 сравнительно мала, так что её выполнимость можно проверить за полиномиальное время.
- Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и не нужно расщепляться вообще (можно просто присвоить значение 1 переменной a).

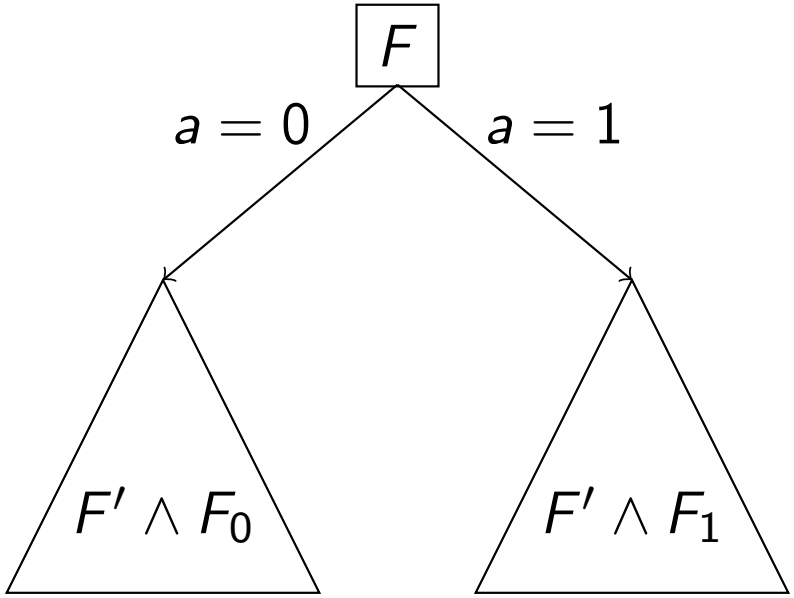
Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \cdots \wedge D_q}_{F_1}$
- Допустим, что F_0 сравнительно мала, так что её выполнимость можно проверить за полиномиальное время.
- Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и не нужно расщепляться вообще (можно просто присвоить значение 1 переменной a).
- В противном случае найдём набор α, выполняющий F_0.

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

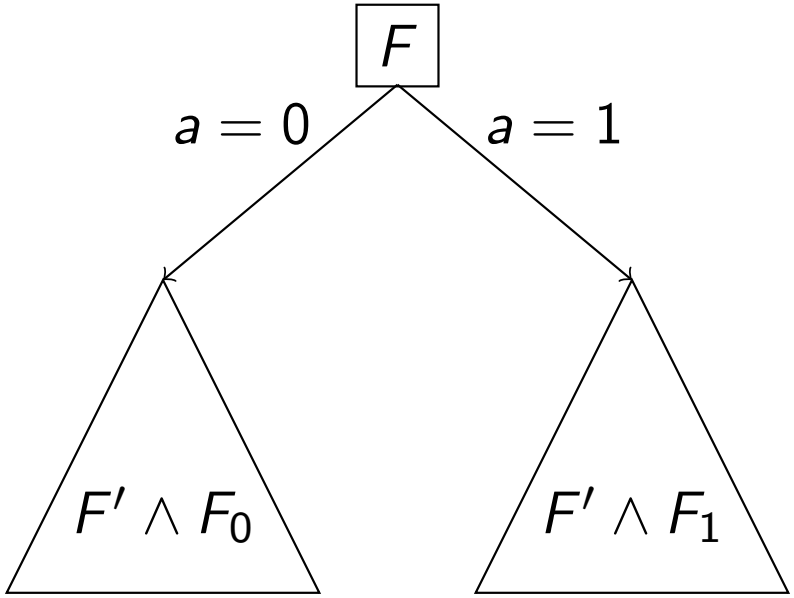
- $F = F' \wedge (a \vee C_1) \wedge \cdots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \ldots (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \cdots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \cdots \wedge D_q}_{F_1}$
- Допустим, что F_0 сравнительно мала, так что её выполнимость можно проверить за полиномиальное время.
- Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и не нужно расщепляться вообще (можно просто присвоить значение 1 переменной a).
- В противном случае найдём набор α, выполняющий F_0.
- Тогда мы можем рекурсивно вызваться для $F[a = 0]$ и если получен ответ “невыполнима”, тогда мы знаем, что α не может быть расширен до выполняющего набора F'. Таким образом, нет необходимости рассматривать расширения α в ветке $F[a = 1]$.

Основные идеи: иллюстрация



Основные идеи: иллюстрация

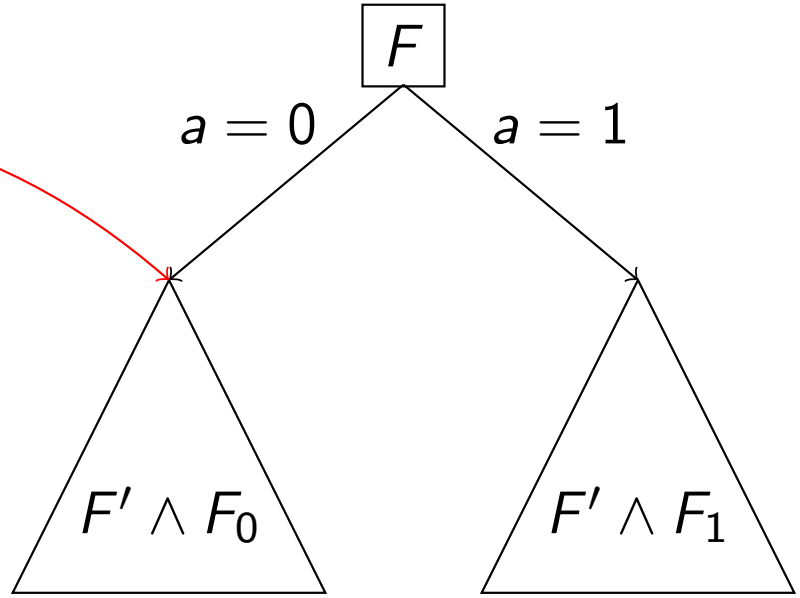
проверяем выполнимость F_0



Основные идеи: иллюстрация

проверяем выполнимость F_0

если F_0 невыполнима, тогда не нужно идти в левую ветку

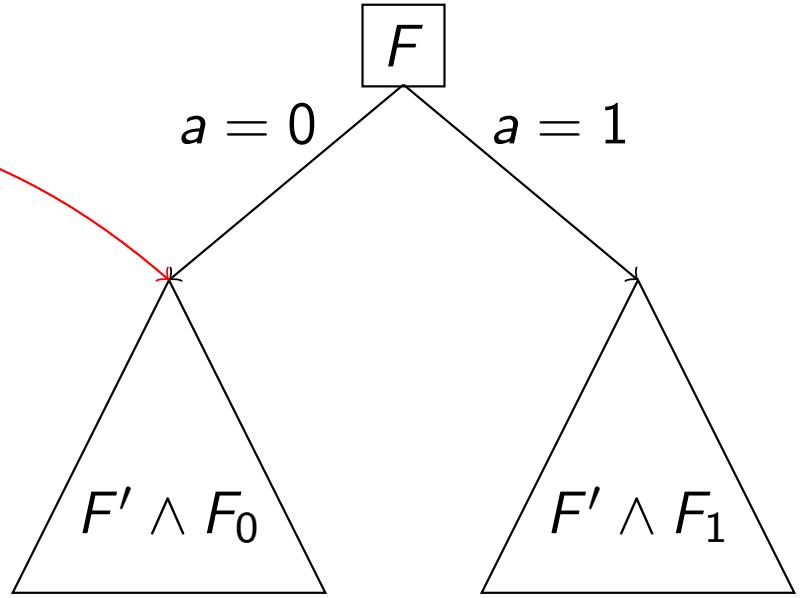


Основные идеи: иллюстрация

проверяем выполнимость F_0

если F_0 невыполнима, тогда не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0



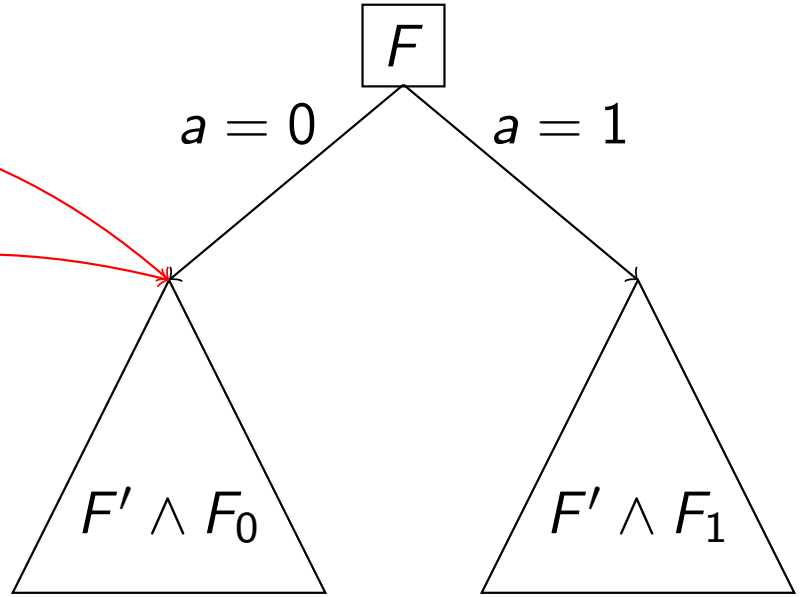
Основные идеи: иллюстрация

проверяем выполнимость F_0

если F_0 невыполнима, тогда не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

тогда идём в левую ветку



Основные идеи: иллюстрация

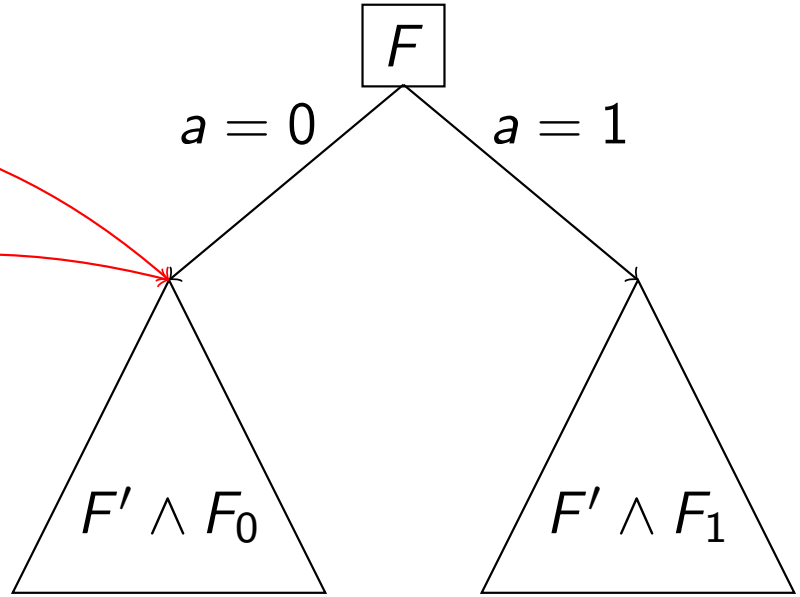
проверяем выполнимость F_0

если F_0 невыполнима, тогда
не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

тогда идём в левую ветку

если $F' \wedge F_0$ выполнима,
просто сообщаем, что выполнима и F



Основные идеи: иллюстрация

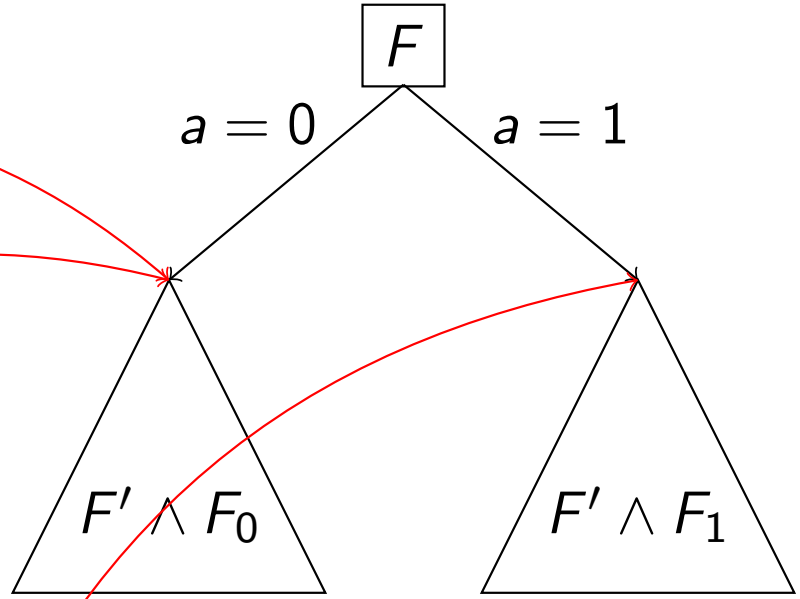
проверяем выполнимость F_0

если F_0 невыполнима, тогда
не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

тогда идём в левую ветку

если $F' \wedge F_0$ выполнима,
просто сообщаем, что выполнима и F



в противном случае идём в правую ветку

Основные идеи: иллюстрация

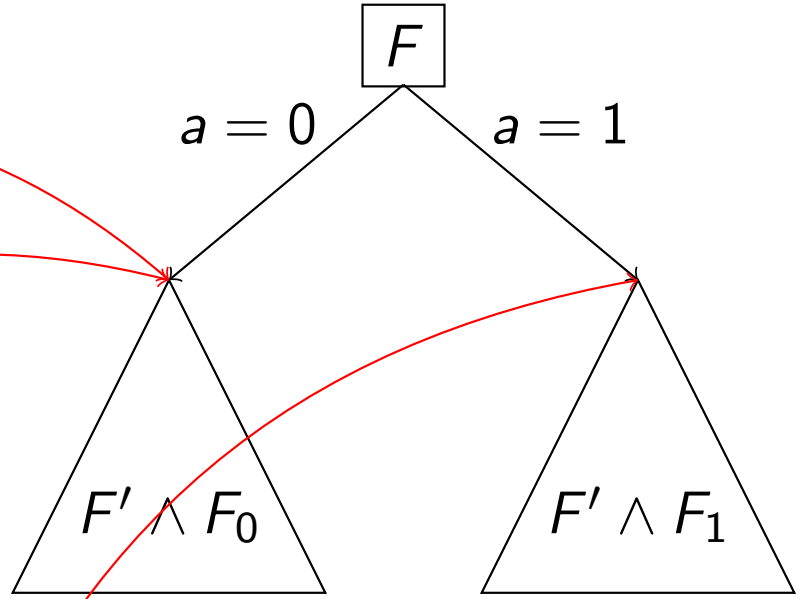
проверяем выполнимость F_0

если F_0 невыполнима, тогда
не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

тогда идём в левую ветку

если $F' \wedge F_0$ выполнима,
просто сообщаем, что выполнима и F



в противном случае идём в правую ветку

однако мы теперь знаем, что расширения α не могут выполнить F'

Основные идеи: иллюстрация

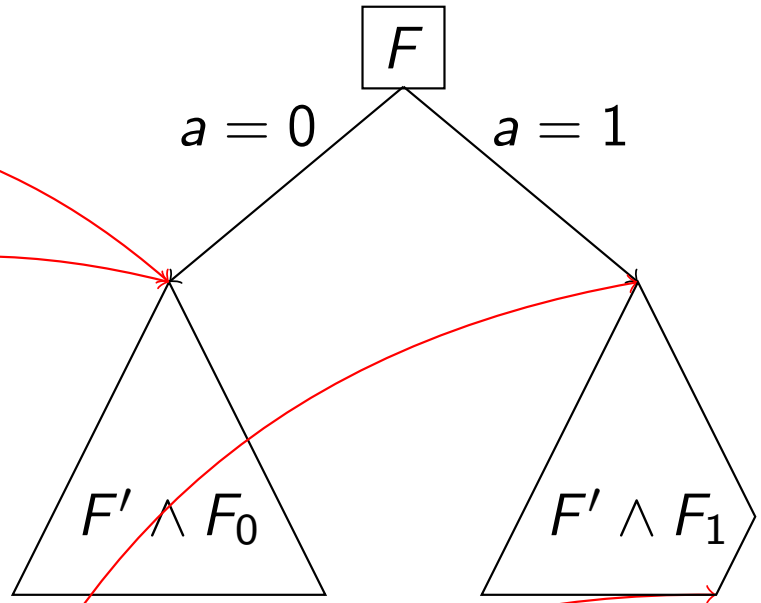
проверяем выполнимость F_0

если F_0 невыполнима, тогда
не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

тогда идём в левую ветку

если $F' \wedge F_0$ выполнима,
просто сообщаем, что выполнима и F



в противном случае идём в правую ветку

однако мы теперь знаем, что расширения α не могут выполнить F'
и пространство поиска сужается

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.
- В противном случае выберем d^--литерал a.

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.
- В противном случае выберем d^--литерал a.
  - Пусть $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$.

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.
- В противном случае выберем d^--литерал a.
  - Пусть $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$.
  - Поскольку a является d^--литералом, F_0 содержит не более d клозов и мы можем проверить выполнимость F_0 за полиномиальное время.

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.
- В противном случае выберем d^--литерал a.
  - Пусть $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$.
  - Поскольку a является d^--литералом, F_0 содержит не более d клозов и мы можем проверить выполнимость F_0 за полиномиальное время.
  - Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и расщепляться не нужно (можно просто присвоить значение 1 переменной a).

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn клозами.

- Если F содержит литерал a, такой что и a, и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a = 0]$ и $F[a = 1]$.
- В противном случае выберем d^--литерал a.
  - Пусть $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$.
  - Поскольку a является d^--литералом, F_0 содержит не более d клозов и мы можем проверить выполнимость F_0 за полиномиальное время.
  - Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и расщепляться не нужно (можно просто присвоить значение 1 переменной a).
  - В противном случае находим набор $\alpha = \{l_1, \ldots, l_t\}$, выполняющий F_0.

Продолжение описания алгоритма

Продолжение описания алгоритма

- Напомним, что a — d^--литерал, $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$, $\alpha = \{l_1, \ldots, l_t\}$ — выполняющий набор F_0.

Продолжение описания алгоритма

- Напомним, что a — d^--литерал, $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$, $\alpha = \{l_1, \ldots, l_t\}$ — выполняющий набор F_0.
- Рекурсивно вызовемся для $F[a = 0]$.

Продолжение описания алгоритма

- Напомним, что a — d^--литерал, $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$, $\alpha = \{l_1, \ldots, l_t\}$ — выполняющий набор F_0.
- Рекурсивно вызовемся для $F[a = 0]$.
- Если $F[a = 0]$ выполнима, то выполнима и F.

Продолжение описания алгоритма

- Напомним, что a — d^--литерал, $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$, $\alpha = \{l_1, \ldots, l_t\}$ — выполняющий набор F_0.
- Рекурсивно вызовемся для $F[a = 0]$.
- Если $F[a = 0]$ выполнима, то выполнима и F.
- В противном случае $F[a = 0]$ невыполнима и мы заключаем, что α не может быть расширен до выполняющего набора F' (ибо $F[a = 0] = F' \wedge F_0$).

Продолжение описания алгоритма

- Напомним, что a — d^--литерал, $F[a = 0] = F' \wedge F_0$, $F[a = 1] = F' \wedge F_1$, $\alpha = \{l_1, \ldots, l_t\}$ — выполняющий набор F_0.
- Рекурсивно вызовемся для $F[a = 0]$.
- Если $F[a = 0]$ выполнима, то выполнима и F.
- В противном случае $F[a = 0]$ невыполнима и мы заключаем, что α не может быть расширен до выполняющего набора F' (ибо $F[a = 0] = F' \wedge F_0$). Значит, в ветке $F[a = 1]$ можно расщепиться так: $F[a = 1, l_1 = 0], F[a = 1, l_1 = 1, l_2 = 0], \ldots,$ $F[a = 1, l_1 = 1, l_2 = 1, \ldots, l_{t-1} = 1, l_t = 0]$.

Анализ времени работы алгоритма

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности

$$\gamma(F) = n(F) + w \cdot m(F)$$

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^--литералу даёт число расщепления не более $\tau(1, 2, \ldots, d + 1)$

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^--литералу даёт число расщепления не более $\tau(1, 2, \ldots, d + 1)$
- положим d равным произвольной константе, большей Δ

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$
- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^--литералу даёт число расщепления не более $\tau(1, 2, \ldots, d + 1)$
- положим d равным произвольной константе, большей Δ
- пусть w — константа, такая что $2^{\frac{1}{1+wd}} = \tau(1, 2, \ldots, d + 1)$

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^--литералу даёт число расщепления не более $\tau(1, 2, \ldots, d + 1)$
- положим d равным произвольной константе, большей Δ
- пусть w — константа, такая что $2^{\frac{1}{1+wd}} = \tau(1, 2, \ldots, d + 1)$
- тогда время работы алгоритмы не превосходит $= 2^{\frac{n+wm}{1+wd}}$

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности
$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^+)-литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^--литералу даёт число расщепления не более $\tau(1, 2, \ldots, d + 1)$
- положим d равным произвольной константе, большей Δ
- пусть w — константа, такая что $2^{\frac{1}{1+wd}} = \tau(1, 2, \ldots, d + 1)$
- тогда время работы алгоритмы не превосходит $= 2^{\frac{n+wm}{1+wd}} < 2^{\frac{n+w\Delta n}{1+wd}} = c^n$,
где $c = 2^{\frac{1+w\Delta}{1+wd}} < 2$ — константа

Заключение

Заключение

- Итак, мы построили алгоритм для задачи выполнимости формулы константной плотности со временем работы c^n, где $c < 2$ — константа, а n — количество переменных.

Заключение

- Итак, мы построили алгоритм для задачи выполнимости формулы константной плотности со временем работы c^n, где $c < 2$ — константа, а n — количество переменных.
- Напомним, что в общем случае задачи выполнимости (без ограничений на длину клоза и количество клозов) таких алгоритмов неизвестно.

Спасибо за внимание!