# 1 Лекция 1

$X = \{x_1, x_2, \ldots, x_n, \ldots\}$ - **алфавит**. Символы $x_i \in X$ называем **буквами** (над алфавитом $X$).
Любая конечная последовательность бкув $x_1, x_2, \ldots, x_n$ называется **словом** (над алфавитом $X$).
Число символов из $X$ в слове называются **длиной слова**. Само слово может обозначается символом $\tilde{x}$, а длина слова обозначается символом $l$.

**Пример 1.1.** *$\tilde{x} = x_1, \ldots, x_n, l(\tilde{x}) = l(x_1, \ldots, x_n) = n$, или $\tilde{x}_k = x_{i_1}, \ldots, x_{i_s}, l(\tilde{x}_k) = l(x_{i_1}, \ldots, x_{i_s}) = s$.*

Множество всех слов над $X$ обозначается через $X^*$. $X^*$ удобно представлять в виде $X^* = \bigcup\limits_{i=0}^{\infty} X^i$, где $X^i = \{$множество слов над $X$ длины $i = 0, 1, 2, \ldots\}$.

**Пример 1.2.** *$X = \{0, 1, 2, \ldots, 9\}$,$X^1 = \{0, 1, 2, \ldots, 9\}$,*
*$X^2 = \{01, 02, \ldots, 09, 10, \ldots, 19, 20, \ldots, 29, \ldots\}$,. . . ,$X^* = \bigcup\limits_{i=0}^{\infty} X^i$*

**Вопрос 1.1.** *А что есть $X^0$?*

$\Lambda$ - пустое слово. $\forall \tilde{x} \in X^* : \Lambda\tilde{x} = \tilde{x}\Lambda = \tilde{x}$, т.е. $X^0 = \Lambda$.
Пусть $A_1, \ldots, A_n$ - некоторые множества. **Декартовым произведением** этих множеств называется $\{(a_{i_1}, \ldots, a_{i_n})\}$ - множество наборов, где $a_{i_1} \in A_1, \ldots, a_{i_n} \in A_n$. Если $A_1 = \ldots = A_n$, то говорят о **декартовой степени** $A^n = \underbrace{A \times \ldots \times A}_{n}$.

**Пример 1.3.** *$A_1 = \{a, b, c\}$, $A_2 = \{a, d\}$,$A_1 \times A_2 = \{(a, a), (a, d), (b, a), (b, d), (c, a), (c, d)\}$.*
*$A_1 \times A_2 \neq A_2 \times A_1$ в общем случае.*

$|A|$ - это обозначение числа элементов в $A$ (Если $A$ - конечное множество) или мощность $A$, если в $A$ бесконечное число элементов.
Пусть $A = A_1 \times A_n$. **Отношением** $\rho = \rho(x_1, \ldots, x_n)$ над $A = A_1 \times A_n$ называется произвольное подмножество $A_\rho \subseteq A = A_1 \times A_n$. Число $n$ называется **арностью**.

**Пример 1.4.** *$\rho = \rho(x_1) \subseteq A$ - унарное, $\rho = \rho(x_1, x_2) \subseteq A$ - бинарное, $\rho = \rho(x_1, x_2, x_3) \subseteq A$ - тернарное. - Отношения.*

Если $\rho = \rho(x_1, \ldots, x_n) \subseteq A_1 \times \ldots \times A_n$, то $x_1$ принимает значение из $A_1, \ldots, x_n$ принимает значение из $A_n$. Наборы из $A_1 \times \ldots \times A_n$ называется **кортежами** (длины $n$). Отношение $A_\rho = A_1 \times \ldots \times A_n \times A_{n+1}$ называется **функциональным** (по $x_1, \ldots, x_n$), если из совпадения любых кортежей по первым $n$ элементами $a'_{i_1} = a''_{i_1} = a_{i_1}, \ldots, a'_{i_n} = a''_{i_n} = a_{i_n}$ следует, что в $A_\rho$ есть либо один кортеж вида $a_{i_1}, \ldots, a_{i_n}, a$, либо ни одного такого кортежа. В этом случае $x_{n+1}$ переобозначают через $y$ и говорят **о фукциональной зависимости** $y = f(x_1, \ldots, x_n)$.

**Пример 1.5.** *В таблице 1 не функционально по $x_1 \in A_1, x_2 \in A_2$, т.к. есть два кортежа $x_1 = a, x_2 = b, c \neq d$. В таблице 2 функционально по $x_1 \in A_1, x_2 \in A_2$, но не функционально по $x_1 \in A_1$ и $x_3 \in A_3$ т.к. есть два кортежа $(a, b, a), (a, a, a)$.*

| $\rho$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| | $a$ | $b$ | $c$ |
| | $a$ | $a$ | $c$ |
| | $a$ | $b$ | $d$ |

| $\rho$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| | $a$ | $b$ | $a$ |
| | $a$ | $a$ | $a$ |
| | $b$ | $b$ | $a$ |

Рис. 1: Таблицы к примеру 1.5

**Вопрос 1.2.** *Какое понятие более общее: отношение или функция ?*

**Замечание 1.1.** *Понятие отношения и функции, или функциональной зависимости играет большую роль в теории баз данных.*

Бинарные отношения допускают геометрическую интерпретацию!
$A = \{a, b, c\}$, пусть $\rho \subseteq A \times A$. Если на пересечении строки $i$ и столбца $j$ стоит 1, то пара $(a_i, a_j) \in \rho$, если 0, то $(a_i, a_j) \notin \rho$.
Возьмем на плоскости три кружка и обозначим их символами из $A$(рис. 3).

| $\rho$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $a$ | 0 | 1 | 1 |
| $b$ | 1 | 0 | 0 |
| $c$ | 0 | 1 | 1 |

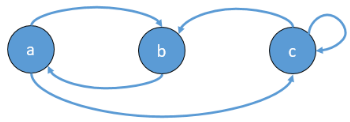



Рис. 2: Таблица и граф бинарного отношения

**Вопрос 1.3.** *По какому правилу проведены стрелки?*

Бинарное отношение называется:

1. **рефлексивным**, если $(a_i, a_i) \in A_\rho, \forall a_i \in A$.
2. **симметричным**, если из $(a_i, a_i) \in A_\rho \Rightarrow (a_j, a_i) \in A_\rho$.
3. **транзитивным**, если из $(a_i, a_j) \in A_\rho$ и $(a_j, a_k) \in A_\rho \Rightarrow (a_i, a_k) \in A_\rho$.

**Вопрос 1.4.** *Пусть $A = \{a_1, a_2, \ldots, a_n\}$. Как узнать, является ли бинарное отношение $A_\rho \in A \times A$: а) рефлексивным, б) симметричным, в) транзитивным?*

В случае симметричного отношения вместо двух стрелок (ориентированных дуг) рисуют просто ребро (неориентированную дугу).

**Определение 1.1.** ***Графом*** *$G$ называется пара множеств $G = (V, E)$, где $V = \{v_1, \ldots, v_n, \ldots\}$ - вершины графа и $E \subseteq V \times V = \{(v_{i_1}, v_{j_1}), (v_{i_2}, v_{j_2}), \ldots\}$ -ребра графа (пары вершин).*

Если $|V| < +\infty$, то граф $G$ называется **конечным** (в противном случае - **бесконечным**). Если все ребра графа ориентированные, то и граф называется **ориентированным**(в противном случае граф называется **неориентированным**). Неориентированному графу $G = (V, E)$ всегда соответствует симметричное бинарное отношение $E \subseteq V \times V$.

**Вопрос 1.5.** *По графам бинарных отношений $\rho_1$ и $\rho_2$ построить их таблицы (матрицы)*

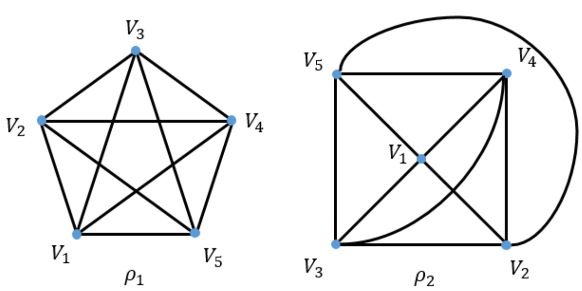



Рис. 3: К вопрос 1.5: Дуги пересекаются только в вершинах!

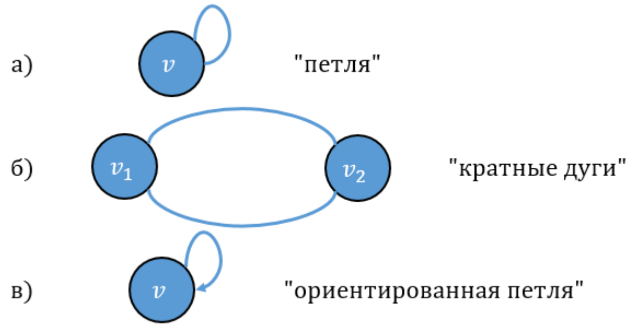



Рис. 4: К вопросу 1.7

**Вопрос 1.6.** *Сформулировать определение изоморфизма двух графов* $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$.

**Вопрос 1.7.** *Какому отношению соответствует неориентированный граф?*

Граф как отношение - нет кратных неориентированных ребер и неориентированных петель.
Граф как геометрический объект - может иметь кратные ребра, петли (в случае кратных ребер такой граф называют **мультиграфом**.) Мультиграф можно интерпретировать как отношение на *расширенном* множестве $G = (V, E)$, где $E \subseteq (V \times V) \times \mathbb{N}$. То есть, каждому ребру присваивается число (номер ребра)

**Пример 1.6.** $G_1 = (\{v_1, v_2\} = V, \{(v_1, v_2, 1), (v_1, v_2, 2), (v_2, v_1, 1), (v_2, v_1, 2)\} = E)$.

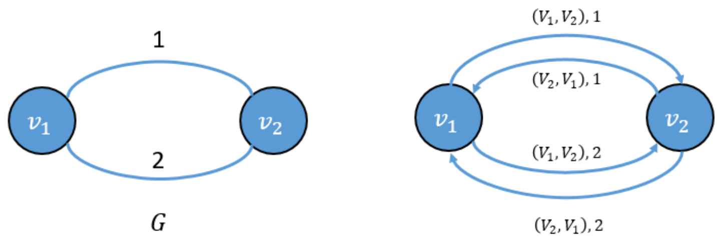



Рис. 5: Граф отношения

## 2 Лекция 2

Формальные системы $\Phi = \langle X, F, A, R \rangle$.

- $X$ - **алфавит** системы(список переменных)
- $F \subseteq X^*$ - выражения системы(**формулы** системы, атомарные(первичные) формулы системы). Формулы из $F$ задаются или списком или правилами их построения.
- $A \subseteq F$ - **аксиомы** системы(как правило соответствуют очевидным фактам предметной области)
- $R$ - **правила** вывода(правила получения новых формул).

**Пример 2.1.** *(Системы продукционного типа)* $\Phi C_1$*:*

- ***Алфавит****:* $X = \{x_1, \ldots, x_n\}; |x| = n$.
- ***Формулы****:* $F = X^*$ *- множество всех слов.*
- ***Аксиомы****: единственное выделенное слово* $\tilde{\alpha}_0 \in X^*$.

- ***Правила вывода:*** $\{R_i\} = \{\langle \tilde\beta_1, \tilde\beta_2, \tilde\beta_3; \tilde\beta_1', \tilde\beta_2', \tilde\beta_3' \rangle\}$.
  *Если* $\alpha \in X^*$ *имеет вид* $\tilde\alpha = \tilde\beta_1\, \tilde\delta_1\, \tilde\beta_2\, \tilde\delta_2\, \tilde\beta_3 \to \tilde\beta_1'\, \tilde\delta_1'\, \tilde\beta_2'\, \tilde\delta_2'\, \tilde\beta_3'$*- новое слово (формула).*
  *"$\to$"-"можно преобразовать".*

Здесь $\tilde\beta_i, \tilde\beta_i', \tilde\delta_i, \tilde\delta_i'$,- некоторое слова из $X^*, i = 1, \dots, 3$. Некоторые из этих слов могут быть $\Lambda$.

$$\tilde\alpha = \tilde\beta_1\, \tilde\delta_1\, \beta_2\, \tilde\delta_2\, \tilde\beta_3 \to \tilde\beta_1'\, \tilde\delta_1'\, \tilde\beta_2'\, \tilde\delta_2'\, \tilde\beta_3'$$

(**Подстановка**: слово $\tilde\beta_2$ заменяется на $\tilde\beta_2'$).
$\Lambda\, \tilde\delta_1\, \tilde\beta_2\, \tilde\delta_2\, \Lambda \to \tilde\delta_1\, \tilde\beta_2'\, \tilde\delta_2$ (**контекстная замена**)
$\Lambda\, \tilde\beta_2\, \tilde\delta_2\, \Lambda \to \Lambda\, \tilde\beta_2\, \tilde\delta_2'\, \Lambda$(**приписывание слова** $\beta_2'$ (которое может быть равно $\beta_2$) в конец после слова $\delta_2$. и т.д. Правил такого типа может быть много).
Выводимость Слово $\tilde\beta$ **выводимо** из слова $\tilde\alpha$ в $\Phi C_1$, если существует такая цепочка

$$\tilde\alpha \xrightarrow[R_{i_1}]{} \tilde\gamma_1 \xrightarrow[R_{i_2}]{} \tilde\gamma_2 \to \dots \xrightarrow[R_{i_n}]{} \tilde\gamma_n = \tilde\beta$$

где $R_{i_1}, R_{i_2}, \dots, R_{i_n}$ правила вывода(**продукции**) системы $\Phi C_1$. Число $n$ называется **длиной вывода** (слова $\tilde\alpha$ из $\tilde\beta$).

**Пример 2.2.** *"Эволюция генотипа"*
$X = \{u, c, G, A\}$*– алфавит.*
$\alpha_0$ *аксиома:* $GA$ *(генотип)*
*правила вывода(эволюция геноьтипа):*

$$R1.\quad \forall \widetilde{p} \in X^*, \widetilde{p} \to \widetilde{p}\widetilde{p} \text{(удвоение или полиплодия)}$$

$$\left.\begin{array}{ll} R2. & \widetilde{p}AGA\widetilde{q} \to \widetilde{p}AA\widetilde{q} \\ R3. & \widetilde{p}GAA\widetilde{q} \to \widetilde{p}AA\widetilde{q} \end{array}\right\} \text{выпадение буквы (или } A \text{ делеция } \widetilde{p} \text{ и } \widetilde{q} \in X^*\text{)(могут быть } \Lambda!\text{)}$$

***Вывод (эволюция) из*** $\widetilde{a}_0$
$\alpha_0 = GA \xrightarrow[R_1]{} G\underbrace{AGA} \xrightarrow[R_2]{} \underbrace{GAA} \xrightarrow[R_3]{} AA \xrightarrow[?]{} \dots$
$\alpha_0 = GA \xrightarrow[R_1]{} GAGA \xrightarrow[R_1]{} G\underbrace{AGA}GAGA \xrightarrow[R_2]{} GA\underbrace{AGA}GA \xrightarrow[R_2]{} GAA\underbrace{AGA} \xrightarrow[R_2]{} \underbrace{GA}AAA \xrightarrow[R_3]{} AAAA \xrightarrow[R_2]{}$
$\alpha_0 = GA \xrightarrow[R_1]{} GA\underbrace{GA} \xrightarrow[R_1]{} GAGA\underbrace{GA} \xrightarrow[R_1]{} GAGAGAGA \xrightarrow[R_1]{} \dots$

Можно показать что любое слово (генотип) вида $(GA)^{n_1}(GA)^{n_2}...(GA)^{n_k}$
$(k \geq 1, n_k \in \mathbb{N})$ может появиться в процессе эволюции.
$(GA)^{n_i} = \underbrace{GAGA...GA}_{n_j \text{ раз}}; A^i = \underbrace{A...A}_{i \text{ раз}}$.
Докажите это!
Никакое слово в котором есть комбинация $GG$ не может появиться.
Можно усложнить эту модель:
$R4.$ Если в слове $\widetilde{p} \in X^*$ два раза появится комбинация $AA$ то этот генотип (потомок слова $\widetilde{a}_0$)
гибнет: $\underbrace{\widetilde{p}_1 AA \widetilde{p}_2 AA}_{\widetilde{p}} \to \Lambda$.
Можно ввсти вместо "гибнет"вероятность выживания:
Пусть появление $AA$ означает, что с вероятностью, например, $1/4$ он погибнет: $\widetilde{a}_0 \to \dots \underbrace{AA}_{1/4} \to \dots \underbrace{AA}_{1/4} \dots \underbrace{AA}_{1/4} \to$.

**Вопрос 2.1.** *Какая будет доля выживших потомков?(напрмер, длины n)*

Главная проблема: По слову $\widetilde{p} \in X^*$ и набору правил $\{R_1, ..., R_n\}$ понять, выводимо ли слово $\widetilde{p}$ из $\widetilde{a}_0$ с помощью правим (продукций) $R = \{R_1, ..., R_n\}$. Иными словами $\widetilde{p} \in \left[X\right]_R^{\widetilde{a}_0}$!
Доказано,что в общем случае эта задача алогритмическим неразрешима!

**Пример 2.3.** ***Задача на разрезание.***
*Дано: Рисунки.*
*Надо: Получить (разрезать) так, чтобы:* $n_1$ *кусков* $0.8$, $n_2$ *кусков* $0.6$, $n_3$ *кусков* $0.3$, *и остакок был наименьшим*

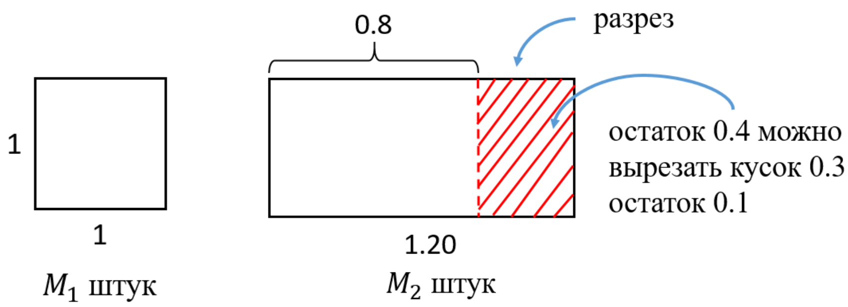



Рис. 6: К примеру 2.3

Формализация задачи:
$$\langle \underbrace{m_1, m_2}_{\text{количество кусков размера 1 и 1.20}} ; \underbrace{n_1, n_2, n_3}_{\text{количество кусков размера 0.8,0.6 и 0.3}} \rangle$$

**Аксиома** $\tilde{\alpha}_0 - \langle M_1, M_2; 0, 0, 0\rangle$
**Правила вывода**(продукции) $R$:
$R_1 : \langle x+1, y; z, u, v\rangle \to \langle x, y; z+1, u, v\rangle$
$R_2 : \langle x+1, y; z, u, v\rangle \to \langle x, y; z, u+1, v\rangle$
$R_3 : \langle x+1, y; z, u, v\rangle \to \langle x, y; z, u+1, v+1\rangle$
$R_4 : \langle x+1, y; z, u, v\rangle \to \langle x, y; z, u, v+3\rangle$
$R_5 : \langle x, y+1; z, u, v\rangle \to \langle x, y; z+1, u, v+1\rangle$
$R_6 : \langle x, y+1; z, u, v\rangle \to \langle x, y; z, u+2, v\rangle$
$R_7 : \langle x, y+1; z, u, v\rangle \to \langle x, y; z, u+1, v+2\rangle$
$R_8 : \langle x, y+1; z, u, v\rangle \to \langle x, y; z, u, v+4\rangle$
$R_9 : \langle x, y+1; z, u, v\rangle \to \langle x, y; z+1, u, v+1\rangle$

**Вопрос 2.2.** *1.Одну продукцию можно удалить какую? -* $R_2$
*2.Что является в этой системе алфавитом* $X$ *и что есть* $\widetilde{x} \in X^*$*?*

**Пример 2.4.** *(преобразования слов)//* $X = \{А,Б,В,Г,...,Э,Ю,Я\}$.
$X^* = \{$*любые конечные последовательности букв из* $X\}$.
***Аксиома***: $\alpha_0 =$*МАТР.*
***Правила****(продукции)*
$R_1 : \forall \widetilde{x}$ *и* $\widetilde{\beta}$ *из* $X^* : \widetilde{x}\widetilde{\beta} \to \widetilde{\beta}\widetilde{x}$
$R_2 :$ *МАТ* $\to$ *ТО*
$R_3 :$ *РАС* $\to$ *ТО*
$R_4 :$ *Р* $\to$ *РАС*
$R_5 :$ *РАС* $\to$ *РОС*
$R_6 :$ *ТО* $\to$ *АВ*
$R_7 : \widetilde{x}Т\widetilde{y} \to \widetilde{x}ТТ\widetilde{y}$*(контексная замена)*
$МАТР \xrightarrow[R_4]{} МАТРАС \xrightarrow[R_3]{} МАТТО \xrightarrow[R_1]{} ТОМАТ$
$МАТР \xrightarrow[R_1]{} РМАТ \xrightarrow[R_4]{} РАСМАТ \xrightarrow[R_5]{} РОСМАТ \xrightarrow[R_5]{} МАТРОС$

**Вопрос 2.3.** *Можно ли из МАТР получить АВТОМАТ используя* $R_1 - R_8$ *и как?*

**Пример 2.5.** $X = \{М,А,П\}$ $R_1 : М \to П$ $R_2 : П \to М$
*аксиома - слово ПАПА:* $ПАПА \xrightarrow[R_2]{} МАПА \xrightarrow[R_2]{} МАМА \xrightarrow[R_1]{} ПАМА \xrightarrow[R_1]{} ПАПА \xrightarrow[...]{} ...$
*аксиома* $АА \xrightarrow[?]{}$
*аксиома АП:* $АП \to АМ \to АП \to АМ \to АП \to ...$

**Пример 2.6.** $X = \{М,А,П\}$
*Правила R:* $М \to П$, $А \to П$, $М \to А$, $П \to М$, $А \to М$, $П \to А$.

**Вопрос 2.4.** *что содержит* $[X]_R^{\widetilde{\alpha}_0}$*?*

$\left[X\right]_R^{\widetilde{a}_0}$ - множество всех слов, которые получаются из $\alpha_0 \in X^*$ применением правил вывода $R$. Это множество называется замыканием $X$ относительно слова $\widetilde{a}_0$ и правила $R$.

**Пример 2.7.** $X = \{M,A,\Pi\}$
*К правилам $R$ из предыдущего примера добавалено правило $R_1 : A \to AA$*

**Вопрос 2.5.** *Отличается ли $\left[X\right]_{R\cup R_1}^{\widetilde{a}_0}$ от $\left[X\right]_R^{\widetilde{a}_0}$ ? Если да, то чем?*

# 3 Лекция 3

Исчисление высказываний (calculus of propositions)
**высказывание** - любое предложения языка(русского, английского,греческого...), про которое можно сказать **истинно(True)** оно или **ложно(False)**.

**Пример 3.1.** *1. "Все студенты - отличники".*
*2. "В десятичной системе $2 \times 2 = 4$".*
*3. "В невырожденном треугольнике сумма двух сторон больше третьей стороны".*
*4. "Луна состоит из зелёного сыра".*
*5. "Вода не может прератиться в лёд".*
*6. "Идёт дождь".*
*7. "Ручка упала на пол".*
*Предложения 1,2,3,4,5 - высказывания; 1,2,3 - истинные, 4,5 - ложные.*
*Предложения 6,7 - не высказывания - они ни истинны и не ложны.*

Будем обозначать предложения высказывания символами $\{P_1, P_2, ..., P_n, ...\}$ - это пропозициональные переменные. Поскольку смысл(sense) высказываний нас не нитересует(важно истинно высказывание или ложно) то можно считать, что $P_1, ..., P_n, ...$ - **булевские переменные** и принимают значение "0"(если высказывание ложно) и "1"(если высказывание истинно).
Из высказываний с помощью связок "и", "или", "не", "если A то B", "$\wedge$", "$\vee$", "$\neg$,-", "$A \supset B$" и т.д. можно строить более **сложные высказывания**.
Истинно сложное высказывание или нет зависит от истинности или ложности входящих в него высказываний и от множества используемых связок.
Нам пока достаточно будет двух связок "$\neg$" - не (или отрицание) и "$A \supset B$"(если A то B).
$\Phi C_{\text{ИВ}}$ :

- **Алфавит системы**
  $X = \{(,), \neg, \supset, P_1, P_2, ..., P_n, ...\}$.

- **Формулы** $F$
  1)$(P_1), (P_2), ..., (P_n), ...$ - формулы(элементраные формулы).
  2)Если A - формула, то $(\neg A)$ тоже формула.
  3)Если A и B - формула, то $(A \supset B)$ тоже формула.
  4)Других формул нет.

- **Аксиомы системы**
  $A_1.(A \supset (B \supset A))$.
  $A_2.((\neg B \supset \neg A) \supset (A \supset B))$.
  $A_3.(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$.
  - Это схема аксиом.

- **Правила вывода**
  $R_1$. Вместо формул A и B разрешаться подставлять любые формула F полученные по пунктам 1-4.
  $R_2$. Если у нас есть формулы A и $A \supset B$, то тогда есть и формула B.
  $R_1$. - правило подстановки.
  $R_2$. - правило отделения формулы(modus ponens).

**Пример 3.2.** $A_1 : A \supset (B \supset A)$ - *аксиома.*
$A \supset (A \supset A)$ - *тоже аксиома.*

*$A \supset ((A \supset A) \supset A)$ - тоже аксиома.*
*...*
*смотри правило $R_1$.*

*Правило $R_2$:*
*$A, A \supset B$, то есть формула $B$.*
*$A$ - $\underbrace{\text{“квадрат четного числа”}}_{\text{высказывание}}$.*
*$B$ - “квадрат четного числа делится на 4”.*
*$A, A \supset B$ - “квадрат четного числа” если “квадрат четного числа” то “квадрат четного числа делится на 4”.*
*Мы из $A$ и $A \supset B$ получил $B$.*

Нас интересует в этой “игре в формулы”.

**Вопрос 3.1.** *$[F]_{R_1,R_2} =?$ - то есть какие формулы можно получить из аксиом $A_1 - A_3$ используя правила вывода $R_1, R_2$.*
*Докажем, что формула $(A \supset A) \in [F]_{R_1,R_2}$. Все формулы над $\{\neg, \supset\}$. Верно ли что $F = [F]_{R_1,R_2}$?*

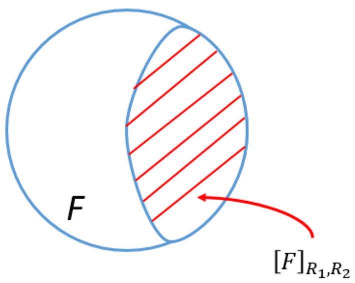



Рис. 7: К вопросу 3.1

Покажем, что формула $(A \supset A) \in [F]_{R_1,R_2}$ для $\forall A \in F$.
1. $(A \supset (\underbrace{B}_{A \supset A \text{ правило } R_1} \supset A))$ - аксиома $A_1$
$(A \supset ((A \supset A) \supset A))$
2. $(A \supset (\underbrace{B}_{A \supset A} \supset \underbrace{C}_{A \text{ правило } R_1})) \supset ((A \supset B) \supset (A \supset C))$ аксиома $A_2$
$(A \supset ((A \supset A) \supset A)) \supset (A \supset (A \supset A)) \supset (A \supset A)$
Из 1 и 2 по правилу $R_2$ (т.р.) $(A \supset (A \supset A)) \supset (A \supset A)$.
3. $(A \supset (\underbrace{B}_{A \text{ правило } R_1} \supset A))$ - аксиома $A_1$.
$(A \supset (A \supset A))$
4. Из 2 и 3 по правилу $R_2, (A \supset A)$ то есть $\forall A \in F$ формула $(A \supset A)$ выводима, то есть $\in [F]_{R_1,R_2}$.
Напоминание: Зная эти таблицы мы можем определить заначение (истинна или ложна) любой формулы из F.

**Пример 3.3.** *1. $A \supset (A \supset A)$. 2. $(A \supset A) \supset A$.*
*Расстановка скобок играет роль!*

**Определение 3.1.** *Формула $\Phi(A_1, ..., A_n)$ называется:*
*1) $\equiv 0$ (тождественно ложной) если при $\forall$ наборе значений формул $A_1, A_2, ..., A_n, \Phi \equiv 0$ (**не выполнимые** формулы).*
*2) $\equiv 1$ (тождественно истинная) если при $\forall$ наборе значений формул $A_1, A_2, ..., A_n, \Phi \equiv 1$ (**тавтологии** или **общезнанимая** формула).*
*3) **Выполнимой** если на некоторых наборах значений формула $A_1, A_2, ..., A_n$. Она принимает $\Phi \equiv 1$, а на остальных $\Phi \equiv 0$.*

| $A$ | $B$ | $(A \supset B)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $A$ | $(\neg A)$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Рис. 8: Напоминание!

| $A$ | $A \supset A$ | $A \supset (A \supset A)$ |
|---|---|---|
| 0 | 1 | $0 \supset (0 \supset 0) = 1$ |
| 1 | 1 | $1 \supset (1 \supset 1) = 1$ |

| $A$ | $A \supset A$ | $(A \supset A) \supset A$ |
|---|---|---|
| 0 | 1 | $\big((0 \supset 0) \supset 0\big) = 1$ |
| 1 | 1 | $\big((1 \supset 1) \supset 1\big) = 1$ |

Рис. 9: К примеру 3.3

Заметим, что $A \supset A \equiv A \vee \bar{A} \equiv 1$ закон “исключенного третьего”!

**Теорема 3.1.** $\forall$ *формула* $\in [F]_{R_1,R_2}$ *является тавтологей.*

Прежде чем доказывать вспомним из круса “дискретной математики”.
По этим таблицам легко проверить, что:

| $A$ | $B$ | $A \vee B$ | $A \wedge B$ | $A \supset B$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Рис. 10: Вспомним из курса ДМ

$$A \supset B \equiv \bar{A} \vee B \Rightarrow \begin{cases} A \vee B \equiv \bar{A} \supset B & (1) \\ A \wedge B \equiv \overline{A \supset \bar{B}} & (2) \\ \quad A \vee \bar{A} \equiv 1 & \end{cases}$$

Поэтому мы далее можем испосльзовать операцию дизъюнкции($\vee$) и конъюнкции($\wedge$), понимая их как формулы (1) и (2). “-” - знак операции отрицания. - Будем его использовать тоже.
*Доказательство* теоремы 3.1:
1. Докажем, что аксиомы являются тавтологиями.
$A \supset (B \supset A) = A \supset (\bar{B} \vee A) = \bar{A} \vee (\bar{B} \vee A) = (\bar{A} \vee A) \vee \bar{B} = 1 \vee \bar{B} \equiv 1$.
2. $(\neg B \supset \neg A) \supset ((\neg B \supset A) \supset B) = (B \vee \bar{A}) \supset ((B \vee A) \supset B) = \overline{(B \vee \bar{A})} \vee \overline{B \vee A} \vee B = (\bar{B} \wedge \bar{A}) \vee (\bar{B} \wedge A) \vee B = \bar{B}(\underbrace{\bar{A} \vee A}_{=1}) \vee B = \bar{B} \vee B \equiv 1$.
3. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)) \equiv 1$. Самостоятельно !
Можно было бы построить таблицу и проверить по ней значения формул. То есть аксиомы $A_1, A_2, A_3$ являются тавтологиями (для любых формул $A, B, C$ из $F$).
Заметим, что если $A$ и $A \supset B$ тавтологии, то $B$ тоже тавтоголия. Если это не так, то при некотором наборе значений подформул, входящих в $B$, она будет $= 0$, но $A = 1$ при этом распределении значений истинности ($A = 1$, тавтология !) Получаем $(A \supset B) \Rightarrow (1 \supset 0) = 0$, т.е. $(A \supset B) \neq 1$. (не тавтология ) -

| *A* | *B* | *C* | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Рис. 11: Таблица к доказательству теоремы

Противоречие. Теорема доказана.
И так, мы доказали теорему 1. (Смысл: если формула $\Phi$ из $F$ выводима из аксиом $A_1 - A_3$ правилами вывода $R_1, R_2$, то $\Phi \equiv 1$($\Phi$ - тавтология)).

**Теорема 3.2.** *Если $\Phi \equiv 1$, то она принадлежит $\left[F\right]_{R_1,R_2}$.*

Доказывается в любом стандартном курсе "Математическая логика".
Объединяя теорему 3.1 и 3.2 получаем

**Теорема 3.3.** *(о полноте исчисления высказываний)*
*Формула $\Phi$ исчисления высказываний принадлежит $\left[F\right]_{R_1,R_2} \Leftrightarrow \Phi \equiv 1$. Иными словами: $\Phi \in \left[F\right]_{R_1,R_2} \Leftrightarrow \Phi \equiv 1$(т.е. общезначима!)*

**Определение 3.2.** *(логического следствия)*
*Формула $\Phi$ является* ***логическим следствием*** *формул $\Phi_1, \ldots, \Phi_n$, если из $\Phi_1, \ldots, \Phi_n = 1$ следует, что и $\Phi = 1$. Т есть если $\Phi_1 \wedge \ldots \wedge \Phi_n = 1$, то и $\Phi = 1$.*

**Утверждение 3.1.** *$\Phi$ является логическим следствием $\Phi_1, \ldots, \Phi_n \Leftrightarrow$ формула ($\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi$)$\equiv 1$(т.е. общезначима).*

*Доказательство*
*Достаточность.* Пусть $\Phi$ - логическое следствие $\Phi_1, \ldots, \Phi_n$, т.е. если $\Phi_1 \wedge \ldots \wedge \Phi_n = 1$, то $\Phi = 1$; если $\Phi_1 \wedge \ldots \wedge \Phi_n = 0$, то $0 \supset \Phi = 1$( по определению $\supset$.) Случая $\underbrace{\Phi_1 \wedge \ldots \wedge \Phi_n}_{=1} \supset \underbrace{\Phi}_{=0}$ быть не может, т.к. $\Phi$ - логическое следствие $\Phi_1, \ldots, \Phi_n$. Следовательно $((\Phi_1 \wedge \ldots \wedge \Phi_n) \supset \Phi) \equiv 1$.
*Необходимость.* Пусть $((\Phi_1 \wedge \ldots \wedge \Phi_n) \supset \Phi) \equiv 1$. Значит если $\Phi_1 \wedge \ldots \wedge \Phi_n = 1$, то и $\Phi = 1$. В противном случае $\Phi_1 \wedge \ldots \wedge \Phi_n = 1$, а $\Phi = 0 \Rightarrow (\Phi_1 \wedge \ldots \wedge \Phi_n) \supset \Phi) = 0$ - т.е. $(\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi)$ - не общезначима. - Противоречие. $\square$

Пусть $I^1_\Phi$ - множество наборов значений всех переменных, входящих в $\Phi$, на которых $\Phi = 1$. Тогда если $\Phi$ - логическое следствие $\Phi_1 \wedge \ldots \wedge \Phi_n$, то $I^1_\Phi \subseteq I^1_{\Phi_1 \wedge \ldots \wedge \Phi_n}$.

**Определение 3.3.** *Пусть $\Phi = \Phi(\rho_1, \ldots, \rho_n)$. Любой набор значений переменных $(\rho_1, \ldots, \rho_n) \in E_2^n = E_2 \times \ldots \times E_2$ ($E_2 = \{0, 1\}$) называется* ***интерпретацией формулы*** *$\Phi$. Множество всех интерпретаций $\Phi$, где $\Phi = 1$, называется* ***моделью*** *для $\Phi$.*

Т.е. если $\Phi$ - общезначима, то $\{I^1_\Phi\} = E_2^n$, если $\Phi$ невыполнима, то $\{I^1_\Phi\} = \Phi$, если $\Phi$ - выполнима, то $\{I^1_\Phi\} = A \subset E_2^n$ и $A \neq 0$.

**Утверждение 3.2.** *Формула* $(\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi)$ *общезначима* $\Leftrightarrow (\Phi_1 \wedge \ldots \wedge \Phi_n \wedge \bar{\Phi})$ *- невыполнима.*

**Определение 3.4.** *Если из* $M_1 \models B_1, M_1 \models B_2, \ldots, M_1 \models B_s$*, то говорим, что множество формул* $M_2 = \{B_1, \ldots, B_s\}$ ***выводимо*** *из множества формул* $M_1$*(обозначение* $M_1 \models M_2$*).*

**Утверждение 3.3.** *Если* $M_1 \models M_2, M_2 \models M_3$*, то* $M_1 \models M_3$*.*

Доказательство самостоятельно !
Теперь у нас есть процесс построения модели ПО: Здесь $A$ - аксиомы - очевидные высказывания о ПО.

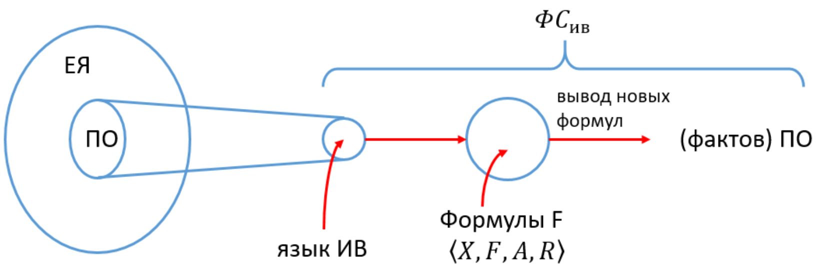



Рис. 12: Процесс построения модели ПО

Мы можем задать вопрос так: верно ли что из фактов $\Phi_1, \ldots, \Phi_n$ следует $\Phi$? Для этого (как мы показали выше) надо доказать, что формула $(\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi) \equiv 1$ или $(\Phi_1 \wedge \ldots \wedge \Phi_n \wedge \bar{\Phi}) \equiv 0$.
*Доказательство.*
Пусть $(\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi) \equiv 1 \Leftrightarrow \overline{(\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi)} \equiv \bar{1} \equiv 0 \Leftrightarrow \overline{(\overline{(\Phi_1 \wedge \ldots \wedge \Phi_n)} \vee \Phi)} \equiv 0$
$\Leftrightarrow \overline{\overline{(\Phi_1, \ldots, \Phi_n)}} \wedge \bar{\Phi} \equiv 0 \Leftrightarrow (\Phi_1, \ldots, \Phi_n) \wedge \bar{\Phi} \equiv 0$.

**Вопрос 3.2.** *Какие булевские тождества здесь были использованы?*

Пусть $M = \{\Phi_1, \ldots, \Phi_n\} \subseteq F$ - некоторое множество формул. Говорим, что формула $\Phi$ **выводима** из $M$ (обозначение $M \models$), если формула $\Phi_1 \wedge \ldots \wedge \Phi_n \supset \Phi$ общезначима.

**Утверждение 3.4.** *1) Если* $M \models A_1$ *и* $M \models A_2$*, то* $M = A_1 \wedge A_2$*.*
*2) Если* $M_1 \models A_1$ *и* $M_2 \models A_2$*, то* $M_1 \wedge M_2 \models A_1 \wedge A_2$*.*
*3) Если* $M \models A_1 \wedge A_2$*, то* $M \models A_1$ *и* $M \models A_2$*.*

Доказать самостоятельно!

**Пример 3.4.** ***Моделирование химических реакций***
*1.* $Na + Cl_2 \to NaCl$
*2.* $CH_4 + O_2 \to CO_2 + H_2O$
*... и так далее*

Обозначим $Na - \Phi_1, Cl_2 - \Phi_2, NaCl - \Phi_3$, тогда реакцию 1. можно записать как $\Phi_1 \wedge \Phi_2 \supset \Phi_3$. $CH_4 - \Phi_1, O_2 - \Phi_2, CO_2 - \Phi_3, H_2O - \Phi_4$, реакцию 2. можно записать как $\Phi_1 \wedge \Phi_2 \supset \Phi_3 \wedge \Phi_4$. Таким образом, аксиомы (они же и элементарные формулы) - это правила вывода.

**Вопрос 3.3.** *Задача. Есть база фактов (база данных) БД=* $\{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$ *- какие-то вещества.*
*Есть база правил (база знаний) - химические реакции. БЗ=* $\{\Phi_1 \wedge \Phi_2 \supset \Phi_5 \wedge \Phi_6, \Phi_5 \wedge \Phi_3 \supset \Phi_7 \wedge \Phi_2, \Phi_7 \wedge \Phi_4 \supset \Phi_8 \wedge \Phi_{10}, \Phi_8 \wedge \Phi_9 \supset \Phi_1\}$*.*
*Можно ли получить вещество* $\Phi_9$*?*

Для ответа нада доказать, что $\Phi(\Phi_1, \Phi_2, \ldots, \Phi_{10}) = ((\Phi_1 \wedge \Phi_2 \supset \Phi_3 \wedge \Phi_4) \wedge (\Phi_1 \wedge \Phi_2 \supset \Phi_5 \wedge \Phi_6) \wedge (\Phi_5 \wedge \Phi_3 \supset \Phi_7 \wedge \Phi_2) \wedge (\Phi_7 \wedge \Phi_4 \supset \Phi_8 \wedge \Phi_{10}) \wedge (\Phi_8 \wedge \Phi_9 \supset \Phi_1)) \supset \Phi_9 \equiv 1$.
Если строить таблицу, то в ней будет $2^{10} = 1024$ бинарных строк длины 10. Как проверять общезначимость?

1. табличный способ

2. метод эквивалентных преобразований

3. метод резолюций и его варианты

4. эвристические методы

**Замечание 3.1.** *(о химии реакциях)*
$2Na + 2H_2O \to 2NaOH + H_2$
$2NaOH + SiO_2 \to Na_2SiO_3 + H_2O$
$Na_2Si_3O_3 + 2HCl \to 2NaCl + H_2SiO_3$
$2NaCl \to 2Na + Cl_2$

# 4 Лекция 4

Рассмотрим формулу $\Phi = (\underbrace{(P_1 \wedge P_2) \supset P_3)}_{\Phi_1} \wedge (\underbrace{(P_4 \wedge P_3)}_{\Phi_2} \supset \underbrace{(P_5 \wedge P_1)}_{\Phi_3})$.

**Вопрос 4.1.** *Определить, к какому классу относится* $\Phi$*:*
*1. общезначимых* $(\equiv 1)$*,*
*2. невыполнимых* $(\equiv 0)$*,*
*3. выполнимых* $(= 1$ *на части наборов значений переменных и* $= 0$ *на другой части)*

**1. Проверка по таблице**
**2. Метод Квайна(семантические деревья)**

$2^5 = 32$ набора

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $\Phi_1$ | $\Phi_2$ | $\Phi_3$ | $\Phi = \Phi_1 \wedge (\Phi_2 \supset \Phi_3)$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | ... | ... | ... | ... |
| | | | | | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ? | ? | ? | ? |
| | | | | | ... | ... | ... | ... |
| 1 | 1 | 1 | 1 | 1 | ... | ... | ... | ... |
| | | | | | 32 | 32 | 32 | 32 |

Рис. 13: Проверка по таблице

Если на всех 32 концевых вершинах $\Phi = 1$, то она общезначима(тавтология); если $\Phi = 0$, то она невыполнима; если есть $\Phi(\alpha_1, \dots, \alpha_5) = 1$ и $\Phi(\beta_1, \dots \beta_5) = 0$, то она выполнима.
Важно: Порядок выбора переменных существенен: принадлежность $\Phi$ к классам $1-3$ можно обнаружить раньше (не перебирая) все варианты. Рассмотрим формулу $\Phi$ (начало лекции): $\Phi = ((P_1 \wedge P_2) \supset P_3) \wedge ((P_4 \wedge P_3) \supset (P_5 \wedge P_1))$.(см. рис. метод Квайна-2)
**3. Метод эквивалентных преобразований**
В дискретной математике приводится система тождеств

- Коммутативность $\vee$ и $\wedge$:
  $x_1 \vee x_2 \equiv x_2 \vee x_1$,
  $x_1 \wedge x_2 \equiv x_2 \wedge x_1$.

- Ассоциативность $\vee$ и $\wedge$:
  $(x_1 \vee x_2) \vee x_3 \equiv x_1 \vee (x_2 \vee x_3)$,
  $(x_1 \wedge x_2) \wedge x_3 \equiv x_1 \wedge (x_2 \wedge x_3)$.

- Дистрибутивность $\wedge$ относительно $\vee$ и наоборот:
  $x_1 \vee (x_2 \wedge x_3) \equiv (x_1 \vee x_2) \wedge (x_1 \vee x_3)$,
  $x_1 \wedge (x_2 \vee x_3) \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3)$.

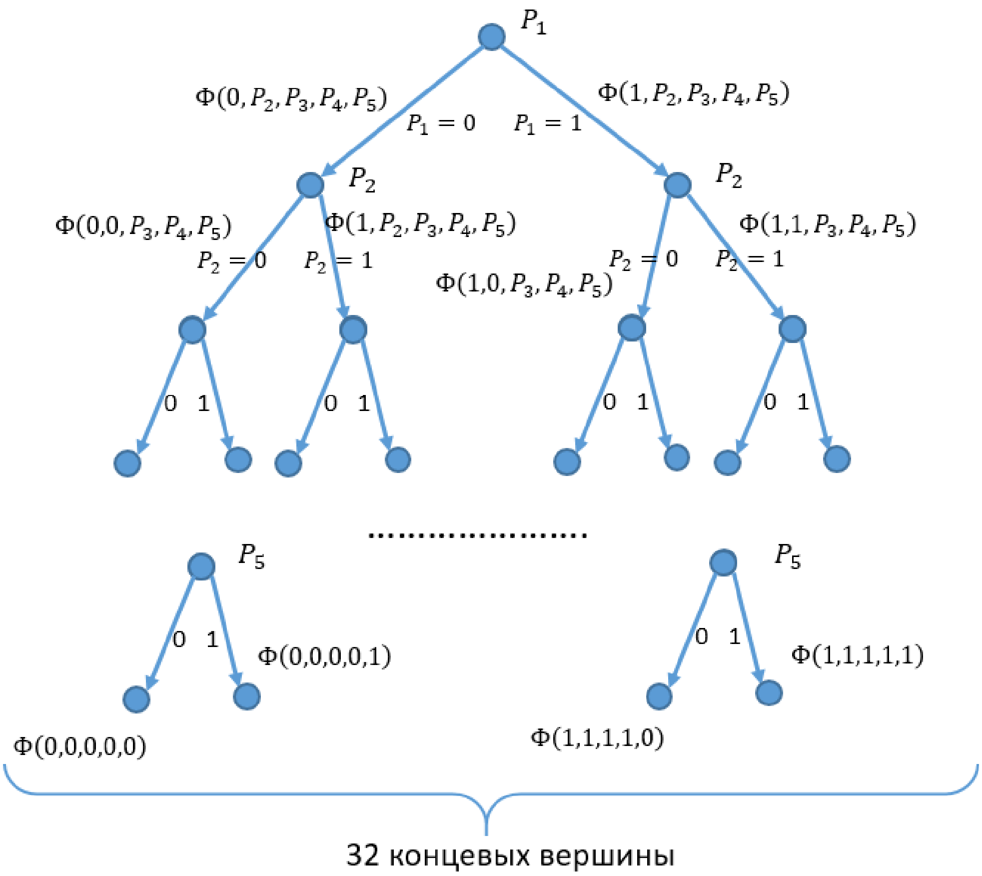



Рис. 14: Метод Квайна-1

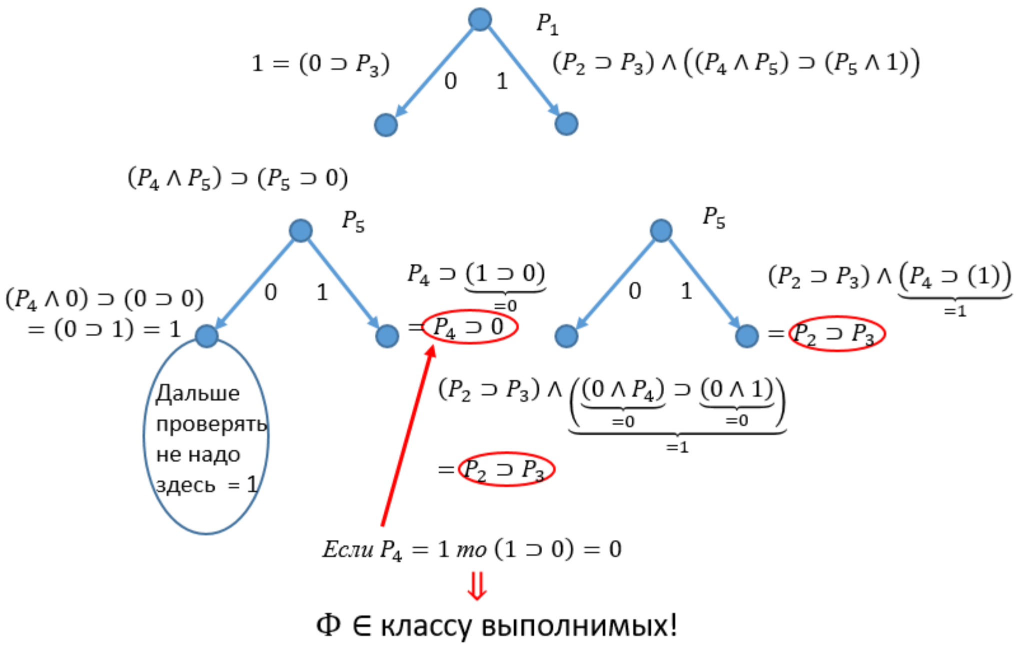



Рис. 15: Метод Квайна-2

- Правила Моргана:
  $\overline{\bar{x}_1 \vee \bar{x}_2} \equiv x_1 \wedge x_2$,
  $\overline{\bar{x}_1 \wedge \bar{x}_2} \equiv x_1 \vee x_2$.

- Правило снятия двойного отрицания:
  $\bar{\bar{x}} \equiv x$.

- $x \vee 1 \equiv 1 \quad x \wedge 1 \equiv x$
  $x \vee 0 \equiv x \quad x \wedge 0 \equiv 0$
  $x \vee \bar{x} \equiv 1 \quad x \wedge \bar{x} \equiv 0$
  $x \vee \bar{x} \equiv x \quad x \wedge x \equiv x$

В курсе "дискретной математики" про систему 1-6 доказывают, что она обладает свойством синтаксической полноты, то есть для любых двух формул( $\Phi = \{\wedge, \vee, -\}$) за конечиное число шагов(эквивалентных преобразований) можно выяснить эквивалентны формулы или нет.

**Пример 4.1.** $\Phi_1 = x, \Phi_2 = (x \wedge y) \vee (x \wedge \bar{y})$
$\Phi_1 = x \underbrace{\Rightarrow}_{x\wedge 1 \equiv x} x \wedge 1 \underbrace{\Rightarrow}_{y \vee \bar{y} \equiv 1} x \wedge (y \vee \bar{y}) \underbrace{\Rightarrow}_{\text{дистриб.}} (x \wedge y) \vee (x \wedge \bar{y})$;
*То есть формулы* $\Phi_1 \backsim \Phi_2$ *("$\backsim$" - эквивалентность из 3).*
$\Phi_2$ *выводится из* $\Phi_1$ *с помощью тождеств 1-6 если есть цепочка подстановок* $\Phi_1 \underbrace{\Rightarrow}_{R_{i_1}} \Phi_1' \underbrace{\Rightarrow}_{R_{i_2}} \ldots \underbrace{\Rightarrow}_{R_{i_s}} \Phi_S' = \Phi_2$. *Здесь* $R_{i_1}, ..., R_{i_s}$ *взяты из 1-6.*
*Подстановка в формулу:* $\Phi = (..., \Phi_1, ...) \Rightarrow \Phi' = \Phi(..., \Phi_2, ...)$ *если есть тождество* $\Phi_1 \equiv \Phi_2$.
*Если с помощью тождественных преобразований удается показать, что* $\Phi \equiv 1 (\Phi \equiv 0)$ *то* $\Phi$ *- общезначаем(невыполнима) в остальных случаях* $\Phi$ *- выполнима.*

**Замечание 4.1.** $x \supset y \equiv \bar{x} \vee y, \quad x \vee y \equiv \bar{x} \supset y, \quad x \wedge y \equiv \overline{x \supset \bar{y}}$.

То есть, любое сложное высказывание построенное с помощью связок $\{\wedge, \vee, -\}$ можно преобразовать в эквивалентное ему высказывание с использованием $\{\supset, -\}$ и наоборот.

**Пример 4.2.** *Вместо* $x \wedge y$ *будем писать просто* $xy$.
*1.* $\Phi_1 = \bar{x}\bar{y} \vee \boldsymbol{xy} \vee \boldsymbol{\bar{x}y} \vee x\bar{y} \underbrace{\Rightarrow}_{\text{коммут. } \vee} \bar{x}\bar{y} \vee \bar{x}y \vee xy \vee x\bar{y} \underbrace{\Rightarrow}_{\text{дистриб.}} \bar{x}\underbrace{(\bar{y} \vee y)}_{\bar{y} \vee y \equiv 1} \vee xy \vee x\bar{y} \Rightarrow \underbrace{\bar{x} \cdot 1}_{\bar{x}\cdot 1 = \bar{x}} \vee xy \vee x\bar{y} \underbrace{\Rightarrow}_{\text{дистриб.}} \bar{x} \vee x\underbrace{(y \vee \bar{y})}_{y \vee \bar{y} = 1} \Rightarrow \bar{x} \vee x \cdot 1 = \bar{x} \vee x \equiv 1$.
*Формула* $\Phi_1 \equiv 1$, *т.е. тавтология.*
*2.*$(x \supset \bar{y}) \wedge (y \supset x) \wedge (y \supset y) \Rightarrow (\bar{x} \vee \bar{y}) \wedge (\bar{y} \vee x) \wedge (\bar{y} \vee y) \Rightarrow \bar{x} \wedge (\bar{y} \wedge x) \vee \bar{y} \wedge (\bar{y} \vee x) \wedge 1 \Rightarrow \bar{x}\bar{y} \vee \underbrace{\bar{x}x}_{=0} \vee \underbrace{\bar{y}\bar{y}}_{=0} \vee \bar{y}x \Rightarrow \bar{x}\bar{y} \vee \bar{y}x \Rightarrow \bar{y}\underbrace{(x \vee \bar{x})}_{=1} \Rightarrow \bar{y}$ *- выполнима (1)* $y = 0$ *то* $\bar{y} = 1$.*(2)* $y = 1$ *то* $\bar{y} = 0$.
*3.* $\underbrace{(x \vee y)(\bar{x} \vee y)}_{=(\underbrace{x\bar{x}}_{=0} \vee xy \vee y\bar{x} \vee \underbrace{yy}_{=y})} \wedge \underbrace{(x \vee \bar{y})(\bar{x} \vee \bar{y})}_{=(\underbrace{x\bar{x}}_{=0} \vee x\bar{y} \vee \bar{y}\bar{x} \vee \underbrace{\bar{y}\bar{y}}_{\bar{y}})} = (xy \vee y\bar{x} \vee y)(x\bar{y} \vee \bar{y}\bar{x} \vee \bar{y}) = (y\underbrace{(x \vee \bar{x} \vee 1)}_{=1})(\underbrace{x \vee \bar{x}}_{=1} \vee 1)\bar{y} = y \cdot \bar{y} \equiv 0$.
*Формула невыполнима.*
*Трудности реализации на компьютер = Наиболее удобные для программирования метод - резолюция.*

**Метод резолюций в $\Phi C_{\text{ИВ}}$.**
Далее мы будем рассматривать формулы специального вида:
Формула $L$ называется **литерой**, если $L = P_i$ или $L = \bar{P}_i$, где $P_i \in X$ - пропозициональная переменная (атомарная формула).
**Дизъюнктом** называется формула вида $(L_1 \vee ... \vee L_n)$, где $L_i$ - литера($i = 1 \ldots n$).

**Определение 4.1.** *Пара литер вида* $(L, \bar{L})$ *называется* ***контрарной парой***.

**Вопрос 4.2.** *Имеет ли решения система булерских уравнений?*

$$\begin{cases} P_1 \vee \bar{P}_2 \vee P_3 = 0 \\ \bar{P}_1 \vee P_2 \vee \bar{P}_3 = 0 \\ \bar{P}_1 \vee \bar{P}_2 \vee P_3 = 0 \end{cases} \tag{1}$$

$P_i \in \{0, 1\}, i = 1, 2, 3.$
*Система* (1) *имеет решение* $\Leftrightarrow$ *формула* $(P_1 \vee \bar{P}_2 \vee P_3) \wedge (\bar{P}_1 \vee P_2 \vee \bar{P}_3) \wedge (\bar{P}_1 \vee \bar{P}_2 \vee P_3)$ *не является тавтологией. Обратите внимание, что уравнения системы* (1) *являются дизъюнктами.*

**Определение 4.2.** *Формула* $K$ *вида* $K = D_1 \wedge ... \wedge D_s$ *где каждый* $D_i$ *является дизъюнктом, называется* ***конъюнктивной нормальной формой(КНФ)***.

**Пример 4.3.** $K = \underbrace{(P_1)}_{*} \wedge \underbrace{(\bar{P}_1 \vee P_2)}_{*} \wedge \underbrace{(P_1 \vee P_2 \vee P_3)}_{*}$. * - *дизъюнкты.*

**Задача о выполнимости КНФ:**
Дана КНФ= $D_1 \wedge ... \wedge D_s = K(P_1, ..., P_n)$. Существует ли набор значений переменных $(\alpha_1, ..., \alpha_n) \in E_\alpha^n$, что $K(\alpha_1, ..., \alpha_n) = 1$.
К этой задаче можно свести много других задач. Например, задача о выяснении является ли формула $\Phi$ логическим следствием $\Phi_1, ..., \Phi_s$, т.е.
$(\Phi_1 \wedge ... \wedge \Phi_n \supset \Phi) \equiv 1 \Leftrightarrow (\Phi_1 \wedge ... \wedge \Phi_n \wedge \bar{\Phi}) \equiv 0.$
Преобразуем $\Phi_1, ..., \Phi_n$ к эквивалентным $\Phi_1^{'}, ..., \Phi_n^{'}, \Phi^{'}$, имеющим вид дизъюнктов и получим задачу о выполнимости КНФ.

**Замечание 4.2.** *В курсе "ДМ" доказывали, что* $\forall$ *булевскую функцию можно записать в виде СКНФ(совершенной конъюнктивной нормальной формы) - это частный случай представления в виде КНФ.*

**Пример 4.4.** $(P_1 \supset P_2) \wedge (P_2 \supset P_1)$.
$(P_1 \vee \bar{P}_2) \wedge (\bar{P}_1 \vee P_2)$ - *совершенная КНФ.*

| $P_1$ | $P_2$ | $P_1 \supset P_2$ | $P_2 \supset P_1$ | $(P_1 \supset P_2) \wedge (P_2 \supset P_1)$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Рис. 16: Таблица к примеру 4.4

*Можно было бы воспользоваться* $p \supset q \equiv \bar{p} \vee q \Rightarrow (P_1 \supset P_2) \wedge (P_2 \supset P_1) \equiv (\bar{P}_1 \vee P_2) \wedge (\bar{P}_2 \vee P_1)$.

Итак, далее будет считать, что нат дана формула $K = D_1 \wedge ... D_s$, где $D_1 \wedge ... D_s$ - дизъюнкты. Нам надо выяснить, выполнима $K = K(p_1, ..., p_n)$ или нет.
Пусть $D_1 = (... \vee L \vee ...)$ и $D_2 = (... \vee \bar{L} \vee ...)$ два дизъюнкта: в первом есть литера $L$, во втором - $\bar{L}$(есть контрарная пара).

**Определение 4.3.** ***Резольветой*** *$D_1$ и $D_2$ называется дизъюнкт $D$, получающийся из $D_1$ и $D_2$ вычеркиванием(удалением $L$ и $\bar{L}$) и объединением остальных членов $D_1$ и $D_2$.*

**Пример 4.5.** $D_1 = (L_1 \vee L_2 \vee L_3), D_2 = (L_1, \bar{L}_2, \bar{L}_3), D = (\underbrace{L_1 \vee L_3}_{D_1 \setminus \{L_2\}} \vee \underbrace{L_1 \vee \bar{L}_3}_{D_2 \setminus \{\bar{L}\}})$.

Дизъюкт $D$(резольвента $D_1$ и $D_2$) будет обозначаться $\mathrm{Res}(D_1, D_2)$
$D_1 = S_1 \vee \cancel{L}, D_2 = S_2 \vee \cancel{\bar{L}} \Rightarrow D = S_1 \vee S_2, \quad \mathrm{Res}(D_1, D_2) = D.$
Если $D_1 = L$, а $D_2 = \bar{L}$(или наоборот), то $Res(L, \bar{L}) = \emptyset$ - пустой дизъюнкт.

Обычно мы будем исследовать не формулы вида $(\Phi_1 \wedge ... \wedge \Phi_n \supset \Phi) \stackrel{?}{\equiv} 1$, а эквивалентную ей формулу $(\Phi_1 \wedge ... \wedge \Phi_n \wedge \bar{\Phi}) \equiv 0$, то есть будет вести доказательство "от противного"(то есть строить "систему опровержения").

**(*)**Формула $\Phi$ выводима из $\{\Phi_1, ...\Phi_n\} = S$ с помощью резолютивного вывода(метода), если существует такая последовательность формул $\Phi_1^{'}, ..., \Phi_p^{'}$, что $\Phi_p^{'} = \Phi$, а любая $\Phi_1^{'}, ..., \Phi_p^{'}$ либо $\in S$, либо получена из $S$ по правилу (метод) резолюция.

**Теорема 4.1.** *Если $S \supset \Phi$ по правилу (*), то $(S \supset \Phi) \equiv 1$.*

*Доказательство.* (От противного) $(S \supset \Phi) \equiv 1 \Leftrightarrow (S \wedge \bar{\Phi}) \equiv 0$ или $((\Phi_1 \wedge \ldots \wedge \Phi_n) \wedge \bar{\Phi}) \equiv 0(**)$
1. $\Phi \in S$, т.е. формула (**) имеет вид $(\Phi_1 \wedge \ldots \wedge \underbrace{\Phi \wedge \Phi_i \ldots \Phi_n) \wedge}_{=0} \bar{\Phi} \Rightarrow$ (**) тоже $\equiv 0$.

2. $\Phi \in S$, доказываем методом математической индукции (по длине вывода - числу шагов в (*) равному $k$).
Базис индукции $k = 1$, т.е. $\Phi$ - получается из $S$ за один шаг (за одно применение правила резолюции) $\Rightarrow \exists \Phi_i = \Phi_i^{'} \vee L$ и $\Phi_j = \Phi_j^{'} \vee \bar{L}$ и $\Phi_i, \Phi_j \in S$ и $\mathrm{Res}(\Phi_i, \Phi_j) = \Phi$, т.е. $\mathrm{Res}(\Phi_i^{'} \vee L, \Phi_j^{'} \vee \bar{L}) = c^{'} \vee c^{''} = \Phi$. Покажем, что если $\Phi_i, \Phi_j$ - тавталогии, то и $\Phi$ - тавталогия. Распишем это $(\Phi_i^{'} \vee L) \wedge (\Phi_j^{'} \vee \bar{L}) \supset (\Phi_i^{'} \vee \Phi_i^{'} \vee \Phi_j^{'})$. Если $\Phi$ не тавтология, то найдется набор значений переменных, входящих в $\Phi$, на котором она обратится в 0, т.е. $\Phi_i^{'} = 0, \Phi_j^{'} = 0$. Но $(\Phi_i^{'} \vee L) \wedge (\Phi_j^{'} \vee \bar{L}) = 1$ на этом же наборе $\underbrace{(\Phi_i \equiv 1, \Phi_j \equiv 1)}_{\text{по условию}} \Rightarrow (0 \vee L) \wedge (0 \vee \bar{L}) = 1$, т.е. $0 = 1$ - противоречие.
Предположение индукции Пусть теорема верна для любого вывода длины (числа шагов) $= i < k$.
Индуктивный переход (шаг индукции) Докажем теорему для $i = k$. Пусть имеется резолютивный вывод, $\Phi$ из $S = \{\Phi_1, \ldots, \Phi_n\}$ длины $k$, т.е.
$\Phi_1^{'}, \Phi_2^{'}, \ldots, \underbrace{\Phi_i^{'}, \Phi_{i+1}^{'}, \ldots, \Phi_j^{'}}_{\mathrm{Res}(\Phi_i^{'}, \Phi_j^{'}) = \Phi_k^{'} = \Phi}, \ldots, \Phi_k^{'} = \Phi$. При этом $i, j < k$.
По предположению индукции $(S \supset \Phi_i^{'}) \equiv 1$, и $(S \supset \Phi_j^{'}) \equiv 1 \Rightarrow S \supset (\Phi_i^{'} \wedge \Phi_j^{'}) \equiv 1$. Но $\Phi = \mathrm{Res}(\Phi_i^{'}, \Phi_j^{'})$ и получена за один шаг $\Rightarrow (\Phi_i^{'} \wedge \Phi_j^{'}) \supset \Phi$ тоже тавтология (см. базис индукции).
Итак, $S \supset \Phi_i^{'} \wedge \Phi_j^{'}$ - тавтология, $\Phi_i^{'} \wedge \Phi_j^{'} \supset \Phi$ тоже тавтология $\Rightarrow$ по свойству транзитивности $\Rightarrow (S \supset \Phi) \equiv 1$. Теорема доказана.

Таким образом, метод резолюций не выводит нас из множества тавтологий. Заметим, что правило modus popens будет частным случаем: $D_1 = L, D_2 = D \vee \bar{L} \Rightarrow \mathrm{Res}(L, D \vee \bar{L}) = D$.
Метод резолюций более удобен для компьютерной модели "доказательства от противного". Позже мы докажем главный результат этого раздела:

**Теорема 4.2.** *Если множество дизъюнктов $D_1, \ldots D_n$ не выполнимо то из него методом резолюций можно получить $\emptyset$ - дизъюнкт.*

# 5 Лекция 5

Покажем, что $\{P_1 \vee P_2, P_1 \vee \bar{P}_2, \bar{P}_1 \vee P_2, \bar{P}_1 \vee \bar{P}_2\}$ невыполнимо, то есть из него методом резолюции можно полусить $\emptyset$ - дизъюникт.

$$\underbrace{P_1 \vee P_2}_{①}, \underbrace{P_1 \vee \bar{P}_2}_{②}, \underbrace{\bar{P}_1 \vee P_2}_{③}, \underbrace{\bar{P}_1 \vee \bar{P}_2}_{④}$$

$$① + ② = P_1 \vee P_1 \equiv \underbrace{P_1}_{⑤} \quad ⑤ + ③ = \underbrace{P_2}_{⑥} \quad ⑥ + ④ = \underbrace{\bar{P}_1}_{⑦} \quad ⑤ + ⑦ = \emptyset$$

Здесь мы изобразили получение(вывод) $\emptyset$ - дизъюникт в виде "дерева". На самом деле это списки дизъюнктов.

$$S = \{\underbrace{P_1 \vee P_2}_{①}, \underbrace{P_1 \vee \bar{P}_2}_{②}, \underbrace{\bar{P}_1 \vee P_2}_{③}, \underbrace{\bar{P}_1 \vee \bar{P}_2}_{④}\} \cup \{P_1, P_2, \bar{P}_2, P_2 \vee \bar{P}_2\}.$$

$$① + ② = P_1 \quad ① + ③ = P_2 \quad ② + ④ = \bar{P}_2 \quad ① + ④ = P_2 \vee \bar{P}_2$$

**Определение 5.1.** *Обозначим через* $S_1 = [S]^1_{Res}$ *множество дизъюнктов, которые можно получить из множества* $S$ *за одны шаг методом резолюций.* $S_2 = [S]^2_{Res} = [\{S\} \cup \{S_1\}]^1_{Res}, .., S_i = [S \cup \{S_1\} \cup ... \cup \{S_{i-1}\}]^1_{Res}$

Любую литеру можно резольвировать столько раз, сколько требуется. Компьютерная реализация(обсуждение).

**Вопрос 5.1.** *Как запрограммировать метод резолюций?*

$$S = \begin{cases} (1) P \vee Q \\ (2) \backsim P \vee Q \\ (3) P \vee \backsim Q \\ (4) \backsim P \vee \backsim Q \text{ получено из:} \\ (5) Q \text{ из } (1) \text{ и } (2), \end{cases}$$

$$S_1 = [S]^1_{Res} \begin{cases} (6) P \text{ из } (1) \text{ и } (3), \\ (7) Q \vee \backsim Q \text{ из } (1) \text{ и } (3), \\ (8) P \vee \backsim P \text{ из } (1) \text{ и } (4), \\ (9) Q \vee \backsim Q \text{ из } (2) \text{ и } (3), \\ (10) P \vee \backsim P \text{ из } (2) \text{ и } (3), \\ (11) \backsim P \text{ из } (2) \text{ и } (4), \\ (12) \backsim Q \text{ из } (3) \text{ и } (4), \end{cases}$$

$$S_2 = [S]^2_{Res} \begin{cases} (13) P \vee Q \text{ из } (1) \text{ и } (7), \\ (14) P \vee Q \text{ из } (1) \text{ и } (8), \\ (15) P \vee Q \text{ из } (1) \text{ и } (9), \\ (16) P \vee Q \text{ из } (1) \text{ и } (10), \\ (17) Q \text{ из } (1) \text{ и } (11), \\ (18) P \text{ из } (1) \text{ и } (12), \\ (19) Q \text{ из } (2) \text{ и } (6), \\ (20) \backsim P \vee Q \text{ из } (2) \text{ и } (7), \\ (21) \backsim P \vee Q \text{ из } (2) \text{ и } (8), \\ (22) \backsim P \vee Q \text{ из } (2) \text{ и } (9), \\ (23) \backsim P \vee Q \text{ из } (2) \text{ и } (10), \\ (24) \backsim P \text{ из } (2) \text{ и } (12), \\ (25) P \text{ из } (3) \text{ и } (5), \\ (26) P \vee \backsim Q \text{ из } (3) \text{ и } (7), \\ (27) P \vee \backsim Q \text{ из } (3) \text{ и } (8), \\ (28) P \vee \backsim Q \text{ из } (3) \text{ и } (9), \\ (29) P \vee \backsim Q \text{ из } (3) \text{ и } (10), \\ (30) \backsim Q \text{ из } (3) \text{ и } (11), \\ (31) \backsim P \text{ из } (4) \text{ и } (5), \\ (32) \backsim Q \text{ из } (4) \text{ и } (6), \\ (33) \backsim P \vee \backsim Q \text{ из } (4) \text{ и } (7), \\ (34) \backsim P \vee \backsim Q \text{ из } (4) \text{ и } (8), \\ (35) \backsim P \vee \backsim Q \text{ из } (4) \text{ и } (9), \\ (36) \backsim P \vee \backsim Q \text{ из } (4) \text{ и } (10), \\ (37) Q \text{ из } (5) \text{ и } (7), \\ (38) Q \text{ из } (5) \text{ и } (9), \\ (39) \emptyset \text{ из } (5) \text{ и } (12). \end{cases}$$

(5)-(38) $Res$ 2 шага методом резольвент.
Появляется очень много резольвент.

**Замечание 5.1.** *Пусть* $D = (L_{i_1} \vee ... \vee L_{i_s}) \equiv 1$*, тогда множество дизъюнктов* $S \equiv 1 \Leftrightarrow S \wedge D \equiv 1 (S \equiv 0 \Leftrightarrow S \wedge D \equiv 0)$*. Иными словами дизъюнкт, являющийся тавтологией можно удалять из списка* $[S]^1_{Res}$*.*

Для этого достаточно заметить, что если $\underbrace{D_1 \wedge ... \wedge D_n}_{S} \wedge D \underbrace{\equiv}_{D \equiv 1} D_1 \wedge ... \wedge \underbrace{D_n \wedge 1}_{D_n} = \underbrace{D_1 \wedge ... \wedge D_n}_{S}$.

**Утверждение 5.1.** *Дизъюнкт* $D \equiv 1 \Leftrightarrow$ *в нем есть хотя бы одна контрараная пара литер.*

Доказательство

$\Rightarrow$

Пусть в $D$ есть контрарная пара литер $L_i$ и $\bar{L}_i$; $D = (... \vee L_i \vee ... \vee \bar{L}_i \vee ...) \underbrace{=}_{\text{коммутативн.``}\vee\text{''}} (... \vee L_i \vee \bar{L}_i \vee ...) = (... \vee 1 \vee ...) \equiv 1$.

$L_i \vee \bar{L}_i \equiv 1 \Rightarrow$ если $L_i = P_{j_i}$ то $\bar{L}_i = \bar{P}_{j_i} \Rightarrow L_i \vee \bar{L}_i = P_{j_i} \vee \bar{P}_{j_i} \equiv 1$.

Аналогично если $L_i = \bar{P}_{j_i}$ то $\bar{L}_i = \bar{\bar{P}}_{j_i} = P_{i_j} \Rightarrow P_{i_j} \vee \bar{P}_{i_j} \equiv 1$.

$\Leftarrow$

Пусть в $D$ нет контрарной пары, а $D \equiv 1$. Если нет контрарной пары, то $D$ имеет вид: $D = (P_{i_1}^{\sigma_{i_1}}, ..., P_{i_n}^{\sigma_{i_n}})$ и все $P_{i_j} (j = 1 \dots n)$ разные. Утверждение доказано.

С учетом утверждения 1 тожно уменьшить число резольвент, соблюдая правила:

(1) $D \vee D \equiv D$ (удаление дублей)

(2) $D \equiv 1$ (удаление тавтологий)

Постмотрим как сократится вывод (со стр. 2)

$\left.\begin{array}{l} (1) P \vee Q \\ (2) \backsim P \vee Q \\ (3) P \vee \backsim Q \\ (4) \backsim P \vee \backsim Q \text{ получено из:} \end{array}\right\}$

(5)$Q$ из (1) и (2),

(6)$P$ из (1) и (3),

$\left.\begin{array}{l} \cancel{(7) Q \vee \backsim Q} \text{ из (1) и (3),} \\ \cancel{(8) P \vee \backsim P} \text{ из (1) и (4),} \\ \cancel{(9) Q \vee \backsim Q} \text{ из (2) и (3),} \\ \cancel{(10) P \vee \backsim P} \text{ из (2) и (3),} \end{array}\right\}$ тавтологии

(11) $\backsim P$ из (2) и (4),

(12) $\backsim Q$ из (3) и (4),

(13)$P \vee Q$ из (1) и (7),

$\left.\begin{array}{l} \cancel{(14) P \vee Q} \text{ из (1) и (8),} \\ \cancel{(15) P \vee Q} \text{ из (1) и (9),} \\ \cancel{(16) P \vee Q} \text{ из (1) и (10),} \\ \cancel{(17) Q} \text{ из (1) и (11),} \\ \cancel{(18) P} \text{ из (1) и (12),} \\ \cancel{(19) Q} \text{ из (2) и (6),} \end{array}\right\}$ дубли

(20) $\backsim P \vee Q$ из (2) и (7),

$\left.\begin{array}{l} \cancel{(21) \backsim P \vee Q} \text{ из (2) и (8),} \\ \cancel{(22) \backsim P \vee Q} \text{ из (2) и (9),} \\ \cancel{(23) \backsim P \vee Q} \text{ из (2) и (10),} \\ \cancel{(24) \backsim P} \text{ из (2) и (12),} \\ \cancel{(25) P} \text{ из (3) и (5),} \\ \cancel{(26) P \vee \backsim Q} \text{ из (3) и (7),} \\ \cancel{(27) P \vee \backsim Q} \text{ из (3) и (8),} \\ \cancel{(28) P \vee \backsim Q} \text{ из (3) и (9),} \\ \cancel{(29) P \vee \backsim Q} \text{ из (3) и (10),} \\ \cancel{(30) \backsim Q} \text{ из (3) и (11),} \\ \cancel{(31) \backsim P} \text{ из (4) и (5),} \\ \cancel{(32) \backsim Q} \text{ из (4) и (6),} \\ \cancel{(33) \backsim P} \vee \backsim Q \text{ из (4) и (7),} \\ \cancel{(34) \backsim P} \vee \backsim Q \text{ из (4) и (8),} \\ \cancel{(35) \backsim P} \vee \backsim Q \text{ из (4) и (9),} \\ \cancel{(36) \backsim P} \vee \backsim Q \text{ из (4) и (10),} \\ \cancel{(37) Q} \text{ из (5) и (7),} \\ \cancel{(38) Q} \text{ из (5) и (9),} \end{array}\right\}$ дубли

(39)$\emptyset$ из (5) и (12).

введем третье правило (упорядочение букв в дизъюнктах) например:

$\underbrace{P_1}_{\text{старшая буква}} \geqslant P_2 \geqslant P_3$ Начинаем делать операцию с начала по старшей букве $P_1$, потом $P_2$, потом $P_3$.
$S = \{\underbrace{P_1 \vee P_2}_{①}, \underbrace{P_1 \vee \bar{P}_2}_{②}, \underbrace{\bar{P}_1 \vee P_2}_{③}, \underbrace{\bar{P}_1 \vee \bar{P}_2}_{④}\}$

① + ③ $= P_2$
① + ④ $= \bar{P}_2$
② + ③ $= \cancel{(P_2 \vee \bar{P}_2) \equiv 1}$ - тавтология больше.
② + ④ $= \cancel{\bar{P}_2}$ ~~- дубль~~ - тавтология больше.
$S_1 = S \cup \{P_2, \bar{P}_2\}$ - по $P_1$ больше нет резольвент, начинаем с $P_2$.
$S_1 = \{\underbrace{P_1 \vee P_2}_{①}, \underbrace{P_1 \vee \bar{P}_2}_{②}, \underbrace{\bar{P}_1 \vee P_2}_{③}, \underbrace{\bar{P}_1 \vee \bar{P}_2}_{④}, \underbrace{P_2}_{⑤}, \underbrace{\bar{P}_2}_{⑥}\}$

① + ② $= P_1$
① + ④ $= \cancel{P_1 \vee \bar{P}_1}$ - тавтологии.
② + ③ $= \cancel{\bar{P}_1 \vee P_1}$ - тавтологии.
③ + ④ $= \bar{P}_1$
⑤ + ⑥ $= \emptyset$
Четвертое правило(предпочтение однолитеральным дизъюнктам).
Резольвируем сначала однолитерные дизъюнкты(если они есть).
$S_1 = \{P_1 \vee P_2, P_1 \vee \bar{P}_2, \bar{P}_1 \vee P_2, \bar{P}_1 \vee \bar{P}_2, P_2, \bar{P}_2\}$, где $P_2, \bar{P}_2$ - однолитерные дизъюнкты $\Rightarrow \emptyset$.
**Использование интерпретации**.
Пусть $S = \{D_1, \ldots, D_n\}$ не выполнимое множество дизъюнктов, т.е. $D_1 \wedge \ldots \wedge D_n \equiv 0$ зависящее от переменных $P_1, \ldots, P_n$. $I_n$ - некоторая интерпретация этого множества $S$, то $I_n = \{P_1 = \alpha_1^{\sigma_1}, \ldots, P_n = \alpha_n^{\sigma_n}\}$, где $\alpha_i^{\sigma_i} = \{0, 1\}$.
Напомню обозначение из курса дискретной математики: $P^\sigma = (P \wedge \sigma) \vee (\bar{P} \wedge \bar{\sigma})$, $P$ - булевская переменная, $\sigma$ - константа $\{0, 1\}$. Например: $P^0 = (P \wedge 0) \vee (\bar{P} \wedge \bar{0}) = \bar{P}, P^1 = (P \wedge 1) \vee (\bar{P} \wedge \bar{1}) = P$.
При фиксированной интерпретации $I_n$ множество $S = S^0_{I_n} \bigcup S^1_{I_n}$, где $S^0_{I_n}$ - множество тех дизъюнктов $S$, которые $= 0$ при этой интерпретации, а $S^1_{I_n}$ - которые равны 1.

**Утверждение 5.2.** *Для любого невыполнимого множества дизъюнктов $S$ имеем $S^0_{I_n} \neq \emptyset$ и $S^1_{I_n} \neq \emptyset$ при любой интерпретации $I_n$.*

*Доказательство.* Если $S^0_{I_n} = \emptyset \Rightarrow$ на $I_n$ все дизъюнкты из $S$ принимают значения 1, т.е. множество $S$ - выполнимо $\Rightarrow$ противоречие. Если $S^1_{I_n} = \emptyset \Rightarrow$ в дизъюнктах из $S$ нет ни одной <u>контрарной пары</u> литер, т.е. присутствуют $L = P_i^{\sigma_i}$, но нет $\bar{L} = P_i^{\bar{\sigma}_i}$, присвоим $P_i = \sigma_i \Rightarrow$(см. напоминание стр. 6)$\Rightarrow$ все дизъюнкты из $S$ будут на этой интерпретации $I_n(I_n = \{P_1 = \sigma_1, \ldots, P_n = \sigma_n\})$ будут равны 1. То есть, множество $S$ будем выполнимым. Противоречие.

**Пример 5.1.** $S = \{(P_1 \vee P_2 \vee \bar{P}_3), (P_1 \vee \bar{P}_3)\}, I_1 = \{P_1 = 1, P_2 = 1, P_3 = 0\}, (1 \vee 1 \vee \bar{0} = 1), (1 \vee \bar{0}) = 1 \Rightarrow S = 1$ *на* $I_1$.

Используем интерпретацию следующим образом: Фикструем $I_n$, представляем $S = S^0_{I_n} \bigcup S^1_{I_n}$. $S^0_{I_n}, S^1_{I_n}$ - Res, т.е. внутри $S^0_{I_n}$ и внутри $S^1_{I_n}$ дизъюнкты не резольвируются. Res ищется только между дизъюнктами из $S^0_{I_n}$ и $S^1_{I_n}$.

**Пример 5.2.** $S = \{P_1 \vee P_2, P_1 \vee P_3, P_1 \vee \bar{P}_2 \vee \bar{P}_3, \bar{P}_1\}, I_n = (0, 0, 0), S^0_{(0,0,0)} = \{P_1 \vee P_2, P_1 \vee P_3\}, S^1_{(0,0,0)} = \{P_1 \vee \bar{P}_2 \vee \bar{P}_3, \bar{P}_1\}$
*Из них:* $P_1 \vee \bar{P}_3, P_2, P_1 \vee \bar{P}_2, P_3$. $\left[S^0_{(0,0,0)}\right]^1 = S^0_{(0,0,0)} \bigcup \{P_2, P_3\}, \left[S^1_{(0,0,0)}\right]^1 = S^1_{(0,0,0)} \bigcup \{P_1 \vee \bar{P}_3, P_1 \vee \bar{P}_2\}$.
$\{P_1 \vee P_2, P_1 \vee P_3, P_2, P_3\}, \{P_1 \vee \bar{P}_2 \vee \bar{P}_3, \bar{P}_1, P_1 \vee \bar{P}_3, P_1 \vee \bar{P}_2\}$.
*Из них:* $P_1$*(4-ая с 3-ой), потом* $P_1$ *с* $\bar{P}_1$ *получаем* $\emptyset$.

Выбранная интерпретация в процессе вывода $\emptyset$ - дизъюнкта не меняется.
Частая ошибка студентов при реализации задач: $S = \{p \vee q, \bar{p} \vee \bar{q}\} \Rightarrow \emptyset$ - это не верно.
Надо попарно получить $(q \vee \bar{q}), (p \vee \bar{p})$ - тавтологии, а не $\emptyset$ - дизъюнкт.

# 6 Лекция 6

Пусть $S = \{D_1, \dots, D_n\}$ - некоторое множество дизъюнктов.

**Определение 6.1.** *Обозначим через $|S|$ **число дизъюнктов** в $S$, а через $\|S\|$ - **число вхождений литер** во все дизъюнкты из $S$. Каждая литера считается столько раз, сколько она входит в дизъюнкты из $S$.*

**Пример 6.1.** $S = \{p, p \vee q, \bar{p} \vee \bar{q} \vee r\}, |S| = 3, \|S\| = 1 + 2 + 3 = 6.$

**Утверждение 6.1.** $\|S\| \geqslant |S|$.

Введем величину $k(S) = \|S\| - |S|, k(S) \geqslant 0$.

**Теорема 6.1.** *(Анрерсен) о полноте метода резолюций для исчисления высказываний.*
*Пусть $S$ - невыполнимое множество дизъюнктов. Тогда методом резолюций (резолютивним выводом) из $S$ можно получить $\emptyset$-дизъюнкт.*

*Доказательство* Индукция по $k(S) = \|S\| - |S|$.
Базис индукции $k(S) = 0 \Leftrightarrow$
1) Все дизъюнкты в $S$ однолитерные, т.е. $S = \{L_{i_1}, \dots, L_{i_n}\}$.
2) $S = \{L_{i_1} \vee L_{i_j}, L_{i_2}, \dots, L_{i_n}, \emptyset\}$.
Рассмотрим 1). В этом случае в $S$ есть контрарная пара $L_i = L, L_j = \bar{L}$ (иначе $S$ выполнимо) $\Rightarrow$ $\mathrm{Res}(L_i, L_j) = \mathrm{Res}(L, \bar{L}) = \emptyset$.
2) $S = \{L_{i_1} \vee L_{i_2}, L_{i_3}, \dots, L_{i_{n-1}}, L_n = \emptyset\}, k(S) = 0$. В этом случае $\emptyset$-дизъюнкт $\in S$ и получается за 0-шагов метода резолюций. $\emptyset$-дизъюнкт просто логическое следствие из множества формы $S$.
Предположение индукции Пусть теорема верна для всех $S$, у которых $k(S) < n$.
Индуктивный переход Докажем теорему для всех $S : k(S) = n$. Если $\emptyset$-дизъюнкт $\in S$, то очевидно, что теорема верна. Можно рассматривать такие $S$, что $\emptyset$-дизъюнкт $\notin S$. Так как $k(S) > 0$, то в $S$ существует хотя бы один дизъюнкт вида $D = (D^{'} \vee L)$, где $L$ - литера, и $D^{'} \neq \emptyset$-дизъюнкт. Рассмотрим множество $S^{'} = S \setminus \{D^{'} \vee L\}$ и образуем два множества дизъюнктов $S_1 = S^{'} \bigcup \{D^{'}\}, S_2 = S^{'} \bigcup L$. Очевидно, что $|S_1| = |S_2| = |S|$ - число дизъюнктов в $S_1$ и $S_2$ такое же, как в $S$, но $k(S_2) \leqslant k(S_1) < k(S)$, т.к. в $S_1$ на одну литеру меньше, а в $S_2$, по-крайней мере на одну литеру меньше. $\|S_1\| = \|S\| - 1, \|S_2\| = \|S\| = \|D^{'}\|$, и $\|D^{'}\| \geqslant 1$.
Докажем теперь, что $S_1$ и $S_2$ невыполнимые множества, если невыполнимо $S$(оно не выполнимо по условию теоремы).
Пусть $I$-множество всех интерпретаций $S$. (Это множество всех двоичных наборов $2^n$, где $n$-число различных переменных в дизъюнктах из $S$). $S \equiv 0$ на $I$ по условию теоремы.
$I_m \subset I$ - множество интерпраций на которых выполнима формула $D^{'} \vee L$. $I_M \neq \emptyset$ - очевидно. На $I_M$

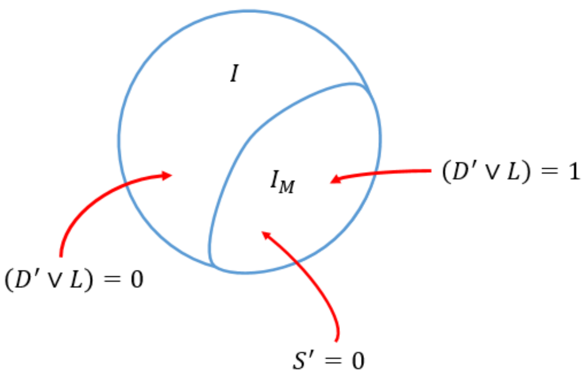



Рис. 17: К доказательству теоремы 6.1

множество формул $S^{'}$ не выполнимо(иначе бы было выполнимо $S$).
Следовательно и $S_1 = D^{'} \cup S^{'}, S_2 = L \cup S^{'}$ тоже не выполнимы на $I_M$.
На множестве $I \backslash I_M$ формула $(D^{'} \vee L) = 0 \Rightarrow D^{'} = 0, L = 0 \Rightarrow S_1 = S^{'} \cup D^{'}$ и $S_2 = L \cup D^{'}$ тоже не выполнимы на $I \backslash I_M$. Следовательно и $S_1$ и $S_2$ не выполнимы на $I$.

Отсюда(т.к. $K(S_1) < n, K(S_2) < n$ и оба они невыполнмы) по предположению индукциии из $S_1$ за $i$ шагов можно получить $\phi$-дизъюнкт, а из $S_2$ за $j$ шагов тоже можно получить $\phi$-дизъюнкт.
Рассмотрим два случая:
а) при выводе $\phi$-дизъюнкта из $S_1 = S^{'} \cup D^{'}$ дизъюнкт $D^{'}$ не использовался. Тогда этот вывод одновременно есть и вывод $\phi$-дизъюнкта из $S$.
б) дизъюнкт $D^{'}$ использовался. В этом случае вместо $D^{'}$ поставим исходный дизъюнкт $(D^{'} \vee L) = D$, тогда в этом выводе $\phi$-дизъюнка мы получим или $\phi$-дизъюнкт или дизъюнкт $L$.
$D^{'} \longrightarrow D^{''} \longrightarrow ... \longrightarrow D^{(i)} = \emptyset$.
$D^{'} \vee L \longrightarrow D^{''} \vee L \longrightarrow ... \longrightarrow D^{(i)} \vee L = \emptyset \vee L = L$.
Но по предположению индукуии из $S_2 = S^{'} \cup L$ за $j$ шагов можно получить $\phi$-дизъюнкт. Тогда из $S$ за $\sigma \le i + j$ тоже можно получить $\phi$-дизъюнкт. Теорема даказана.

Напомню, что $S = S^{'} \cup \{D^{'} \vee L\}, S_1 = S^{'} \cup \{D^{'}\}, S_2 = S^{'} \cup \{L\}$.
Заметим, что если из $S \supset \emptyset$, то $S$ не выполнимо. Для этого контрарную пару $3L, \bar{L}$ заменим на эквивалентные им формулы $L \vee 0, \bar{L} \vee 0 \Rightarrow Res(L \vee 0, \bar{L} \vee 0) = 0$(ложь), то есть 0 логическое следствие $S \Rightarrow (S \supset 0) \equiv 1$ то есть $S$ не выполнимо.
Можно выяснение истинности или ложности высказываний свести к решению систем алгебраических уравнений.
“Истина” - 1 $\quad \bar{p} \quad p \wedge q \quad p \vee q$.
“Ложь” - 0 $\quad 1 - p \quad p \cdot q \quad p + q - p \cdot q$.
$\Rightarrow p \supset q = \bar{p} \vee q = (1 - p) + q - (1 - p)q = 1 - p + p \cdot q$.
Но!!! Степень нелинейности возрастает $p \vee q \vee r = p + q - p \cdot q + r - (p + q - pq)r$.
Докажем, например, что $p \supset r$ является логическим следствием формул $(p \supset q)$ и $(q \supset r)$, т.е. если $(p \supset q) \wedge (q \supset r) = 1$, то и $(p \supset r) = 1$. Переписываем это в виде алгебраических уравнений.
$(p \supset q) = 1 \Rightarrow 1 - p + p \cdot q = 1, (q \supset r) = 1 \Rightarrow 1 - q + q \cdot r = 1$. Надо показать, что $1 - p + p \cdot r = 1$, если

$$\begin{cases} 1 - p + p \cdot q = 1 \Leftrightarrow p(1 - q) = 0 & (1) \\ 1 - q + q \cdot r = 1 \Leftrightarrow q(1 - r) = 0 & (2) \end{cases}$$

Варианты:
1.1) $p = 0, q = 0 \rightarrow r = 0$ или $r = 1$.
1.2) $p = 0, q = 1 \rightarrow r = 1$.
2) $p = 1, q = 1\, (1) \rightarrow$ из $(2) r = 0$.
Итак, при $p = 0, 1 - p + pr = 1$,
при $p = 1$ из $(1) \Rightarrow q = 1$, из $(2) \Rightarrow r = 1 \Rightarrow$,
Надо проверить наборы

$$\begin{array}{ccc l} p & q & r & \\ & & & \\ \left.\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right\} & & & 1 - p + pr = 1 \Leftrightarrow 1 = 1, \text{ т.к. } p = 0. \\ 1 \quad 1 \quad 1 & & & 1 - 1 + 1 \cdot 1 = 1, \text{ т.е. } (p \supset r) - \text{ логическое следствие} \end{array}$$

**Пример 6.2.** *Пример-задача*
*База данных(БД)=*$\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$.
*База знаний(БЗ)=*$\{(\alpha_1 \wedge \alpha_2) \supset (\alpha_5 \wedge \alpha_6), (\alpha_4 \wedge \alpha_3) \supset (\alpha_7), (\alpha_7 \wedge \alpha_6) \supset \alpha_8\}$.
*БД - это факты из ПО (аксиомы ПО).*
*БД - это знания(что делать) о фактах ПО.*
*Система извлечения знаний: БД*$\cup$*БЗ.*
*Вопрос к системе: можно ли получить* $\alpha_8$*?*

Сводим вопрос к логическому следствию
$(((\alpha_1 \wedge \alpha_2) \supset (\alpha_5 \wedge \alpha_6)) \wedge ((\alpha_4 \wedge \alpha_3) \supset \alpha_7) \wedge ((\alpha_7 \wedge \alpha_6) \supset \alpha_8)) \wedge \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4) \supset \alpha_8$, то есть является ли $\alpha_8$ логическим следствием формул из БД$\cup$БЗ?
Преобразуем формулы к дизъюнктам $((\bar{\alpha_1} \vee \bar{\alpha_2} \vee \alpha_5) \wedge (\bar{\alpha_1} \vee \bar{\alpha_2} \vee \alpha_6) \wedge (\bar{\alpha_3} \vee \bar{\alpha_4} \vee \alpha_7) \wedge (\bar{\alpha_4} \vee \bar{\alpha_3} \vee \alpha_7) \wedge \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4 \wedge \bar{\alpha_8}) \equiv 0$ то есть надо доказать невыполнимость этой формулы.

$$\underbrace{\bar{\alpha_1} \vee \bar{\alpha_2} \vee \alpha_5}_{\text{этот дизъюнкт не потребовался}}, \underbrace{\bar{\alpha_1} \vee \bar{\alpha_2} \vee \alpha_6}_{\textcircled{1}}, \underbrace{\bar{\alpha_4} \vee \bar{\alpha_3} \vee \alpha_7}_{\textcircled{2}}, \underbrace{\bar{\alpha_7} \vee \bar{\alpha_6} \vee \alpha_8}_{\textcircled{3}}, \underbrace{\alpha_1}_{\textcircled{4}}, \underbrace{\alpha_2}_{\textcircled{5}}, \underbrace{\alpha_3}_{\textcircled{6}}, \underbrace{\alpha_4}_{\textcircled{7}}, \underbrace{\bar{\alpha_8}}_{\textcircled{8}}$$

$① + ⑤ = \underbrace{\bar{\alpha_1} \vee \alpha_6}_{⑨}$; $② + ⑥ = \underbrace{\bar{\alpha_4} \vee \alpha_7}_{⑩}$; $⑨ + ④ = \underbrace{\alpha_6}_{⑪}$; $⑩ + ⑦ = \underbrace{\alpha_7}_{⑫}$;

$⑪ + ③ = \underbrace{\bar{\alpha_7} \vee \alpha_8}_{⑬}$; $⑬ + ⑫ = \underbrace{\alpha_8}_{⑭}$; $⑭ + ⑧ = \emptyset$

Ответ: Да, $\alpha_8$ можно получить.
Переменные $\alpha_1 \dots \alpha_8$ могут быть любой природы - это могут быть химические формулы, технологические процессы и т.д.

**Замечание 6.1.** *Пусть $S_1$ - невыполнимое множество формул, а $S_2$ - любое множество формул. Тогда множество $S = S_1 \cup S_2$ тоже невыполнимо.*

# 7 Лекция 7

Метод резолюции и его вариант с использованием интерпретации невыполнимого множества $S$ обладают свойством полноты, т.е. $\emptyset$-дизъюнкт $\in [S]_{\text{Res}}$. В большинстве задач невыполнимое множество формул $S$ можно представить в виде $S = A \cup \Phi$, где $A = \{A_i\}$ - аксиомы предметной области, а $\Phi = \{\Phi_j\}$ - множество всех формул, которые мы хотим доказать, т.е. $\forall \Phi_j \in \Phi \Rightarrow (A \wedge \bar{\Phi}_i) \equiv 0$.

$\overbrace{\underbrace{A_1, \dots, A_n}_{A} \quad \underbrace{\bar{\Phi}_1, \dots, \bar{\Phi}_m}_{\bar{\Phi}}}^{S}.$

Если $A$ - это множество аксиом ПО, то оно должно быть непротиворечивым и при невыполнимом $S$ $\emptyset$-дизъюнкт может быть получен $\text{Res}(B_1, B_2)$, где один из дизъюнктов $B_1$ и $B_2$ принадлежит $\bar{\Phi}$. Т.е. получение резольвент выглядит так:

$A \quad \underbrace{\Phi \cup \{ \text{ все резольвенты из } S \text{ вида } \text{Res}(B_1, B_2)\}}_{\text{несущее множество } T}.$

Из них образуется $\text{Res}(B_1^{'}, B_2^{'})$; если $B_1^{'} \in A$, то $B_2^{'} \in T$ и наоборот. Внутри $A$ Res запрещено по определению.
Зачем эта процедура нужна? Цель - уменьшит число резольвент. Как правило $|A| >> |\Phi|$.

**Определение 7.1.** *Пусть $S$ - невыполнимое множество назовем $S_H \subseteq S$* ***наименьшим невыпольнимым множеством****, если:*
*1) $S_H$ - невыполнимо*
*2) $\forall S^* \subset S_H$, множество $S^*$ - выполнимо.*

Заметим, что если $S_H \subseteq S$ наименьшее невыполнимое множество, то $\emptyset$-дизъюнкт можно искать так: $\forall c \in S_H \quad \underbrace{S_H \setminus \{c\}}_{\text{множество } A} \quad \underbrace{\{\bar{c}\} \cup \{\text{все резольвенты } \text{Res}(B_1, B_2)\}}_{\text{несущее множество } T}.$

Из них получить Res. $S_H \setminus \{c\}$ - выполнимо по определению $S_H$, т.е. это аналог аксиом ПО. Искать $S_H$ - трудно.

**Пример 7.1.** *(Самостоятельно)*
$S = \{p \vee q, \bar{p} \vee \bar{r}, p \vee q \vee r, \bar{p} \vee \bar{q}, \bar{q} \vee \bar{r}, p, q, r\}$ *Найти $S_H$, если их несколько, найти все $S_H \supseteq S$.*

***Линейная резолюция***

**Определение 7.2.** *Для заданного множества дизъюнктов $S$ и дизъюнктов $C_0 \in S$* ***линейный вывод (линейная резолюция)*** *дизъюнкта $S_n \in S$ с верхним дизъюнктов $C_0$ - это вывод, имеющий следующий вид: (см. рис 7.1). При этом*
*1. $\forall i = 0, 1, \dots, n-1$ дизъюнкт $C_{i+1} = \text{Res}(C_i, B_i)$.*
*2. $\forall B_i$ либо $\in S$, либо есть $C_j$ для $i < j$.*

**Определение 7.3.** *$C_i$ называется* ***центральным дизъюнтком****. $B_i$ называется* ***боковым дизъюнтком****.*

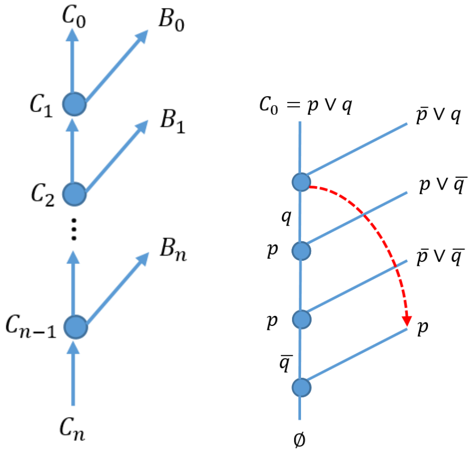



Рис. 18: Линейная резолюция / К примеру 7.2

**Пример 7.2.** $S = \{p \vee q, \bar{p} \vee q, p \vee \bar{q}, \bar{p} \vee \bar{q}\}$

Можно доказать следующее

**Утверждение 7.1.** *Если $S_H \in S$ - наименьшее невыполнимое множество дизъюнктов, то существует линейный вывод $\emptyset$-дизъюнкт с верхним дизъюнктком $C \in S_H$.*

Важно Утверждение не гарантирует, что $\forall C \in S_H$ можно получить $\emptyset$-дизъюнкт. Пусть $S_H = \{C_1, \ldots, C_n\}$ Выводы $\emptyset$-дизъюнкта могут быть разной длины.

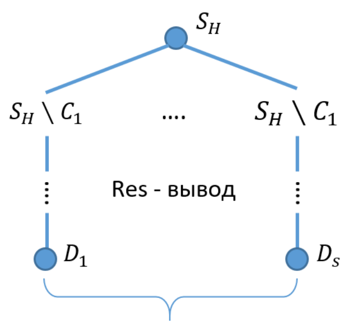



Рис. 19: Важно отметить!

$\emptyset$ есть среди $\{D_1, \ldots, D_s\}$, но какие $D_i = \emptyset$-дизъюнкт мы не знаем.(перебор!) Все стратегии получения $\emptyset$-дизъюнкта, которые мы рассматривали до этого обладали свойством полноты, т.е. гарантировали получение $\emptyset$-дизъюнкта из $S$ при выпольнимости его.
**Эвристики** - стратегии не обладающие свойством полноты, но в отдельных случаях позволяющие быстро получить результат. Например, те, которые мы уже раньше сформултровали:
1) Удаление дублей: $D \vee D = D$.
2) Удаление тавтологий $D \equiv 1$.
3) Упорядочение букв в дизъюнктах $P_1 \geqslant P_2 \geqslant \ldots \geqslant P_n$ и резольвирование букв по старшинству.
Можно добавить еще одино правило:
4) Если в $S$ есть $D = D_1 D_2$ и есть $D_1$, то $D$ можно удалить $S$. Это не влияет на невыполнимость: $D_1 \vee D_1 D_2 = D_1(1 \vee D_2) = D_1$.

Правило формулируется так: "короткий" дизъюнкт поглощает "длинный" дизъюнкт.
Ещё можно предложить такую стратегию. Вывод выглядит так: $S = \{S_1, \ldots, S_n\} \cup \{S_0\}$, где $\{S_0\}$ - выделенный дизъюнкт. Все боковые дизъюнкты из $S$ и верхний дизъюнкт $S_0$.
Это частный случай линейной резолюции. В линейной резолюции разрешается использовать $S^{(1)}, \ldots, S^{(n)}$.

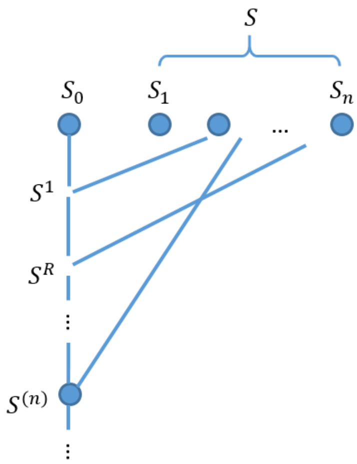



Рис. 20: S-резолюция

Назовем этот процесс $S-$**резолюцией (или входной резолюцией)**.

**E-резолюция(единичная резолюция)**

**Определение 7.4.** *В* $E$***-резолюции*** *по-крайней мере один из дизъюнктов - литерал (единичный дизъюнкт), т.е.* $\forall$ *шаг вывода есть* $\mathrm{Res}(A_i, A_j)$*, где* $A_i$ *или* $A_j$ *(или оба вместе) есть* $L$ *или* $\bar{L}$*.*

**Определение 7.5.** ***S-опровержение и E-опровержение*** *- это вывод из* $S$ $\emptyset$*-дизъюнкта* $E-$*(соответственно* $S-$*) резолюцией. Эти процедуры не обладают свойством полноты.*

**Теорема 7.1.** *Для невыполнимого множества дизъюнктов* $S$ *существует* $E$*-опровержение* $\Leftrightarrow$ *существует* $S-$*опровержение.*

**Использование оценочных функций** Пусть есть резолютивный вывод начинающийся с верх-

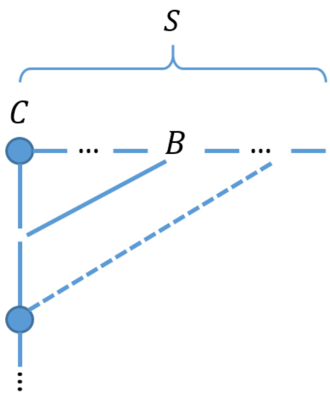



Рис. 21: Использование оценочных функций

него дизъюнкта $C$ и бокового дизъюнкта $B$, где $C, B \in$ некоторому множеству дизъюнктов $S$. Обозначим через $L^*(C, B)$ **наименьшее число шагов** метода Res для получения $\emptyset$-дизъюнкта из этой пары $(C, B)$. Обычно можно построить некоторые примеры для некоторых пар $(C_1, B_1), \ldots, (C_n, B_n) \rightarrow L^*(C_1, B_1), \ldots, L^*(C_n, B_n)$ - примеры просто подсчитаны (например в ручную).

Введем ещё понятие **характеристической функции** пары $(C, B)$
1) $f_1(C, B) =$ число литер в $C$ (или в $B$).
2) $f_2(C, B) =$ число боковых дизъюнктов, которые можно резольвировать с $C$.
3) $f_3(C, B) =$ длина $C$ + длина $B$ - 2.

...
Эти функции легко вычислить. Можно придумать много таких полезных характеристических функций пар. Рассмотрим функцию пар следующего вида:

$$L(C,B) = W_0 + W_1 \cdot f_1(C,B) + \ldots + W_n \cdot f_n(C,B).$$

где $f_1, \ldots, f_n$ - характеристические функции пар.

<u>Гипотеза:</u> Функцию $L^*$ можно "хорошо" аппроксимировать линейной комбинацией характеристических функций $f_1, \ldots, f_n$, например, методом наименьших квадратов.
Рассмотрим

$$\begin{aligned}
&S(W_0, W_1, \ldots, W_n) = \\
&= ((L^*(C_1,B_1) - (W_0 + W_1 \cdot f_1(C_1,B_1) + \ldots + W_n \cdot f_n(C_1,B_1))))^2 + \ldots \\
&+ ((L^*(C_n,B_n) - (W_0 + W_1 \cdot f_1(C_n,B_n) + \ldots + W_n \cdot f_n(C_n,B_n))))^2
\end{aligned}$$

Для того, чтобы найти минимум $S(W_0, W_1, \ldots, W_n)$, надо её продифференцировать по $W_0, W_1, \ldots, W_n$ и решить систему уравнений

$$\frac{\partial S}{\partial W_0} = 0, \ldots, \frac{\partial S}{\partial W_n} = 0, \ldots$$

То есть

$$\begin{cases}
2 \cdot (L^*(C_1,B_1) - (W_0 + W_1 \cdot f_1(C_1,B_1) + \ldots + W_n \cdot f_n(C_1,B_1))) \cdot 1 = 0 \\
2 \cdot (L^*(C_2,B_2) - (W_0 + W_1 \cdot f_1(C_2,B_2) + \ldots + W_n \cdot f_n(C_2,B_2))) \cdot f_1(C_2,B_2) = 0 \\
\ldots \\
2 \cdot (L^*(C_n,B_n) - (W_0 + W_1 \cdot f_1(C_n,B_n) + \ldots + W_n \cdot f_n(C_n,B_n))) \cdot f_n(C_n,B_n) = 0
\end{cases}$$

Это линейная относительно $W_0, W_1, \ldots, W_n$ система уравнений $(n+1) \times (n+1)$ и её легко можно решить методом Гаусса (например).
Результаты будет "хорошим", если функция $L$ (линейная комбинация $f_1, \ldots, f_n$) будет хорошо приближать $L^*$ на других дизъюнктах.

# 8 Лекция 8

**Исчисление предикатов** - логика первого порядка (фрагмент).

**Пример 8.1.** *Рассмотрим следующие предложения:*
*1. "Every man is mortal"* $A_1$
*2. "Peter is a man"* $A_2$
*3. "Peter is mortal"* $A_3$
*Кажется, что из* $A_1 \wedge A_2$ *следует* $A_3$, *но* $(A_1 \wedge A_2 \supset A_3) \not\equiv 1$.

Причина в том, что исчисление высказываний не учитывает структуру предложения. Необходимо в примере, приведенном выше, учитывать структуру предложения.

**Определение 8.1.** ***Предикат*** - *любое выражение, имеющее форму высказывания, содержащее переменные величины (предметные переменные).*

При придании значений всем предметным переменным в этом выражении, оно превращается в высказывание (истинное или ложное).

**Пример 8.2.** $P(x) = x$ *- есть человек.* $P(\text{Джон}) = 1, P(\text{собака}) = 0$. $Q(x) =$ *"x смертен"("x is mortal").* *Тогда* $A_1, A_2, A_3$ *записывается так:*

$$(\forall x(P(x) \supset Q(x)) \wedge P(Peter)) \supset Q(Peter)$$

Здесь и далее будут использоваться знаки (кванторы) $\forall$ - "для любого" и $\exists$ - "существует".

**Символика и язык исчисления предикатов (логики первого порядка)**
$G = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$.

- $G_1 = \{x, y, z, \ldots\}$ - **латинские буквы** (возможно с нижними индексами - предметные (индивидуальные) переменные);
- $G_2 = \{P_j^{n_i}\}, i, j \in I \subseteq \mathbb{N}$ - **предикатные символы** (большие латинские буквы, возможно с верхними или нижними индексами или без них);
- $G_3 = \{f_k^{n_m}, m, k \in I, \subseteq \mathbb{N}\}$ - **функциональные символы** (с индексами или без, верхними индексами)
- $G_4 = \{a_l\}, l \in K \subseteq \mathbb{N}$ - **предметные** (индивидуальные) **константы**
- $G_5 = \{-, \wedge, \vee, \supset, \forall, \exists\}$ - **логические символы** (операций, действий).
- $G_6 = \{$“,”,“(”,“)”$\}$ - **вспомогательные символы**

$P_j^m, f_k^n$ - в этих символах их $G_2$ и $G_3$ нижний индекс есть *номер* символа, а верхний символ - *арность* символа, т.е. число переменных (и/или констант) от которых он зависит.

**Пример 8.3.** *$P_1^2$ - это предикат вида $P_1(x_{i_1}, y_{i_2})$; $f_2^3$ - функциональный символ вида $f_2(x, y, z)$. Иногда верхние и нижние индексы мы будем не писать*

**Определение 8.2.** ***Темры***

1. *Предметные переменные и константы есть термы*
2. *Если $f^n$ есть n-арный функциональный символ, а $t_1, \ldots t_n$ - термы, то $f(t_1, \ldots, t_n)$ - терм*
3. *Других термов нет*

**Определение 8.3.** *Если $P$ - n-арный предикатный символ и $t_1, \ldots, t_n$ - термы, то $P(t_1, \ldots, t_n)$ - **атомарные формулы (атомы)**.*

**Определение 8.4.** ***Формулы (правильно построенные выражения)***

1. *Любой атом есть формула*
2. *Если $A$ и $B$ - формулы, то $\bar{A}, (A \wedge B), (A \vee B), (A \supset B)$ - формулы*
3. *Если $A$ - формула, $x$ - предметная переменная, то $\forall x(A)$ и $\exists x(A)$ - формулы*
4. *Других формул нет*

Выражения $\forall x(A), \exists x(A)$ - круглые скобки обозначают область действия кванторов, т.е. $\forall x(A)$ означает, что $\forall$ действует на $x$ входящих в формулу $A$ свободно, т.е. это те вхождения $x$, которое не лежат в области действия других кванторов “$\forall$” и “$\exists$”, которые возможно входят в формулу $A$.
В противном случае вхождение переменной $x$ называется **связанным**.

**Пример 8.4.**

$$\underbrace{A(x, y)}_{\text{свободные вхотдения } x \text{ и } y} = x \geq y.$$

$$\forall x(A(\underbrace{x}_{\text{связанное}}, \underbrace{y}_{\text{свободное}}) = \forall x(x \geq y).$$

$$\forall x \exists y(\underbrace{A(x, y)}_{\text{оба связанные}}) = \forall x \exists y(x \geq y).$$

Правильно было бы записать $\forall x(\exists y(A(x, y)))$, чтобы отметить скобками области действия кванторов. Одна и таже переменная может входить и связанно в формулу $A$ и свободно.

**Пример 8.5.** *$\forall x(B(x))$ - область действие квантора $\forall$.*

$$\underbrace{\forall x(B(x))}_{\text{связанное вхождения } x} \supset \underbrace{C(x)}_{\text{свободное вхождения } x} = A(x).$$

**Вопрос 8.1.** *1.$\forall x(A(x) \wedge (\exists x, B(x, x)))$.*
*2. $\forall x, \exists x, A(x, x)$.*
*Правильное ли записаны формулы?*

Если в формуле $A(x, x_1, ..., x_5)$ переменная $x$ - связанная, то её можно "переименовать", т.е. заменить любой другой переменной, не входящей в список переменных из $A$.
$A(t, x_1, ..., x_5), t \notin \{x_1, ..., x_5\}$.

**Пример 8.6.** $\forall x(A(x)) \supset B(y) \Leftrightarrow \forall t(A(t)) \supset B(y)$.
*Переименовывать можно только связанные вхождения переменной свободные вхождения из переименовываются.*
$\forall \underbrace{x(A(x))}_{\text{связанное}} \supset \underbrace{B(y)}_{\text{свободное}} \Leftrightarrow \forall t(B(t)) \supset B(x)$.

**Определение 8.5.** *Формула $A$ в которой вхождения <u>всех её</u> переменных <u>связанные</u> называется* ***замкнутой формулой****.*

**Пример 8.7.** $P(x, y, z) = (x + y - z = 0)$ - *атом(атомарная формула).*
$\forall x, \forall y, \exists z(P(x, y, z))$ - *замкнутая формула(смысл: для любых $x$ и $y$, существует $z = x + y$).*

**Замечание 8.1.** *В формулах $A(x)$ и $B(x)$ которые мы рассматриваем выше, предполагается, что связанного значения $x$ не встречается.*

**<u>Интерпретация формул</u> исчисления предикатов.(первого порядка)**
Пусть имеется предметная область $D \neq \emptyset$.
1. $\forall$ предметной константе $\alpha_l$ сопоставим элемент из $D$.
2. $\forall$ функциональному систему $f_i^{n_i}$ отоборажение $f_i^{n_i} : \underbrace{D \times ... \times D}_{n_i} \to D$.
3. $\forall$ предникатному систему $P_j^k$ отображение $P_j^k : \underbrace{D \times ... \times}_{k} \to \{0, 1\}$; 0 - ложь, 1 - истина.

**Определение 8.6.** *Формула исчисления предикатов 1-го порядка называется:*

- ***общезначимой****, если она истинна во <u>всех</u> интерпретациях*
- ***не выполнимой****, если она <u>ложна</u> во всех интерпретациях.*
- ***выполнима*** *в остальных случаях.*

Покажем, что следствие в примере 1(см. начало лекции) верно. Итак
$A_1$: $\forall x(man(x) \supset mortal(x))$
$A_2$: $man(Peter)$
Надо показать, что mortal(Peter) есть логическое следствие $A_1$ и $A_2$. То есть, если в интерпретации $D$ истинны $A_1$ и $A_2$, то mortal(Peter) тоже истинна в $D$.
Имеем: $A_1 \wedge A_2$ истинны в $D \Leftrightarrow A_1$ истинна и $A_2$ истинна т.к. $man(x) \supset mortal(x)$ истина, $\forall x \Rightarrow man(Peter) \supset mortal(Peter)$ тоже истинна $\Rightarrow \overline{man(Peter)} \vee mortal(Peter)$ - истинна, но $\overline{man(Peter)}$ ложна(т.к. истинна $A_2 : man(Peter)$) $\Rightarrow$ истинна $mortal(Peter)$ в $D$, если в $D$ истинна $A_1 \wedge A_2$.

# 9 Лекция 9

**Определение 9.1.** *Формулы $\Phi_1$ и $\Phi_2$ сигиатуры $\Sigma$ называются* ***эквивалентиыми****, если их значения совпадают на $\forall$ интерпретации $I_j$.*

<u>Важно</u> все эквивалентности, введенные нами для булевских формул(дистрибутивность, ассоциативность, правили Маргана, правило подстановка...) остаются справедливыми для формул логики предикатов. Правда добавляются ещё эквивалентности, связанные с кванторами $\exists$ и $\forall$:
1) $\neg\forall\ x\Phi(x) \equiv \exists x(\neg\Phi(x))$ или $\overline{\forall x\Phi(x)} \equiv \exists x(\bar{\Phi}(x))$. $\neg$ - знак отрицания.
2)$\overline{\exists x\Phi(x)} \equiv \forall x(\bar{\Phi}(x))$.
3)$(\Psi \wedge \forall x\Phi(x)) \equiv \forall x(\Psi \wedge \Phi(x))$.
4)$(\Psi \wedge \exists x\Phi(x)) \equiv \exists x(\Psi \wedge \Phi(x))$.
5)$(\Psi \vee \forall x\Phi(x)) \equiv \forall x(\Psi \vee \Phi(x))$.
6)$(\Psi \vee \exists x\Phi(x)) \equiv \exists x(\Psi \vee \Phi(x))$.
7)$(\Psi \supset \forall x\Phi(x)) \equiv \forall x(\Psi \supset \Phi(x))$.
8)$(\Psi \supset \exists x\Phi(x)) \equiv \exists x(\Psi \supset \Phi(x))$.
9)$(\forall x\Phi(x) \supset \Psi) \equiv (\exists x(\Phi(x) \supset \Psi))$.
10)$(\exists x\Phi(x) \supset \Psi) \equiv (\forall x(\Phi(x) \supset \Psi))$.

**Замечание 9.1.** *При этом $\Psi$ не должна содержать свободных вхожденний переменной $x$.*

Докажем тождество 9):
$(\forall x\Phi(x)\supset\Psi)\equiv((\overline{\forall x\Phi(x)})\vee\Psi\equiv(\overline{(\exists x\Phi(x))}\vee\Phi)\equiv\exists x(\bar{\Phi}(x)\vee\Psi)\equiv\exists x(\Phi(x)\supset\Psi)$;
Использовали тождества 1) и 4). Тождество 10) доказывается аналогично.
Если $x$ может входить свободно в $\Psi(x)$ то тождества 1)-10) неверны.

**Пример 9.1.** $\Psi(x)=(x=0)$, $\Phi(x)=(x=x)$ $\Psi(x)\wedge\forall x\Phi(x)=(x=0)\wedge(\forall x(x=x))$ *(*)*
$\forall x(\Psi(x)\wedge\Phi(x))=\forall x(\underbrace{(x=0)}_{\forall x(x=0)\equiv \text{ложно}})\wedge(x=x))$ *(**)*

Пусть $\spadesuit_i\in\{\forall,\exists\}$.
Если в формуле $\varphi(x)$ все свободные вхождения $x$ заменить на $y$, (переменная, отличная от других переменных, которые быть может входят в $\varphi(x)$) то $\spadesuit x\varphi(x)\equiv\spadesuit y\varphi(y)$ - замена тогда в пример 9.1 можно получить так:
$\Psi(x)\wedge(\forall x\Phi(x))\equiv\Psi(x)\wedge(\forall y\Phi(y))\equiv\forall y(\Psi(x)\wedge\Phi(y))$.
Дальше, основывалсь на известных нам тождествах, надо научиться приводить любую формулу ИП к следующему виду:
$\spadesuit_1x_1\spadesuit_2x_2...\spadesuit_nx_n\Phi$, где $\Phi$ - формула, не содержащая кванторов $\exists$ или $\forall$. Формула такого вида называется **предваренной нормальной формой**.
В курсе математической логики доказывается, что для $\forall$ формулы ИП существует эквивалентная ей предваренная нормальная форма.
Покажем это на пример.

**Пример 9.2.** *Пусть дана формула $\exists yP(x,y)\wedge\neg(\forall xP(x,y)\supset\exists xQ(x,z))$.*
*Переименуем связанные переменные*
$\exists y_1P(x,y_1)\vee\neg(\forall x_1P(x_1,y)\supset\exists x_2Q(x_2,z))\equiv$
$\equiv\exists y_1P(x,y_1)\vee\neg(\overline{\forall x_1P(x_1,y)}\vee\exists x_2Q(x_2,z))\equiv$
$\equiv\exists y_1P(x,y_1)\vee\neg(\exists x_1\overline{P(x_1,y)}\vee\exists x_2Q(x_2,z))\equiv$
$\equiv\exists y_1P(x,y_1)\vee\neg(\exists x_1\exists x_2\overline{P(x_1,y)}\vee Q(x_2,z))\equiv$
$\equiv\exists y_1P(x,y_1)\vee(\forall x_1\forall x_2\neg(P(x_1,y)\supset Q(x_2,z)))\equiv$
$\equiv\underbrace{\underbrace{\exists y_1\forall x_1\forall x_2}_{\text{кванторная приставка}}\quad\underbrace{(P(x,y_1)\wedge\overline{(P(x_1,y)\supset Q(x_2,z))})}_{\text{бескванторная формула}}}_{\text{предваренная нормальная форма}}.$
*Дальше из кванторной приставки можно специальным образом удалить кванторы существования. Заметим, что бескванторную часть формулы мы(используя известные нам тождества для булевских формул) можем привести к конъноиктивной нормальной форме(КНФ).*
$P(x,y_1)\wedge\overline{(P(x_1,y)\supset Q(x_2,z))}=P(x,y_1)\wedge\overline{(\overline{P(x_1,y)}\vee Q(x_2,z))}=\underbrace{P(x,y_1)\wedge P(x_1,y)\wedge\overline{Q(x_2,z)}}_{\text{КНФ}}.$

Итак, пусть у нас есть формула в предваренной нормальной форме.
$\Phi=\spadesuit_1x_1\spadesuit_2x_2...\spadesuit_nx_n\ M$, где $M$ - бекванторная формула в конъюнктивной нормальнной форме.
$\spadesuit_i\in\{\forall,\exists\}$ $i=1\dots n$.
Пусть $\spadesuit_r$ есть $\exists r, 1\le r\le n$, если левее этого квантора нет им одного квантора $\forall$, то возьмем константу "С"(не встречающуюся в $M$), заменим $x_r$ в $M$ на "С" и из кванторной приставки вычеркнем $\exists_rx_r$.
Если левее $\exists x_r$ стоят кванюры всеобщности $\forall_{s_1}x_{s_1}...\forall_{s_m}x_{s_m}, 1\le s_1<s_2<...<s_m<r$, то заменим все $x_r$ в $M$ на $f(x_{s_1},...,x_{s_m})$, где $f$-функциональный символ, не встречающийся в $M$ и вычеркием $\exists_rx_r$ из кванторной приставки. Проделаем эту процедуру для всех кванторов $\exists_i$ встречающихся в кванторной приставке. В итоге получим формулу $\underbrace{\forall_{i_1}x_{i_1}...\forall_{i_t}x_{i_t}}_{\text{только кванторы}}\quad\underbrace{M'}_{\text{бескванторная формула}}$ - это (скулеловская) стандартная форма формулы $\Phi=\spadesuit_1x_1,...\spadesuit_nx_n\quad M$.

**Пример 9.3.**

$$\Phi=\exists x_1\forall x_2\forall x_3\exists x_4\forall x_5\exists x_6\quad M(x_1,x_2,x_3,x_4,x_5,x_6).$$

*1) левее $\exists x_1$ нет кванторов $\forall$, $x_1:=c$*
*2) левее $\exists x_4$ стоят $\forall x_2,\forall x_3$, $x_4:=f(x_2,x_3)$*

*3) левее $\exists x_6$ стоят $\forall x_2, \forall x_3, \forall x_5$, $x_6 := g(x_2, x_3, x_5)$.*
*В итоге получаем стандартную форму для $\Phi$:*

$$\forall x_2 \forall x_3 \forall x_5 \quad M(c, x_2, x_3, f(x_2, x_3), x_5, g(x_2, x_3, x_5)) = \forall x_2 \forall x_3 \forall x_5 \quad M'(x_2, x_3, x_5)$$

Далее можно кванторы $\forall$ опустить и считать, что в любой интерпретации $I$ переменные $(x_2, x_3, x_5)$ принимают любое допустимое значение из $I$. (т.е. управляющая кванторами $\forall$). Итак, мы проделали следующие преобразования:
Исходная формула ИП $\Phi \Rightarrow \Phi'$ - **предваренная нормальная форма** формулы $\Phi$. $\Rightarrow \Phi^{''}$ - **стандартная форма** $\Phi' \Rightarrow \Phi^{'''} = M'$ - **бескванторная формула** в конъюнктивной нормальной форме (т.е. конъюнкция дизъюнктов $D_1 \wedge \ldots \wedge D_s$). Только дизъюнкты состоят из формул ИП, не содержащих кванторов $\exists$ и $\forall$.

**Пример 9.4.** $P(x, f(x)) \vee Q(y, g(x, y))$ - *пример дизъюнкта*

Далее можно использовать те стратегии и определения, которые мы вводили для исчисления высказываний (контрарные пары литер, стратегии вывода и.т.д.). Однако есть одна тонкость - поясним её на примере

**Пример 9.5.** *Рассмотрим два дизъюнка $D_1 = P(x) \wedge Q(x)$ и $D_2 = \overline{P(f(x))} \vee R(x)$. Формально здесь нет контрарных пар. Однако если мы сделаем подстановку в $D_1$ вместо $x$ подставим $f(a)$, а в $D_2$ вместо $x$ подставим $a$, то $D_1^{'} = P(f(a)) \vee Q(f(a)), D_2^{'} = \overline{P(f(a))} \vee R(a)$. Тогда $\mathrm{Res}(D_1^{'}, D_2^{'}) = \underline{Q(f(a))} \vee R(a)$. Можно было бы в $D_1$ вместо $x$ подставить $f(x)$: $D_1^{'} = P(f(x)) \vee Q(f(x)), D_2 = \overline{P(f(x))} \vee R(x) \Rightarrow \mathrm{Res}(D_1^{'}, D_2) = Q(f(x)) \vee R(x)$ - другая резольвента.*

Говорят, что дизъюнкт $Q(f(a)) \vee R(a)$ есть **частный случай**(пример) дизъюнкта $Q(f(x)) \vee f(x)$.

**Определение 9.2.** *Пусть $v_1, \ldots, v_n$ - различные переменные и $t_1, \ldots, t_n$ - термы, отличные от переменных $v_1, \ldots, v_n$. Множество $\{t_1|v_1, \ldots, t_n|v_n\}$ называется **подстановкой** (вместо $v_i$ подставляется терм $t_i$), $i = 1, \ldots, n$.*

**Пример 9.6.** $\{y|z, f(z)|x, a|w, f(g(a))|u\}$.

Пусть $\tau = \{t_1|v_1, \ldots, t_n|v_n\}$ - подстановка и $E$ - формула (выражения) ИП. Тогда применение $\tau$ к $E$(запись $E\tau$) - это формула $E^{'}$ получения из $E$ заменой всех вхождений переменной $v_i (1 \leqslant i \leqslant n)$ на терм $t_i$.

**Пример 9.7.**

$$E = P(x, y, z), \tau = \{a|x, f(b)|y, c|z\} \quad E^{'} = E\tau = P(a, f(b), c)$$

**Определение 9.3.** *Подстановка $\tau$ является **унификатором** для множества формул (выражений). $\mathcal{E} = \{E_1, \ldots, E_k\}$, если $E_1\tau = E_2\tau = \ldots = E_k\tau$. В этом случае говорят, что множество $\mathcal{E}$ **унифицируемо**.*

**Пример 9.8.** $\mathcal{E} = \{\overbrace{P(a, y)}^{=E_1}, \overbrace{P(x, f(b))}^{=E_2}\}$ *унифицируемо (подстановкой $\tau = \{a|x, f(b)|y\}$.) $E_1^{'} = E_1\tau = P(a, f(b)), E_2^{'} = E_2\tau = P(a, f(b))$. $E_1^{'} = E_2^{'} \Rightarrow \tau$ унификатор для $\mathcal{E} = \{E_1, E_2\}$.*

Существует несложный алгоритм унификации, который для множества выражений $\{E_1, \ldots, E_n\}$ либо выдает унификатор, либо сообщает, что это множесто выражений не унифицируемо. Поскольку наша задача есть рассмотрение <u>приложений</u> математической логики, мы не будем далее углубляться в обоснование сказанного. Я отошлю интересующихся деталями к любому стандартному курсу математической логики (например к книге Мендельсок “Математическая логика”).
<u>Главное</u>, для вывода логических следствий, выяснения невыполнимости множества формул языка предикатов. Надо каждую формулу $\Phi_i$ из $S = \{\Phi_1, \ldots, \Phi_n\}$ сначала:
$\Phi_i \Rightarrow \Phi_i^{'}$(стандартной нормальной форме) $\Rightarrow$
$\Phi_i^{''}$(сколемовская нормальная форма) $\Rightarrow$
убрать все кванторы $\forall$ и получить бескванторную формулу $\Phi_i^{'''}$, которую (используя булевским эквивалентности) преобразовать к виду КНФ $D_1 \wedge \ldots \wedge D_n$, где $D_i$ - дизъюнкты $(i = 1, \ldots, n) \Rightarrow$ используя аналоги стратегии получения $\emptyset$-дизъюнкта для ИВ и используя при необходимости алгоритм унификации получить методом Res $\emptyset$-дизъюнкт из $\{D_1, \ldots, D_n\} = D$.

Ещё раз: есть $\Phi_i$ - формула ИП. $\Phi \Rightarrow \spadesuit_1 x_1 \spadesuit_2 x_2 \dots \spadesuit_n x_n \quad M(x_1, \dots, x_n) \Rightarrow \forall_1 x_{i_1} \dots \forall_s x_{i_s} \underbrace{M'(x_{i_1}, \dots, x_{i_s})}_{\text{Бескванторная формула}} \Rightarrow$
$\Rightarrow M'(x_{i_1}, \dots, x_{i_s}) \Rightarrow D_1 \wedge \dots \wedge D_n$ (от тех же переменных $x_{i_1}, \dots, x_{i_s}$) $\Rightarrow$ образовать $D = \{D_1, \dots, D_n\} \Rightarrow$ Res+унификация $\Rightarrow \emptyset$-дизъюнкт. (здесь $\spadesuit = \exists$ или $\forall$ )
Главное: эти преобразования сохраняют свойство невыполнимости, т.е. если $\Phi_i$ невыполнима, то и множество $D = \{D_1, \dots, D_n\}$ тоже невыполнимо (и наоборот).

**Замечание 9.2.** *Если в сигнатуре $\langle G_2, G_3, G_4 \rangle$ (см. стр.2, конец страницы), $G_3 = \emptyset$, а $G_4$ - конечно, то в интерпретации $I = G_4$ кванторы $\exists x P(x) = P(a_1) \vee \dots \vee P(a_n)$ и $\forall x P(x) = P(a_1) \wedge \dots \wedge P(a_n)$, т.е. кванторы сводятся к формулам над $\{\wedge, \vee, \neg\}$.*

ИП лежит в основе языка Prolog, язык запросов SQL, частично в основе решателя задач Подколзина А.С.

# 10 Лекция 10

**Продукционные модели представления знаний.**

Исчисление высказываний, исчисление предикатов 1-го порядка - это **дедуктивные модели** представления и извлечения знаний. Знания, которые могут содержаться в них надо ещё получить, т.е. вывести из аксиом с помощью правил вывода. Мы рассмотрим здесь ещё одну модель извлечения знаний **- продукционную систему.** (см. лекция 2). В ней правила вывода (продукции) имеют специальный вид:

$$(*) L_1 \vee \dots \vee L_n \subset L,$$

где $L_i, L$ - литеры, т.е. или $p_i$ или $\bar{p}_i$, $p_i$ - логические переменные.

**Определение 10.1.** *Дизъюнкты вида (*) называются **хорновскими** дизъюнктами, или **н-формулами**.*

Н-формулы соответствуют высказыванию типа: Если <условие> то <действие> <следствие>....
Все что мы говорили о формулах исчисления высказываний (логическое следствие, эквивалентные преобразования и.т.д.) справедливо и для н-формул.

**Пример 10.1.** *Н-формула $\Phi$ является логическим следствием Н-формул $\Phi_1, \dots, \Phi_n \Leftrightarrow (\Phi_1 \wedge \dots \wedge \Phi_n \supset \Phi) \equiv 1$. Доказательство аналогично доказательству для формул ИВ.*

Эти формулы удобны при создании экспертных систем (expert systems), или можно описывать технологические процессы производства продукции и.т.д.

**Технологичекий процесс t** $t : L_1 \wedge \dots \wedge L_m \to L$

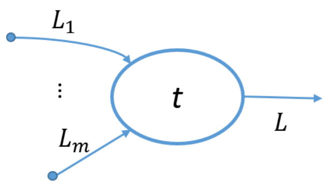



Рис. 22: технологический процесс

"$\to$" - это символ тоже самое, что "$\supset$".
$L_1, \dots, L_n$ - входные продукты (материалы). $L$ - выходной продукт (что получится, если на входы $t$ подать материалы $L_1, \dots, L_m$).

**Пример 10.2.** *$t_1 : L_1 \to L$, $L_1$ - дерево, $L$ - доски. $t_1$ - распилить дерево на доски.
$t_2 : \{L, L_2, L_3\} \to L_4$, $L_2$ - клей, $L_3$ - шурулы, $L_4$ - стол. $t_2$ - из досок с помощью клея и шурулов сделать стол.*

Заметим, что Н-формулы могут иметь более сложный вид:

$$t : L_1 \vee \ldots \vee L_n \to \Phi_1 \vee \ldots \vee \Phi_s$$

Мы будем их рассматривать как множество формул.

$$t^{'}, t^{''}, \ldots, t^{(n)} \begin{cases} L_1 \vee \ldots \vee L_n \to \Phi_1 \\ \ldots \\ L_1 \vee \ldots \vee L_n \to \Phi_n \end{cases}$$

Из процессов $t_1, \ldots, t_l$ можно строить более сложные (составные процессы). В примере 10.1: $t_2 : \{L_1, L_2, L_4\} \to L_4 \vee L_5$, где $L_5$ - шкаф. Тогда $\{L_1, L_2, L_4\} \to$ стол : $t_2$, $\quad \{L_1, L_2, L_4\} \to$ шкаф : $t_2^{'}$. Если $t_1, \ldots, t_n$ - интер-

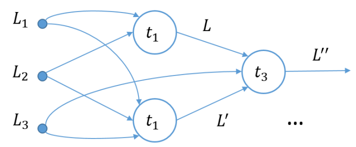



Рис. 23: сетевой график

претировать как время, необходимое для изготовления продукта, то получается сетевой график.
Пусть есть БД, которая содержит исходные материалы (продукты). БД=$\{A, B, C, E, H, G\}$ и есть база знаний (содержит информацию о процессах $t_1, \ldots, t_n$). Например, БЗ $=\{t_1 : F \wedge B \to E, t_2 : C \wedge D \to F, t_3 : A \to D\}$.

**Вопрос 10.1.** *Можно ли получить $Z$, используя БД и БЗ?*

**Прямой процесс (вывод)**
$t_3 : \underbrace{A, A \to D}_{D} \Rightarrow \text{БД}_1 = \text{БД} \cup \{D\}$.

$t_2 : \underbrace{C, D, C \wedge D \to F}_{F} \Rightarrow \text{БД}_2 = \text{БД}_1 \cup \{F\}$.

$t_3 : \underbrace{F, B, F \wedge B \to Z}_{Z} \Rightarrow Z$ можно получить из БД используя процессы $t_3 \to t_2 \to t_1$.

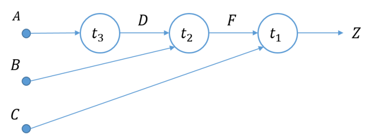



Рис. 24: прямой процесс

$\text{БД}_i$ - состояния БД на $i$-ом шаге.
$\text{БД}_i = \text{БД}_{i-1} \cup \{$результат применения БЗ к $\text{БД}_{i-1};\}$.
$\text{БД}_0$ - начальное состояние.

**Обратный процесс(вывод)**
Ищем в БЗ продукцию, в правой части которой стоит $Z$. Если такой продукции нет, то ответ отрицательный ($Z$ получить нельзя).
Есть в БЗ. (см. рис. обратный процесс). $\quad$ * - обозначает, что соответствующий продукт есть в БД. Т.к. все концевые вершины (листья) дерева $T$ помечены "*", то $Z$ получить можно.

**Масштабирование задачи-десятки тысяч материалов и миллионы процессов**.
Рассмотрим более сложный

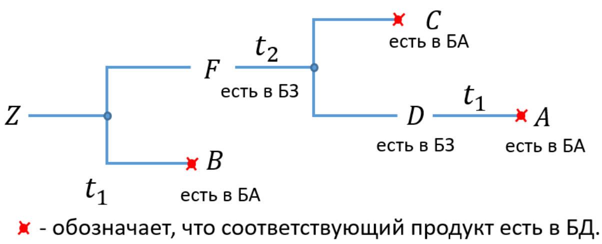



Рис. 25: обратный процесс

**Пример 10.3.** *БД* $= \{A_1, A_2, A_3.A_4, A_5, A_9\}$
*БЗ* $= \{A_1 \wedge A_2 \wedge A_3 \to A_{10}; A_3 \wedge A_1 \to A_7; A_2 \wedge A_5 \wedge A_1 \to A_6; A_9 \wedge A_8 \wedge A_7 \to A; A_1 \wedge A_3 \to A_9; A_1 \wedge A_4 \wedge A_6 \to A_8\} = \{t_1, t_2, t_3, t_4, t_5, t_6\}$.
*Можно ли получить $A$ из {БД} и {БЗ}?*

Заметим что $\forall$ $H$-формулу легко преобразовать в дизъюнкт:
$(L_1 \wedge ... \wedge L_n \to L) = (\overline{L_1 \wedge L_2 \wedge ... \wedge L_n}) \vee L = \bar{L_1} \vee \bar{L_2} \vee ... \vee \bar{L_n} \vee L$ и далле, преобразовать все наши $H$-формулы в обычные дизъюнкты, можно применять известные нам резолютивиые($Res$) процедуры вывода.
Трудности: Как перевести с языка формул на естественный язык?

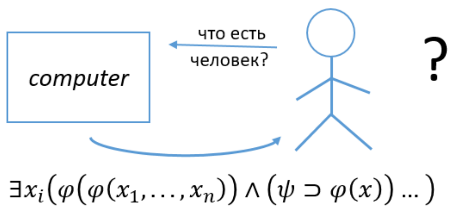



Рис. 26: трудность

**Вопрос 10.2.** *Истинна ли формула ЧЕТ*$(x) \supset$ *ЧЕТ*$(x+1)$

$$\text{где ЧЕТ}(x) = \begin{cases} 1 \text{ если } x=2k \\ 0 \text{ если } x=2k+1 \; k = 0, 1, 2... \end{cases}$$

**Трехзначные логики**

$\{\underbrace{0}_{\text{ложь}}, \underbrace{1}_{\text{неопределено}}, \underbrace{2}_{\text{истина}}\}$ Оказывается что $\supset$ сложно выражается через аналоги $\neg, \wedge$ и $\vee$

| $x$ | $\neg x$ |
|---|---|
| 0 | 2 |
| 1 | 0 |
| 2 | 1 |

| $\vee$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | 1 | 2 |
| 2 | 2 | 2 | 2 |

| $\wedge$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 2 |

| $\supset$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 2 |

Рис. 27: трехзначные логики

$(x \supset y) = \neg\neg(x \vee \neg\neg x) \vee (x \wedge \neg\neg x) \vee (\neg\neg(\neg x \vee \neg\neg x) \wedge y)$

**Замечание 10.1.** *см. рис.*

О языке $PROLOG$ - язык, использующися синтаксис и семантику языка исчисления предикатов. Аксиомы языка $PROLOG$-факты.

| $x$ | $\neg\neg x$ | $\neq x$ |
|---|---|---|
| 0 | 1 | |
| 1 | 2 | |
| 2 | 0 | |

Рис. 28: замечание 10.1

**Пример 10.4.** $F(A,B) = A$ *отец* $B$.
$M(A,B) = A$ *мать* $B$.
$W(A) = A$ - *женщина.*
....

База данных: БФ=$\{$факт 1, ..., факт N$\}$ - база фактов.
База значение: БП=$\{$правила 1, ..., правила K$\}$ - база правил.
БФ=$\{$А отец Б, Б мать В, В мать Г, Г родитель Д, Г - женщина$\}$
БП=$\{$ ① $x$ родитель $y$ если $x$ отец $y$, ② $x$ родитель $y$ если $x$ мать $y$,
③ $x$ дедушка $y$ если ($x$ отец $z$) $\wedge$ ($z$ родитель $y$), ④ $x$ бабушка $y$ если ($x$ мать $z$) $\wedge$ ($z$ родитель $y$),
⑤ $x$ предок $y$ если $x$ родитель $y$, ⑥ $x$ предок $y$ если ($x$ родитель $z$) $\wedge$ ($z$ предок $y$),
⑦ $x$ мать $y$ если ($x$ родитель $y$) $\wedge$ ($x$ женщина), ⑧ $x$ отец $y$ если ($x$ родитель $y$) $\wedge$ ($x$ мужчина)$\}$
**Запрос:** $A$ родитель ?
Интерпретатор $PROLOGF$ обращается к БФ и ищет запрос с начала.
Среди фактов. Факта "А радитель Б" нет. Далее он ищет подходящее правило:
если $x :=$ А, $y :=$ Б то правино ① из БП:
А родитель Б если (А отец Б). Далее нам надо "доказать" чть (А отец Б) но такой факт есть в БФ $\to$ ответ на запрос "А родитель Б"- "ДА".
Более сложно получтиь польжительный ответ на запрос "А предок Д"? Здесь придется перебирать правила ① $-$ ⑧ , прежде чем путем подстановок констант А и Д получться отве "ДА".

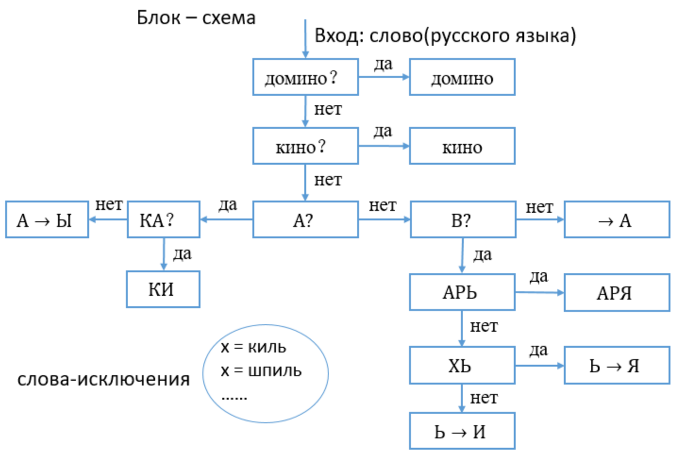
**Замечание 10.2.** *Попробуите самостоятельно получить ответ на этом запрос.*

**Очень Важный факт**
Не существуем интерпретатора(алгоритма), который для любой $PROLOG$-программы и любого ДА/НЕТ запрос позволяет за конечное время(число шатов) получить ответ("ДА" или "НЕТ")
Сравни с существованием универсальной машиный Тьюринга!

# 11 Лекция 11

**Об экспертных системах**
Рассмотрим слова = $\{$ мама, папа, дом, ложка, вилка, кино, домино, печь, ночь, киль, шпиль, лекарь, токарь$\}$. Это слово в именительном падаже. (отвечают на вопрос "кто?" "что?"). Надо написать алгоритм, преобразующий слова в родительный падеж (отвесает на вопросы "кого нет?" "чего нет?"). В таком преобразований меняются только окночание слова. Как меняются окночания этих слов в родительном падеже?
$\{$мамы, папы, дома, ложки, вилки, кино, домино, печи, ночи, киля, шпиля, лекаря, токаря $\}$.
Видим, что "кино" и "домино" не меняются. (см. Блок-схема)
мама, папа: А$\to$Ы, лекарь, токарь: АРЬ$\to$АРЯ, ложка, вилка: КА$\to$КИ.
ночь, печь: ЧЬ$\to$ЧИ, шпиль, киль: Ь$\to$Я.

Как изменить алгоритм (блок-схему), если добавляются слова
$\{$отношение, время, реакция, индукция, задача, функция$\}$ - именительный падеж.
$\{$отношения, времени, реакции, индукции, задачи, функции $\}$.
Алгоритм изменить несложно, но придётся переделывать(перепрограммировать) алгоритм.




| | условие | действие |
|---|---|---|
| 1 | кино | кино |
| 2 | домино | домино |
| 3 | - ЧА | - ЧИ |
| 4 | - КА | - КИ |
| 5 | - А | - Ы |
| 6 | - АРЬ | - АРЬ |
| 7 | (¬Ь) ∧ (ХЬ) | Я |
| 8 | Ь | И |
| 9 | ИЕ | ИЯ |
| 10 | МЯ | МЕНИ |
| 11 | ИЯ | ИН |

Рис. 29: Блок-схема и продукционные системы

Другая идея - использовать продукционные системы Если <условие> то <действие>. (см. 29 справа)
* - слова-исключения={киль,шпиль}.
Легко добавлять новые продукции. Механизм вывода не меняется.
<u>Статика</u> - алгоритм. <u>Динамика</u> - алгоритм на основе продукций.

**<u>Масштабирование задачи</u>**
(миллионы банковских транзакций в день). Язык продукций лежит в основе многих экспертных систем.
**<u>Задача дискретной оптимизации</u>**

1. Задача оптимизации - это последовательность $PR = \{T_1, T_2, \ldots\}$ индивидуальных (конкретных) задач, получающихся из $R$ при конкретном выборе числовых параметров, участвующих в постановке задачи. $T_i \in$PR, $T_i$ - индивидуальная задача.

2. $\forall T_i$ определена совокупность (множество) $R$ допустимых решений $r$ этой задачи.

3. Каждое решение $r \in R$ характеризуется сложностью $l(r)$ - как правило целое число.

В задаче оптимизации требуется по произвольной $T_i \in PR$ найти алгоритм, который бы находил решение $r$ с оптимальным ($l(r) \to \min$ - задача на поиск минимума, или $l(r) \to \max$ - задача на поиск максимума) значением $l(r)$.

Будем рассматривать эти задачи в форме вопроса “верно ли, что для данной индивидуальной задачи $T_i \in PR$ и заданного значения $l$ существует такое решение $r_i \in R$, что $l(r_i) \leqslant l$(или $l(r) \leqslant l$ в задаче на максимум)”. Ответом на вопрос будет “да” или “нет”.

**<u>Кодирование задачи</u>**
Входами в задачах дискретной оптимизации могут быть графы, матрицы, булевы функции, дизъюнкты и.т.д. Предпологаем, что они как-то закодированы в виде наборов нулей и единиц (например так, как их кодируют в компьютере). Такой код имеет определенную длину $n$ (например, число бит или байт), необходимо для представления задачи в компьютере. Кодирование может быть разным (коды одной задачи могут иметь разную длину $n_1, n_2, \ldots$), но как правило для $n_1, n_2$ можно указать такие полиномы $P_1$ и $P_2$, что $n_1 \leqslant P_1(n_2)$, и $n_2 \leqslant P_2(n_1)$. То есть “перекодирование” одного кода в другой требует не больше, чем полином шагов.

**Определение 11.1.** ***Класс P*** *- класс дискретных задач (в форме распознавания), для которых существуют алгоритмы, решающие эти задачи за число шагов* $\leqslant$ *некоторый полином от размера входа задачи.*

Что такое “шаг” или “такт” работы?. Класс $PR \neq \emptyset$, этому классу принадлежит задача о построении минимального остовного дерева графа $G$.

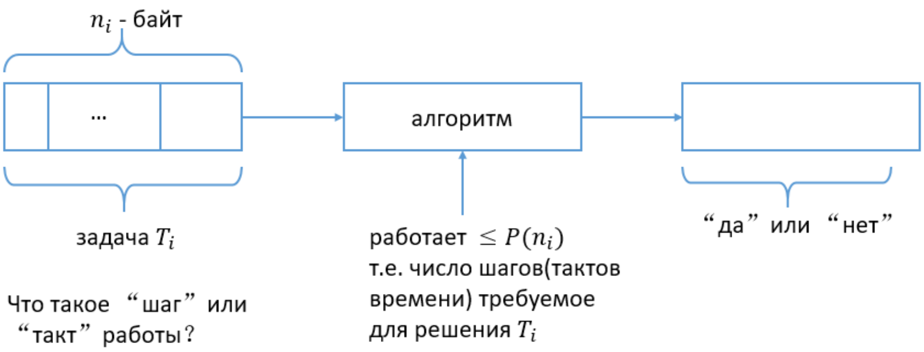



Рис. 30: Кодирование задачи

**Определение 11.2.** ***Класс NP*** *- это задачи, для которых*

1. *Существует алгоритм (переборного типа), решающий эту задачу.*
2. *Мы не знаем, есть ли алгоритм, решающий эту задачу "лучше" (например за полином шагов от размера входа)*
3. *Если нам предъявлены набор значений(возможное решение задачи), то мы за полином шагов можем сказать, является ли это решением задачи ("да" или "нет")*

Позже мы покажем, что класс $NP$ тоже не пуст.

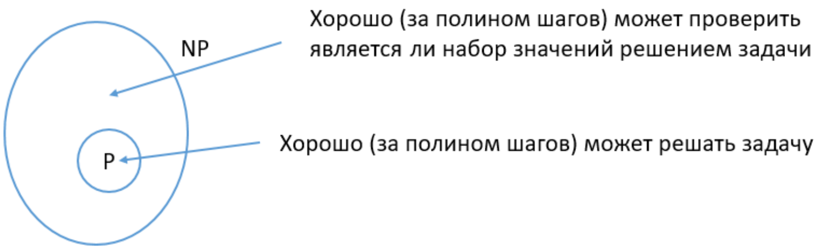



Рис. 31: $P$ и $NP$

**ГЛАВНЫЙ ВОПРОС:** $P? = NP$.

**Задача о покрытии таблицы:**

Таблица из $0, 1$ размера $m \times n$ без строк целиком нулевых.

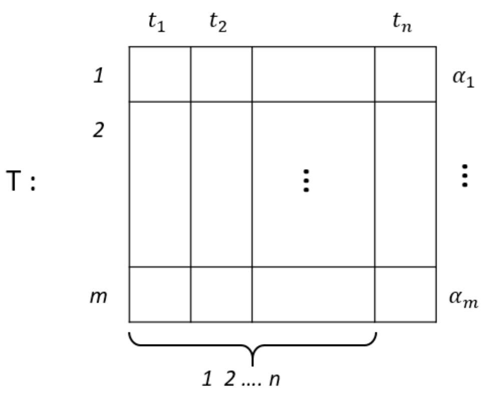
**Определение 11.3.** *Говорим, что стоблец* ***покрывает*** *строку, если на их пересечении в таблице $T$ стоит единица.*

Задача. Найти покрытие всех строк таблицы $T$ столбцами $t_{i_1}, \dots, t_s \subseteq \{t_1, \dots, t_n\}$ так, чтобы $s$ было минимальным.

**Пример 11.1.** *(СМ. рис. 32 справа)*
$t_1, t_3$ *образуют оптимальное покрытие строк таблицы $T$ $(m \cdot n)$ - размер входа задачи $T$.*

Эта задача $\in NP$:

1. Есть переборный алгоритм $= 2^n$ шагов.
2. Неизвестен алгоритм лучше чем перебор (в частности неизвестно, есть ли полиноминальный алгоритм)




T :

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $\alpha_1$ | 0 | 0 | 1 | 1 |
| $\alpha_2$ | 1 | 1 | 0 | 1 |
| $\alpha_3$ | 1 | 0 | 0 | 1 |
| $\alpha_4$ | 0 | 1 | 1 | 0 |
| $\alpha_5$ | 1 | 0 | 0 | 0 |

Рис. 32: Задача о покрытии таблицы и пример 11.111.2

3. Если дан набор столбцов $t_{i_1}, \dots, t_{i_p}$, то проверить, является ли он покрытием $T$ (покрывает ли он все строки) можно сделать за $O(\underbrace{m \cdot p}_{\text{размер входа}})$ шагов. Шаг - это одно сравнение между собой элементы строки и столбца.

**Эвристический (приближенный) алгоритм решения задачи о покрытии таблицы.**

**Определение 11.4.** *Число единиц в столбце $l_i$ называется его **весом** и обозначается $w_i = w(l_i)$.*

АЛГОРИТМ:

1. В таблице $T$ берем столбец $l_i$ наибольшего веса. Если их несколько, то берем любой из них.
2. Вычеркиваем из $T$ все строки, накрытые $l_i$ и сам столбец $l_i$, получим таблицу $T_1$
3. Далее с таблицей $T_1$ проделываем шаги 1 и 2. Получим $T_2$.
4. ......

Условие останова: в очередной подтаблице $T'$ все строки вычеркнуты.

**Пример 11.2.** *(СМ. рис. 32 справа) $t_1$ и $t_2$ - столбцы наибольшего веса $w(t_1) = w(t_n) = 3$. Берем $t_1$ и вычеркиваем строки $\alpha_2, \alpha_3, \alpha_5$ и сам столбец. После вычеркивания $\alpha_1, \alpha_4$ и $t_3$. Больше строк нет. Итак, $t_1, t_3$ оптимальное покрытие.*

$$\mathbf{T_1}: \begin{array}{cccc} & t_2 & t_3 & t_4 \\ \alpha_1 & 0 & 1 & 1 \\ \alpha_4 & 1 & 1 & 0 \end{array}, \quad w(t_3) = 2.$$

Посмотрим (опять рис. 32 справа), что произойдет, если в качестве начального взять столбец $t_4(w(t_4) = 3)$. Вычеркиваем $t_4$ и строки $\alpha_1, \alpha_2, \alpha_3$, накрытые им, получим $T_1$. После этого заметим, что все столбцы веса 1. Берем $t_3$ - вычеркиваем его и строку $\alpha_4$, получим $T_2$. Здесь берем $t_1$(он накрывает $\alpha_5$). Все строки накрыты. $\Rightarrow$ Ответ: $t_4, t_3, t_1$. - не оптимальное.

$$\mathbf{T_1}: \begin{array}{cccc} & t_1 & t_2 & t_3 \\ \alpha_4 & 0 & 1 & 1 \\ \alpha_5 & 1 & 0 & 0 \end{array}, \quad \mathbf{T_2}: \begin{array}{ccc} & t_1 & t_2 \\ \alpha_5 & 1 & 0 \end{array}$$

Проверить, является ли $\{t_2, t_3, t_4\}$ покрытием $T$ (самостоятельно).

Рассмотрим теперь таблицу (рис. 33)следующего вида: она состоит из трех подтаблицы
В первом столбце $2^n$, во втором столбце $2^n$, в третьем столбце $2^n$. —–> единиц
В четвертом $3 \cdot 2^{n-1}$, в пятом $3 \cdot 2^{n-1}$.
...
В $n + 4 : 3 \cdot 2^{n-1}$ единиц.
Эвристический алгоритм выберет в качестве покрытия столбцы с номером $4, 5 \dots, n+4$, т.к. их вас будет $3 \cdot 2^{n-1} > 2^n$ - вес первого, второго и третьего столбца.

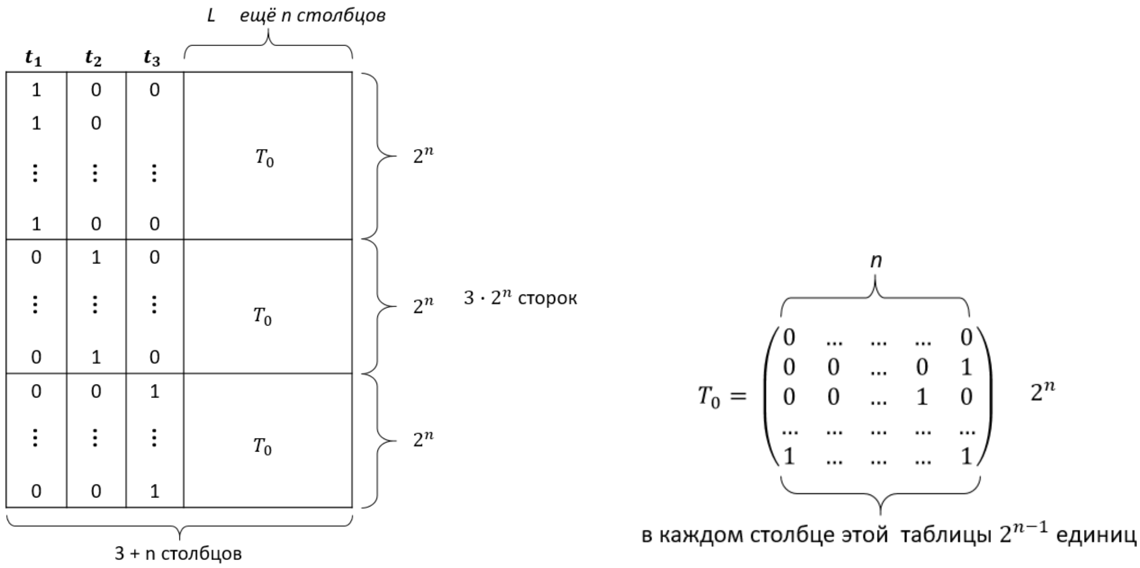



Рис. 33: Таблица специального вида

**Замечание 11.1.** *Легко видно, что столбец* $1, 2, 3$ *является оптимальным решением задачи о покрытии.*

Такие алгоритмы называются **жадными** (greedy).

# 12 Лекция 12

**Содержательно:** проблема $PR_1$ сводится к проблеме $PR_2$, если алгоритм решения проблемы $PR_2$ может быть использован и для решения $PR_1$.
**Формально:**

**Определение 12.1.** *Пусть* $PR_1$ *и* $PR_2$ *- две дискретные задачи в форме распознавания (т.е. ответом в них будет "да" или "нет"). Задача* $PR_1$ ***полиномиально*** *сводится к задаче* $PR_2$*, если для некоторого полинома* $P_i$ *по любой индивидуальной (конкретной)* $T_i^{'} \in PR_1$ *можно построить за* $P(|T_i^{'}|)$ *число шагов индивидуальную задачу* $T_i^{''} \in PR_2$ *так, чтобы ответы в задачах* $T_i^{'}$ *и* $T_i^{''}$ *совпадали (или оба ответа "да" или оба ответа "нет").*

Обозначение $PR_1 \mapsto PR_2$. $|T_i^{'}|$ - размер входа задачи $T_i^{'}$.
Заметим, что отношение "$\mapsto$" транзитивно

$$T_i \underbrace{\mapsto}_{P_1(|T_i|)\text{ полином 1}} T_i^{'} \underbrace{\mapsto}_{P_1(|T_i^{'}|)\text{ полином 2}} T_i^{''} \Rightarrow T_i \mapsto T_i^{''}.$$

Заметим, что подстановка полинома $P_2(P_1(T_i))$ в полином будет опять полиномом. Отсюда следует, что если $PR_1 \mapsto PR_2$ и $PR_2 \in$ классу $P$, то и $PR_1$ тоже $\in P$ (класс задач, решаемых за полиномиальное (от размера входа) число шагов).

На предыдущей лекции мы рассматривали задачу о покрытии таблицы набором столбцов. А в начале курса мы исследовали задачу о выполнимости набора дизъюнктов $D_1, \ldots, D_n$. (Нас интересовало выполнима формула $D_1, \ldots, D_n$ или нет). Нетрудно убедиться, что обе задачи принадлежат классу $NP$.

1. Их можно решить перебором всех подмножеств столбцов - $2^m$ вариантов (или перебором всех значений переменных, входящих в формулу $K(x_1, \ldots, x_s) = D_1 \wedge \ldots \wedge D_p$ - $2^s$ вариантов).

2. Нам не известен алгоритм решения этой задачи, лучший, чем перебор вариантов (например, полиномиального типа).

3. Если нам задан набор столбцов $l_1, \dots, l_p$, $p \leqslant m$ (набор значений переменных $x_1 = \alpha_1, \dots, x_n = \alpha_n (\alpha_i \in \{0,1\})$), то ответ на вопрос задач занимает не более чем $O(|T|)$ шагов, где $T$ или задача о покрытии или задача о выполнимости КНФ $K(x_1, \dots, x_n) = D_1 \wedge \ldots \wedge D_p$.

**Теорема 12.1.** *Задача о выполнимости полиномиально сводится к задаче о покрытии талблиц. Коротко: Задача о выполнимости $\mapsto$ задаче о покрытии.*

*Доказательство.* Пусть $K(x_1, \dots, x_n) = D_1 \wedge \ldots \wedge D_m$ - КНФ. Построим таблицу $T_k$, которая имеет $2n$ столбцов, обозначенных $x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n$, и $n+m$ строк. Строки (*) называются вспомогательными.

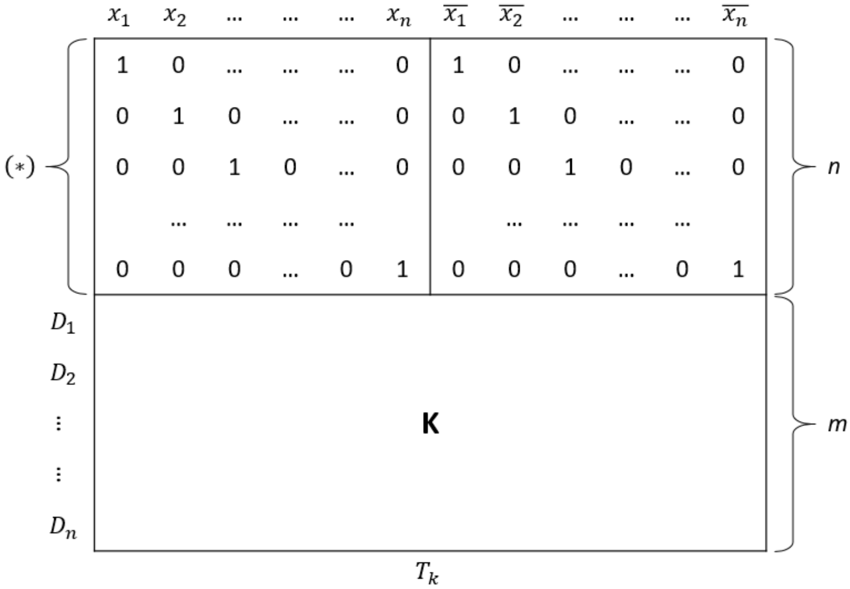



Рис. 34: Таблица $T_k$

Правило заполнения $T_k$ в области $K$: Если переменная $x_i^\delta$ входит в дизъюнкт $D_j$, то на пересечении столбца $x_i^\delta$ и $j$-ой строки ставится 1, в противном случае 0; $j = 1, \dots, m, \delta \in \{0,1\}, i = 1, \dots, n$.

| | $x_1$ | $x_2$ | $x_3$ | $\overline{x_1}$ | $\overline{x_2}$ | $\overline{x_3}$ |
|---|---|---|---|---|---|---|
| Вспомогательные строки | 1 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 1 |
| $D_1$ | 0 | 0 | 0 | 1 | 0 | 1 |
| $D_2$ | 1 | 0 | 0 | 0 | 1 | 0 |
| $D_3$ | 0 | 1 | 1 | 0 | 0 | 0 |

Рис. 35: Таблица к примеру

**Пример 12.1.** $K = \underbrace{(\bar{x}_1 \vee \bar{x}_3)}_{D_1} \wedge \underbrace{(x_1 \vee \bar{x}_2)}_{D_2} \wedge \underbrace{(x_2 \vee x_3)}_{D_3}$.

ВАЖНО: Если какие-то $n$-столбцов образуют покрытие всех вспомогательных строк, то каждая $i$ вспомогательная строка накрыта или $x_i$ или $\bar{x}_i$ - в противном случае, (т.к. в покрытии $n$ столбцов) какая-то переменная не будет участвовать в покрытии и соответствующая строка не будет накрыта.

Следовательно, любое покрытие $T_k$, состоящее из $n$ столбцов (если оно существует) имеет вид $\{x_1^{\alpha_1},\ldots,x_n^{\alpha_n}\}$ - все переменные разные.

**Замечание 12.1.** *Дальше будем считать, что нужное покрытие состоит ровно из $n$ столбцов, т.к. если есть покрытие из $l < n$ столбцов, то добавляя к $l$ столбцам ещё какие-то $n-l$ столбцов мы получим тоже покрытие.*

Каждое покрытие строк $T_k$ столбцами имеет вид $\{x_1^{\alpha_1},\ldots,x_n^{\alpha_n}\}$. Каждая строка $D_i$ "пересекается" по единице с некоторым столбцом $x_i^{\alpha_i}$. Тогда при $x_i^{\alpha_i}=1$ (т.е. $x_i=\alpha_i$) получим, что $D_j=1$.

Полагая $x_1=\alpha_1, x_2=\alpha_2,\ldots,x_n=\alpha_n$ получим что $D_1=1, D_2=1,\ldots,D_m=1 \Rightarrow K(x_1,\ldots,x_n)=D_1\wedge\ldots\wedge D_m=1$, т.е. КНФ выполнима, т.е. если $\{x_1^{\delta_1},\ldots,x_n^{\delta_n}\}$ покрытие $T_k$, то $K(x_1,\ldots,x_n)$ выполнима.

Обратно: Пусть $x_1=\alpha_1,\ldots,x_n=\alpha_n$ и на нем $K(\alpha_1,\ldots,\alpha_n)=1$. Тогда $\{x_1^{\alpha_1},\ldots,x_n^{\alpha_n}\}$ образует покрытие $T_k$.

В самом деле: $K(\alpha_1,\ldots,\alpha_n)=1 \Rightarrow D_1=1,\ldots,D_n=1$, т.е. в каждой строке, соответствующей $D_1,\ldots,D_n$ есть хотя бы одна единица и $i$-ую вспомогательную строку, т.е. если КНФ$(\alpha_1,\ldots,\alpha_n)$ выполнима, то $\{x_1^{\alpha_1},\ldots,x_n^{\alpha_n}\}$ покрытие таблицы.

Теорема доказана. $\square$

Вернемся к нашему примеру 12.1.

1. $\{\bar{x}_1,\bar{x}_2,x_3\}$ образуют покрытие всех строк таблицы. $\{x_1=0,x_2=0,x_3=1\}$ обращают $K(x_1,x_2,x_3)=1$: $K(0,0,1)=(\bar{0}\vee\bar{1})\wedge(0\vee\bar{0})\wedge(0\vee 1)=1$.

2. Возьмем набор значений переменных: $x_1=1,x_2=1,x_3=0 \Rightarrow K(1,1,0)=1$. Тогда $\{x_1,x_2,\bar{x}_3\}$ - эти столбцы накрывают все строки таблицы.

Итак, зная ответ на вопрос "существует ли покрытие таблицы из $n$ столбцов", мы можем ответить и на вопрос "выполнима ли КНФ $K(x_1,\ldots,x_n)$" и наоборот.

Идея сводимости за полином шагов приводит к следующей проблеме: Существуют ли проблемы $PR$, к которым полиномиально сводятся все остальные $PR_i$ ($PR$ и $PR_i$ конечно $\in NP$).

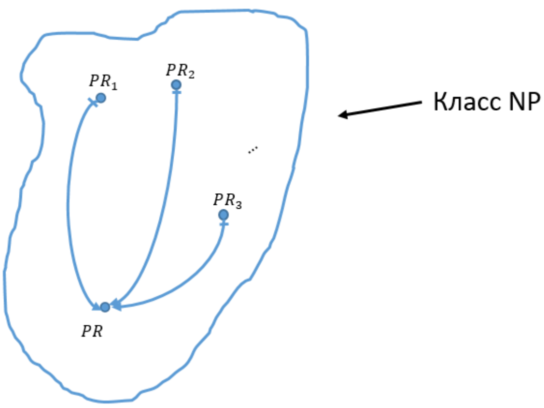



Рис. 36: Класс $NP$

**Определение 12.2.** $\forall PR\in NP \quad (PR_i \mapsto PR\in NP)$. *Такая задача $PR$ (если она существует) будет называется* ***NP-полной*** *задачей.*

Исторически оказалось, что первой $NP$-полной задачей оказалось задача о выполнимости КНФ $K=D_1\wedge\ldots\wedge D_n$. Подумайте о связи с резолютивными методами для ИВ.

Заметим, что вопрос $P=NP$ можно теперь сформулировать так: $P?=NP \Leftrightarrow (\exists PR\in NP\setminus P), PR-NP-$ полная задача.

**Алгоритм приближенного решения некоторых $NP$-полных задач.**

**Задача об упаковки (в контейнеры)**

Задано множество $S=\{p_1,\ldots,p_n\}$ - предметы, $\forall p_i$ обладает "размером" ("весом",...) $r_i=r(p_i)$. Если $S^{'}\subseteq S$, то размер (вес) подмножества $S^{'}$ равен сумме размеров (весов), входящих в него предметов (Обозначение: $r(S^{'})$). Требуется найти такое разбиение $S$, что

1. $S = S^1 \cup S^2 \cup \ldots \cup S^t, S^i \cap S^j = \emptyset, i \neq j, i, j = 1, \ldots, n.$
2. $r(S^1) \leqslant 1, \ldots, r(S^t) \leqslant 1$
3. $t \to \min$

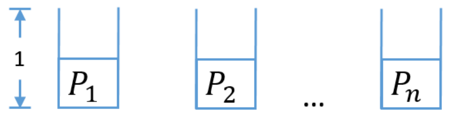



Рис. 37: Задача об упаковки

Если бы были 1) и 2), то это тривиальное решение. (см. рис.) В задаче предпологается, что $\forall i = 1, \ldots, n, r_i = r(p_i) \leqslant 1$. При учете 3) задача становится трудной ($\in NP$).

Обозначим через $t_{opt}$ - минимальное число контейнеров единичного размера, в которые можно упаковать(уложить, поместить) все предметы из $S$. Отметим, что при такой укладке только один контейнер

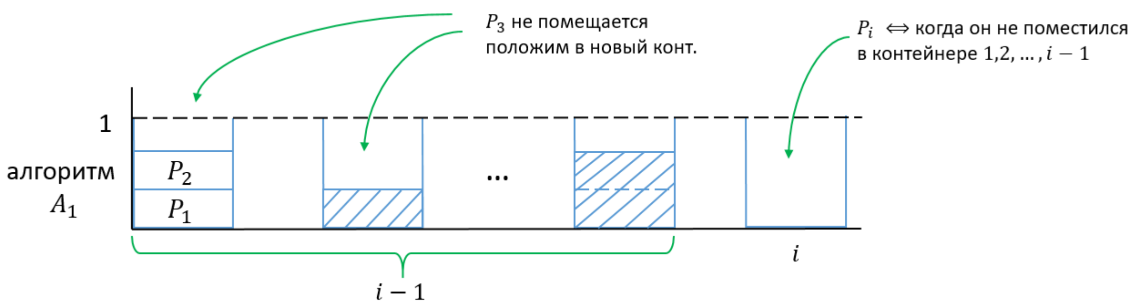



Рис. 38: Алгоритм $A_1$

может быть заполнен меньше, чем на 1/2. Т.е. вот такой случай не возможен: Наш алгоритм предметы

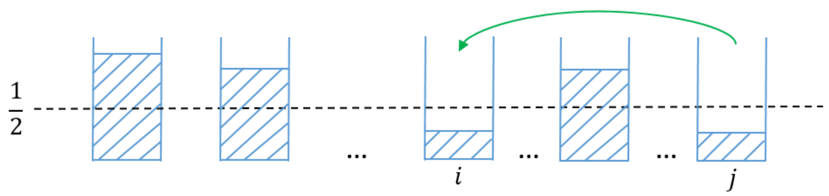



Рис. 39: Невозможный случай

из $j$-го контейнера переместил бы в $i$-й.

Итак, все контейнеры заполнены $> 1/2$ и только один будет (возможно) заполнен меньше, чем на $1/2$. Пусть число контейнеров (при укладке этим алгоритмом) будет $l_{A_1}$. Отбросим последний контейнер.

$$\frac{1}{2}(t_{A_1} - 1) \leqslant \sum_{i=1}^{n} r_i = r(p_i) \leqslant l_{\text{opt}} \Rightarrow \frac{1}{2}(t_{A_1} - 1) < l_{\text{opt}} \quad \boldsymbol{t_{A_1}} \leqslant \boldsymbol{2l}_{\text{opt}}$$

Можно доказать, что $l_{A_1} \leqslant \frac{17}{10} l_{\text{opt}} + 2$ и $\frac{17}{10}$ неулучшаема!

То есть, в худшем случае алгоритм $A_1$ потребует в 2 раза больше контейнеров, чем $l_{opt}$.

АЛГОРИТМ $A_2$:

1. Отсортируем предметы в порядке убывания их весов $r_{i_1} \geqslant \ldots \geqslant r_{i_n}$


Рис. 40: Отбросить последний контейнер

2. К отсортированной последовательности применим алгоритм $A_1$, т.е. сначала берем самый большой предмет $p_{i_1}$, помещаем его в контейнер, затем берем предмет $p_{i_2}$ и пытаемся поместить его в этот же контейнер. Если не помещается - берем новый контейнер, и.т.д.

Можно доказать, что

$$l_{A_2} \leqslant \frac{3}{2} l_{\text{opt}} \text{ и } l_{A_2} \leqslant \frac{11}{9} l_{\text{opt}} + 4$$

и константа $\frac{11}{9}$ не улучшаема! ПРОСЬБА ПОСМОТРЕТЬ САЙТ `packer3d.com`!

# 13 Лекция 13

Рассмотрим граф $K_n$, каждому ребру которого приписано число больше нуля, называемое **весом** ребра, расстоянием между вершинами графа и.т.д. ($K_n = (V, E), |V| = n, E = \{(v_i, v_j)\}$ и $i \neq j$ : нет петель в $K_n$) Предполагается при этом, что выполнено неравенство треугольника, т.е. $\forall$ трех вершин $v_i, v_j, v_k$ графа $K_n$ выполнено

$$w((v_i, v_k)) + w((v_k, v_j)) \leqslant w((v_i, v_j))$$

Из нерваенства треугольника следует обобщенное неравенство треугольника: $\forall$ вершин $v_i, v_{i_1}, \dots, v_{i_n}, v_j$ графа $K_n$ верно

$$w((v_i, v_j)) \leqslant w((v_i, v_{i_1})) + w((v_{i_1}, v_{i_2})) + \ldots + w((v_{i_n}, v_j))$$

Пусть $T$ - минимальное остовное дерево графа $K_n$ и величина $w(T) = \sum\limits_{\text{по всем } (v_i, v_j) \in T} w((v_i, v_j))$ - минимальна и пусть $\pi$-гамильтонов цикл в $K_n$ наименьшего веса, т.е.

$$w(\pi) = w((v_{i_1}, v_{i_2})) + \ldots + w((v_{i_{n-1}}, v_{i_n})) + w((v_{i_n}, v_{i_1})) = w_{\text{opt}}$$

Заметим, что $w(T) < w(\pi)$. (т.к. если из цикла удалить одно ребро, то получим остовное дерево, не обязательно минимальное) В остовном дереве $T$ удвоим ребра (сохраняя отметки на ребрах). Теперь степень каждой вершины четна и граф (полученный из $T$ удвоением ребер) связен, а это необходимое домтаточное условие существования в нем эйлерова цикла. Очевидно $w(C) = 2w(T)$. Эйлеров цикл проходит каждое ребро графа один раз $\Rightarrow$ значит он проходит и все вершины графа (необязательно один раз). Отметим, в эйлеровом цикле все первые вхождения вершин из $V$.

$$C = (v_{j_1}, \ldots, v_{j_2}, \ldots v_{j_n}, \ldots)$$

и рассмотрим обход по циклу $\mu = \{v_{j_1}, v_{j_2}, \ldots, v_{j_n}, v_{j_1}\}$. Из обобщенного неравенства треугольника получаем, что $w(\mu) = w((v_{j_1}, v_{j_2})) + w((v_{j_2}, v_{j_3})) + \ldots + w((v_{j_{n-1}}, v_{j_n})) + w((v_{j_n}, v_{j_1})) \leqslant w(C)$. Но $w(C) = 2w(T) < 2w_{\text{opt}} \Rightarrow w(\mu) < 2w_{\text{opt}}$. Заметим, что "жадный" алгоритм $g$ ближайшего соседа может сильно ошибаться

$$w_g \leqslant \frac{1}{2}\left([\log_2 n] + 1\right) w_{\text{opt}}$$

АЛГОРИТМ $g$: (ближайший сосед) находясь в вершине $v_j$ графа $K_n$ перейти в ближайщую (в смысле величины $w((v_j, v_i))$) вершину $v_i$, которую ещё не проходил.

Можно показать, что существует такой алгоритм посторения гамильтонова цикла $\mu_1$ в $K_n$, что

$$w(\mu_1) \leqslant \frac{3}{2} w_{\text{opt}}$$