Бинарные отношения

22 сентября 2017 г. 21:27

Множество пар А и В - $(a, b \,|a, b \in M)$. Из множества всех пар выделяется некоторое подмножество $(x, y) \in R, xRy - x$ входит в отношение с y.
Свойства:
1. $\forall x \in M \ xRx -$ **рефлексивность**
2. $\forall x \in M \ \overline{xRx} -$ **антирефлексивность**
3. $\forall x, y \in M\colon\ xRy \leftrightarrow \underline{yRx} -$ **симметрия**
4. $\forall x, y \in M\colon\ xRy \rightarrow \overline{yRx} -$ **ассиметрия**
5. $\forall x, y \in M\colon\ xRy, yRx \rightarrow x = y -$ **антисимметрия**
6. $\forall x, y, z \in M \ xRy, yRz \rightarrow xRz -$ **транзитивность**

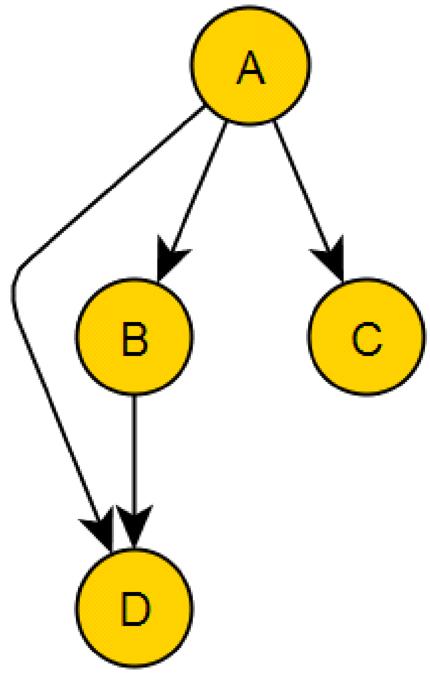
Любое бинарное отношение на конечном множестве может быть представлено как ориентированный граф.

Отношение эквивалентности - 1 + 3 + 6
Отношение толерантности - 1 + 3
Отношение предпорядка - 1 + 6
Отношение порядка $(\geq)$ - 1 + 5 + 6
Отношение строгого порядка $(>)$ - 4 + 6

Метод раскраски
1. Каждой вершине присваивается "цвет"
2. Выбирается необработанное ребро с наименьшим весом. Если добавление ребра в ответ не создаст цикл, оно добавляется. Вершины ребра перекрашиваются в один цвет
3. Если в графе не осталось необработанных ребер, алгоритм завершен. Иначе возврат на шаг 2.

В результате все классы эквивалентности будут окрашены в свой цвет.

Полный (линейный) порядок - $\forall x, y\colon (xRy) \vee (yRx)$. Иначе - **частичный**

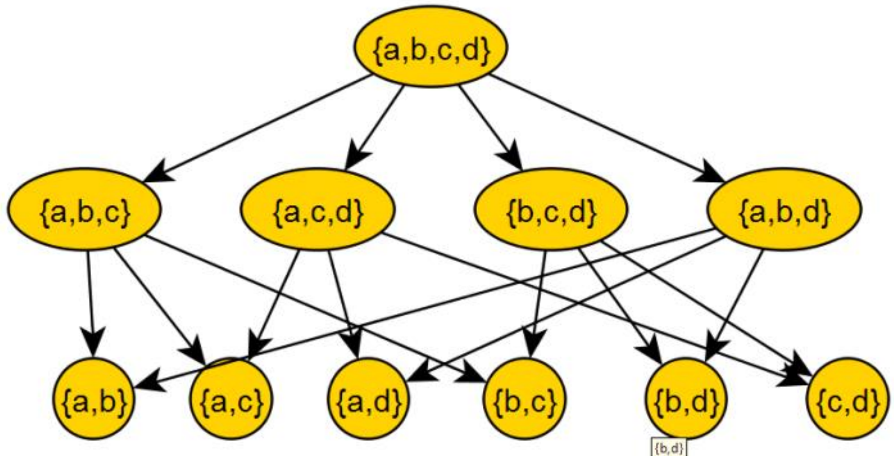


- Отношение неполного порядка.
$L -$ **согласованное отношение**, если $L \geq R$ и достраивает R до полного порядка. В любом множестве, в котором есть порядок, есть минимальный элемент

Алгоритм топологической сортировки

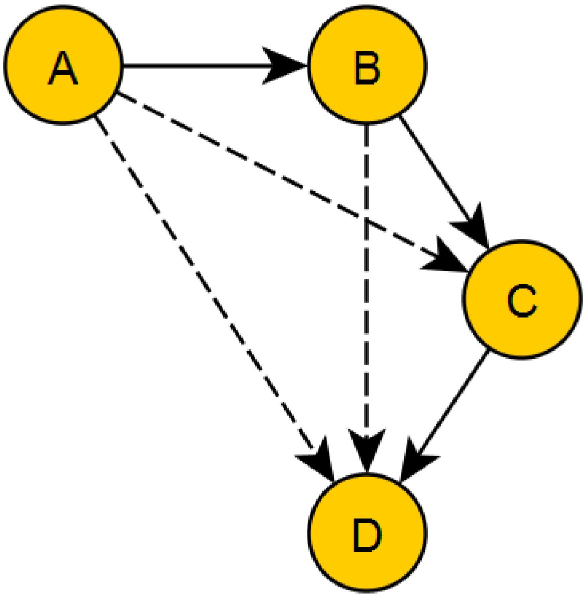
1. Берется минимальный элемент и удаляется. Элементу присваивается номер
2. Если в графе есть элементы, вернутся в 1.
3. Произведена нумерация графа

Диаграмма Хассе

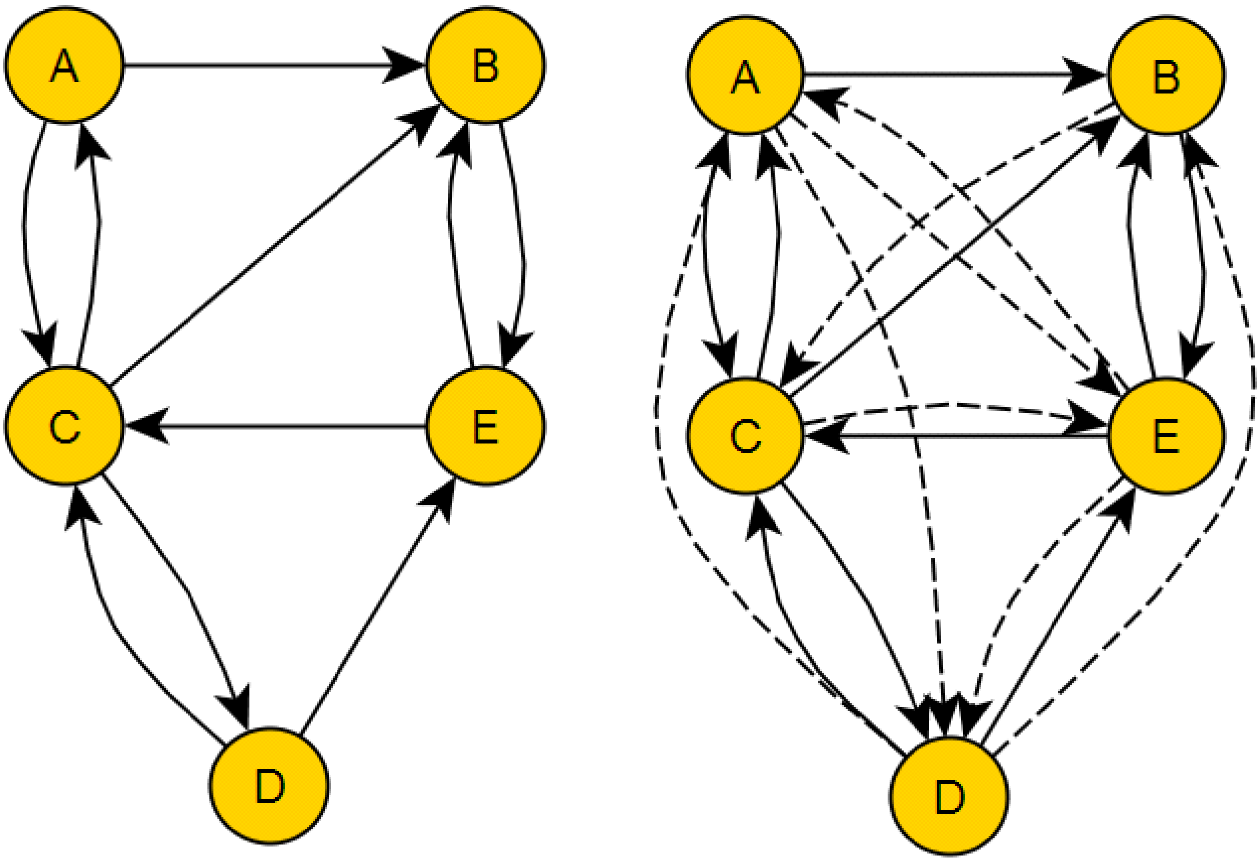


Транзитивное замыкание бинарного отношения
$R -$ бинарное отношение, $P -$ свойство, которым R не обладает.
$\hat{R} \subset R -$ минимальное дополнение R, чтобы свойство выполнялось.

Пример: $\{(a,b)|a > b\} \vee \{(a,b)|b > a\} \rightarrow a \neq b$
Транзитивное замыкание:



Алгоритм Уоршелла
Тройной цикл по u, v, w
$D[u, v] \coloneqq \max\{D[u, v], (D[u, w] \wedge D[w, v])\}$



	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	0	1
C	1	1	0	1	0
D	0	0	1	0	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

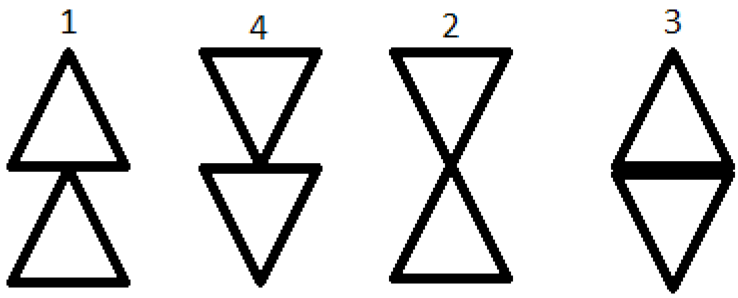
	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	0	1
C	1	1	0	1	1
D	1	1	1	0	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

D	0	0	1	0	1
E	0	1	1	0	0

D	1	1	1	0	1
E	1	1	1	1	0

Логика высказываний

15 октября 2017 г. 15:07



Высказывание - повествовательное предложение, имеющее значение 1 или 0

Основные логические функции

X	Y	$X \wedge Y$	$X \vee Y$	$X \to Y$	$X \leftrightarrow Y$	$\bar{X}$
0	0	0	0	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	0	0
1	1	1	1	1	1	0

$x_1 \dots x_n$	$F(\dots)$
$0 \dots 0$	?
$0 \dots 1$	?
…	…
$1 \dots 1$	1

Вариантов значения n переменных - 2^n. Значит, вариантов таблиц истинности (и функций) от этих переменных -2^{2^n}

Задача Кислера

Браун, Джонсон и Смит обвиняются в подделке сведений о подлежащих налоговому обложению доходах. Они дают под присягой такие показания:
Браун: Джонсон виновен, а Смит невиновен.
Джонсон: Если Браун виновен, то виновен и Смит.
Смит: Я невиновен, но хотя бы один из них двоих виновен.
Кто виновен?

$$\begin{cases} B: J \wedge \bar{S} \\ J: B \to S \\ S: \bar{S} \wedge (B \vee J) \end{cases}$$

$\boldsymbol{B}$	$\boldsymbol{J}$	$\boldsymbol{S}$	$\boldsymbol{B \to S}$	$\boldsymbol{J\bar{S}}$	$\boldsymbol{\bar{S}(B \vee J)}$

0	0	0	1	0	0
0	0	1	1	0	0
0	**1**	**0**	**1**	**1**	**1**
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	1	1	0	0

Равносильность функций

Принцип Дирихле: количество функций неограниченно, количество таблиц истинности ограничено. Значит, есть функции с одинаковыми таблицами истинности.

Две функции **равны**, если у них совпадают таблицы истинности

X	Y	$X \to Y$	$\bar{X} \vee Y$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

1. $FF = F$
2. $F \vee F = F$
3. $FG = GF$
4. $F \vee G = G \vee F$
5. $F(GH) = FGH$
6. $F \vee (G \vee H) = (F \vee G) \vee H$
7. $F(GH) = FG \vee GH$
8. $F \vee (GH) = (F \vee G)(F \vee H)$
9. $F\bar{F} = 0$
10. $F \vee \bar{F} = 1$
11. $\overline{FG} = \bar{F} \vee \bar{G}$
12. $\overline{(F \vee G)} = \bar{F}\bar{G}$
13. $\bar{\bar{F}} = F$
14. $F \to G = \bar{F} \vee G$
15. $F \leftrightarrow G = (F \to G)(G \to F)$

Принцип двойственности

F^* – двойсвенная операция к $\bar{F}$, если в ней все операции заменены на двойственные

$F^*(x_1, \dots, x_n) = \overline{F(\overline{x_1}, \dots, \overline{x_n})}$

$\wedge$ – двойственная операция по отношению к $\vee$

$F = G \Rightarrow F^* = G^*$

$(F^*)^* = F$

Формула G – **логическое следствие** набора $\{F_1, F_2, \dots, F_n\}$, если

$\{F_1, \dots, F_n\}, G$

$\forall (p_1, \dots p_n); p \in [0,1]$

$$\begin{cases} F_1(p_1, \dots p_n) = 1 \\ F_2(p_1, \dots p_n) = 1 \\ \dots \\ F(p_1, \dots, p_n) = 1 \end{cases} \Rightarrow G(p_1, \dots, p_n) = 1$$

Задача

1. Никто, кроме Брауна, Джонсона или Смита, в хищении не участвовал.
2. Браун ходит на дело только с подельником
3. Смит невиновен
4. Джонсон виновен

Является ли 4 следствием 1,2,3?
$F_1 = X \vee Y \vee Z$
$F_2 = X \to Y \vee Z \to (?)G = Y$
$F_3 = \bar{Z}$

$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$	$\boldsymbol{F_1}$	$\boldsymbol{F_2}$	$\boldsymbol{F_3}$	$\boldsymbol{G}$
0	0	0	0	1	1	0
0	0	1	1	1	0	0
0	**1**	**0**	**1**	**1**	**1**	**1**
0	1	1	1	1	0	1
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	**1**	**0**	**1**	**1**	**1**	**1**
1	1	1	1	1	0	1

Выполнимость
Набор формул - **выполнимый,** если
$\exists(p_1, \dots p_n): F_{1\dots n}(p_1, \dots p_n) = 1$
$G -$ логическое следствие набора, когда множество $\{F, .., \bar{G}\}$ невыполнимо

Нормальные формы

15 октября 2017 г. 15:49

Литерал - переменная или её отрицание
Формула с тесными отрицаниями - такая, что отрицания стоят только у переменных
Элементарная конъюнкция - литерал или конъюнкция литералов
Функция имеет **дизъюнктивную нормальную форму (ДНФ),** если она - дизъюнкция элементарных конъюнкций. Любую функцию равносильными преобразованиями можно привести к ДНФ.

Функция, всегда равная 0 - **противоречие,** всегда равная 1 - **тавтология**.

Алгоритм приведения к ДНФ

1. Убираются знаки $\rightarrow, \leftrightarrow$
2. С помощью законов де Моргана отрицания заносятся в формулу
3. $F(H \vee G) = FH \vee FG$

Совершенная ДНФ - СДНФ
СНДФ -

1) Представлена в виде ДНФ
2) Каждая элементарная конъюнкция содержит все литералы
3) Нет одинаковых элементарных конъюнкций

Для любой функции, кроме противоречия, можно построить СДНФ.

Алгоритм, построенный на таблице истинности:

x	y	z	F	
0	0	0	1	$\bar{x}\bar{y}\bar{z}$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{x}yz$
1	0	0	0	
1	0	1	0	
1	1	0	1	$xy\bar{z}$
1	1	1	1	xyz

$F(x, y, z) = \bar{x}\bar{y}\bar{z} \vee \bar{x}yz \vee xy\bar{z} \vee xyz$

Элементарная дизъюнкция - литерал или дизъюнкция
Конъюнктивная нормальная форма (КНФ) - конъюнкция элементарных дизъюнкций
КНФ - двойственная функция к ДНФ. Любую функцию можно привести к

КНФ.

Алгоритм приведения к КНФ

1. Убрать $\rightarrow, \leftrightarrow$
2. С помощью законов де Моргана отрицания занести в формулу
3. $F \vee GH = (F \vee G)(F \vee H)$

Ссовершенная конъюктивная нормальная форма (СКНФ):

1. Представлена в виде КНФ
2. Каждая элементарная дизъюнкция содержит все литералы
3. Нет одинаковых элементарных дизъюнкций

Для любой функции, кроме тавтологии, можно построить СКНФ

Алгоритм, построенный на таблице истинности:

x	y	z	F	
0	0	0	1	
0	0	1	0	$x \vee y \vee \bar{z}$
0	1	0	0	$x \vee \bar{y} \vee z$
0	1	1	1	
1	0	0	0	$\bar{x} \vee y \vee z$
1	0	1	0	$\bar{x} \vee y \vee \bar{z}$
1	1	0	1	
1	1	1	1	

$F(x, y, z) = (x \vee y \vee \bar{z})(x \vee \bar{y} \vee z)(\bar{x} \vee y \vee z)(\bar{x} \vee y \vee \bar{z})$

Методы минимизации формул

15 октября 2017 г. 16:06

Метод минимизирующих карт

Задача - построить ДНФ, минимальную по вхождению переменных. Метод заключается в уменьшении количества перебора ДНФ

Пример

$\cancel{\overline{x_1}}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_1}\,\overline{x_2}$	$\overline{x_1}\,\overline{x_3}$	$\overline{x_2}\,\overline{x_3}$	$\overline{x_1}\,\overline{x_2}\,\overline{x_3}$	0
$\overline{x_1}$	$\overline{x_2}$	x_3	$\overline{x_1}\,\overline{x_2}$	$\overline{x_1}x_3$	$\boldsymbol{\overline{x_2}x_3}$	$\overline{x_1}\,\overline{x_2}x_3$	1
$\overline{x_1}$	x_2	$\overline{x_3}$	$\overline{x_1}x_2$	$\overline{x_1}\,\overline{x_3}$	$x_2\overline{x_3}$	$\overline{x_1}x_2\overline{x_3}$	0
$\overline{x_1}$	x_2	x_3	$\overline{x_1}x_2$	$\overline{x_1}x_3$	x_2x_3	$\overline{x_1}x_2x_3$	0
x_1	$\overline{x_2}$	$\overline{x_3}$	$\boldsymbol{x_1\overline{x_2}}$	$\boldsymbol{x_1\overline{x_3}}$	$\overline{x_2}\,\overline{x_3}$	$x_1\overline{x_2}\,\overline{x_3}$	1
x_1	$\overline{x_2}$	x_3	$\boldsymbol{x_1\overline{x_2}}$	x_1x_3	$\boldsymbol{\overline{x_2}x_3}$	$x_1\overline{x_2}x_3$	1
x_1	x_2	$\overline{x_3}$	x_1x_2	$\boldsymbol{x_1\overline{x_3}}$	$x_2\overline{x_3}$	$x_1x_2\overline{x_3}$	1
x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	0

$F = (\overline{x_2}x_3) \vee (x_1\overline{x_3})$

Вычеркиваются все строчки с нулями. Удаляются все вычеркнутые элементы из невычеркнутых строчек.

Из каждой строки выбирается самая короткая комбинация

Метод Куайна - Мак-Класки

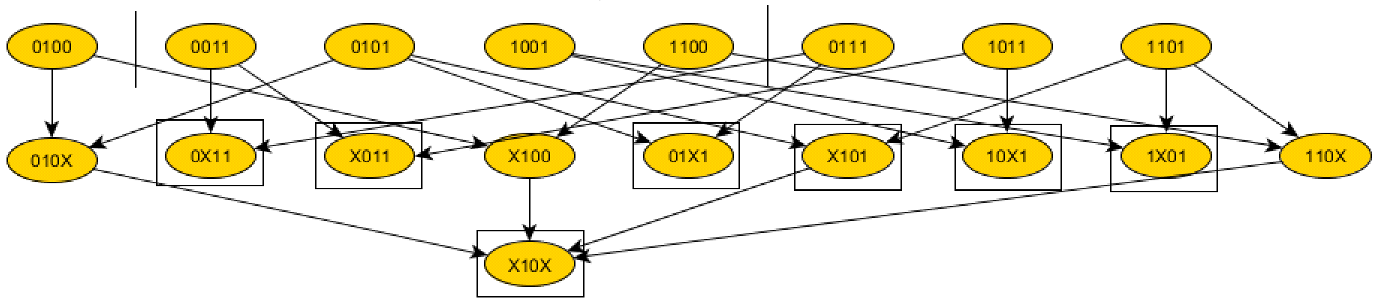
Основан на преобразовании следующего вида:

$x_1x_2x_3 \vee \overline{x_1}x_2x_3 = x_2x_3(x_1 \vee \overline{x_1}) = x_2x_3$

Пример:

$F(x_1, x_2, x_3, x_4)$
$= \overline{x_1}\,\overline{x_2}x_3x_4 \vee \overline{x_1}x_2\overline{x_3}\,\overline{x_4} \vee \overline{x_1}x_2\overline{x_3}x_4 \vee \overline{x_1}x_2x_3x_4 \vee x_1\overline{x_2}\,\overline{x_3}x_4 \vee x_1\overline{x_2}x_3x_4$
$\vee\, x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1x_2\overline{x_3}x_4$

$\{0011, 0100, 0101, 0111, 1001, 1011, 1100\}$



Элементы таблицы истинности делятся на классы по количеству единиц, соединяются элементы только из соседних классов

	0100	0011	0101	1001	1100	0111	1011	1101
0X11		+				+		
X011		+					+	
01X1			+			+		
10X1				+			+	
1X01				+				+
X10X	+		+		+			+

$F = x_2\overline{x_3} \vee x_1\overline{x_2}x_4 \vee \overline{x_1}x_3x_4$

Метод резолюций

$\{F_1, \dots, F_k\} \to G$

Чтобы доказать, что G — следствие, нужно доказать, что $\{F_1, \dots, F_k, \bar{G}\}$ невыполнимо, т.е. вывести из множества формул пустой дизъюнкт

$$\begin{cases} F \vee x \\ G \vee \bar{x} \end{cases} \to F \vee G$$

Пример

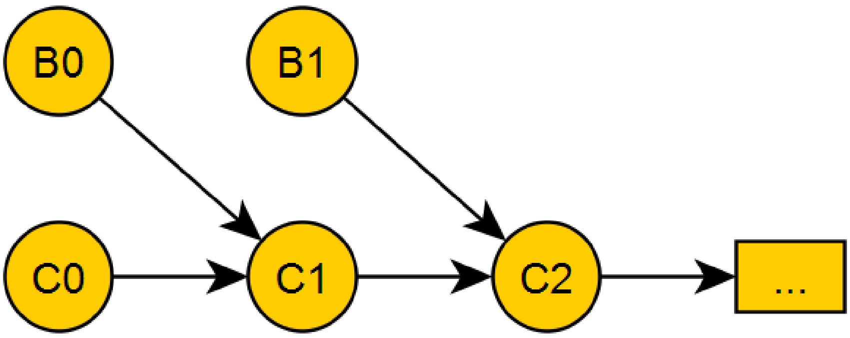
$S = \{D_1 = A \vee B \vee C;\ D_2 = \bar{A} \vee B \vee C; D_3 = B \vee \bar{C}; D_4 = \bar{B} \vee C; D_5 = \bar{C} \vee A; D_6 = \bar{C}\}$

$D_7 = [D_1, D_2] = B \vee C; D_8 = [D_4, D_7] = C; D_9 = [D_6; D_8] = \square$

Для метода резолюций необходима стратегия выбора узлов. Один вариант - перебор всех возможных вариантов.

Линейная резолюция

В этом варианте каждая новая резольвента резольвируется с результатом от предыдущей резолюции



C_{i+1} — **резольвента** C_i, B_i

Задача

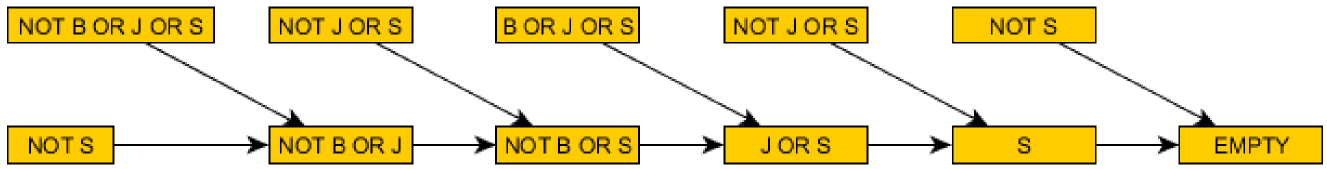
1. Никто, кроме Брауна, Джонсона или Смита, в хищении не участвовал.
2. Если Браун виновен, а Джонсон невиновен, то Сит виновен
3. Джонсон и Смит виновны или невиновны одновременно
4. Если виновен Смит, то виновен и Браун

Следует ли из этого, что Смит виновен

$$\begin{cases} F_1 = B \vee J \vee S \\ F_2 = B \vee \bar{J} \to S \\ F_3 = JS \vee \bar{J}\bar{S} \\ F_4 = S \to B \\ G = S \end{cases}$$

$D_1 = B \vee J \vee S; D_2 = \bar{B} \vee J \vee S; D_3 = \bar{J} \vee S; D_4 = J \vee \bar{S};$

$D_5 = \bar{S} \vee B; D_6 = \bar{S}$


NOT B OR J OR S
NOT J OR S
B OR J OR S
NOT J OR S
NOT S
NOT S
NOT B OR J
NOT B OR S
J OR S
S
EMPTY

Булевы функции

15 октября 2017 г. 20:54

$\{B\} = F(x_1, \dots, x_n)$
$\{0,1,\dots,n\} \to \{0,1\}$
$|B| = 2^{2^n}$
$F(S_1, \dots, S_{k-1}, 0, S_{k+1}, \dots S_n) \neq F(S_1, \dots, S_{k-1}, 1, S_{k+1}, \dots S_n)$

Основные функции:

X	Y	$X \wedge Y$	$X \vee Y$	$X \to Y$	$X \leftrightarrow Y$	$\bar{X}$	$X \oplus Y$	$X \uparrow Y$
0	0	0	0	1	1	1	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	0	1	0
1	1	1	1	1	1	0	0	1

Действия с Булевыми функциями:
1) Переименование переменных
2) Подстановка одной функции в другую

Замкнутость
Набор - **замкнутый**, если никаким конечным числом переименований и подстановок нельзя получить функцию вне набора.
Множество булевых функций **полное**, если его замыкание совпадает с множеством всех булевых функций

Шифр Вернама
Булевы функции могут использоваться в криптографии. Например, в Шифре Вернама исходное сообщение поразрядно объединяется с ключом операцией xor.
Например, исходное сообщение: `01010`, ключ: `11000`, зашифрованное сообщение: `10010`. При применении аналогичной операции к зашифрованному сообщению и ключу, получается исходное сообщение: `01010`.
Данный шифр удобен тем, что не поддается взлому путём перебора, так как в результате перебора всех ключей получатся все возможные сообщения исходной длины. Недостаток тот, что для передачи зашифрованной информации адресату необходимо безопасно передать ключ такой же длины. Также, если каким-то образом узнать исходное сообщение и зашифрованное, можно легко узнать ключ.
Этот шифр применялся во время Второй Мировой Войны. Агент и штаб имели одинаковые блокноты с наборами шифров. После использования страница с набором шифров уничтожалась.

Классы Поста. Критерий Поста

15 октября 2017 г. 21:01

При анализе классов рассматривается, (не) является ли класс пустым, замкнутым и полным.

1. $\boldsymbol{T_0}$ **— множество булевых функций, на 0-м наборе дающих 0**

$f(\dots) \in T_0$

$g_1, \dots, g_k \in T_0$

$p\big(g_1(\dots), \dots, g_k(\dots)\big) = p(0, \dots, 0) = 0$ - класс замкнутый

$|T_0^n| = 2^{2^n-1}$

2. $\boldsymbol{T_1}$ **— набор функций, сохраняющих единицы на наборе из единиц**

$f(\dots) \in T_1$

$g_1, \dots, g_k \in T_1$

$p\big(g_1(\dots), \dots, g_k(\dots)\big) = p(1, \dots 1) = 1 -$ класс замкнутый

$|T_1^n| = 2^{2^n-1}$

3. $\boldsymbol{L}$ **— класс линейных функций**

Булева функция **линейна**, если линеен её полином Жегалкина

Для любой функции существует многочлен Жегалкина:

$$\begin{cases} \bar{x} = x \oplus 1 \\ x \vee y = xy \oplus x \oplus y \end{cases}$$

Поиск многочлена Жегалкина

Метод неопределенных коэффициентов

x	y	z	$p(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$p(x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3$$

$p(0,0,0) = a_0 = \mathbf{0}$

$p(1,0,0) = a_0 \oplus a_1 = 0 \Rightarrow a_1 = \mathbf{0}$

$p(0,1,0) = a_0 \oplus a_2 = 0 \Rightarrow a_2 = \mathbf{0}$

$p(0,0,1) = a_0 \oplus a_3 = 1 \Rightarrow a_3 = \mathbf{1}$

$p(1,1,0) = a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = 1 \Rightarrow a_{12} = \mathbf{1}$

$p(1,0,1) = a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = 1 \Rightarrow a_{13} = \mathbf{0}$

$p(0,1,1) = a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = 0 \Rightarrow a_{23} = \mathbf{1}$

$$p(1,1,1) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3 = 0 \Rightarrow a_{123} = \mathbf{1}$$

$$\boldsymbol{p(x_1, x_2, x_3) = x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_2x_3}$$

Через треугольник Паскаля

	x_1	x_2	x_3									
	0	0	0	0	0	1	0	0	0	1	1	0
x_3	0	0	1	1	1	1	0	0	1	0	1	
	0	1	0	0		0	1	0	1	1	1	
x_2x_3	0	1	1	0		1	1	1	0	0		
	1	0	0	0			0	0	1	0		
	1	0	1	1			0	1	1			
x_1x_2	1	1	0	1				1	0			
$x_1x_2x_3$	1	1	1	0				1				

$$\boldsymbol{p(x_1, x_2, x_3) = x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_2x_3}$$

Полином Жегалкина - **линейный**, если в нем нет нелинейных частей.
Так, x_1x_2
 - нелинейная часть

Лемма о нелинейных функциях
$xy \in [p(x_1, \dots, x_n) \notin L, 0{,}1, \bar{x})$ – из нелинейной функции, констант и отрицания можно получить конъюнкцию

Доказательство:
Так как функция нелинейна, то существует такой набор: $x_1, x_2, \ldots$, при подстановке которого получается самая короткая нелинейная часть x_1x_2.
Подстановка в функцию набора $(x_1, x_2, 1, \dots, 1{,}0, \dots 0)$, где единицы - переменные, участвующие в $x_1, x_2, \dots$, а нули - все остальные

$$p(x_1, x_2, \dots) = x_1x_2 \oplus \alpha x_1 \oplus \beta x_2 \oplus \gamma$$

$$x_1x_2 \oplus x_1 \oplus x_2 = x_1(x_2 \oplus 1) \oplus x_2 \oplus 1 \oplus 1 = x_1\overline{x_2} \oplus \overline{x_2} \oplus 1 = (x_1 \oplus 1)\bar{x}_2 \oplus 1 = \overline{\overline{x_1}\ \overline{x_2}}\blacksquare$$

4. Класс самодвойственных функций

Двойственная функция: $g(x_1, \dots, x_n) = \overline{p(\overline{x_1}, \dots, \overline{x_n})}$
Если $g = p$, то функция самодвойственна
$x_1x_2 \vee x_2x_3 \vee x_1x_3$ - самодвойственна, значит класс непуст
$f \in S; g_1, \dots, g_k \in S$

$$h(x_1, \dots, x_k) = f\big(g_1(x_1, \dots, x_k), \dots, g_n(x_1, \dots, x_k)\big)$$

$$h(\overline{x_1}, \dots, \overline{x_k}) = f\big(g_1(\overline{x_1}, \dots, \overline{x_k}), \dots, g_n(\overline{x_1}, \dots, \overline{x_k})\big) = \overline{f\big(\overline{g_1(x_1, \dots, x_k)}, \dots, \overline{g_n(x_1, \dots, x_k)}\big)} = \overline{f\big(g_1(x_1, \dots, x_k), \dots, g_n(x_1, \dots, x_k)\big)}$$

$h \in S \Rightarrow$ класс замкнут

Лемма о несамодвойственных функциях

Если функция несамодвойственна, то подстановкой в неё вместо аргументов переменной x и $\bar{x}$ можно получить константу

Доказательство:

$[p(x_1, \dots, x_k) \notin S]$

$p(\overline{x_1}, \dots, \overline{x_n}) = p(x_1, \dots, x_k)$

$p(\varepsilon_1, \dots, \varepsilon_n) = p(\overline{\varepsilon_1}, \dots, \overline{\varepsilon_n})$

$p(0, \dots, 0{,}1, \dots, 1) = p(1, \dots, 1{,}0, \dots, 0)$

$g(x) = p(x, \dots, x, \bar{x}, \dots, \bar{x})$

$g(1) = g(0) = const$■

5. **Класс монотонных функций**

Функция $p(x_1, \dots, x_k), \alpha, \beta$ — наборы значений

$\alpha = (\alpha_1, \dots, \alpha_k), \alpha_i \in [0{,}1]$

$\beta = (\beta_1, \dots, \beta_\kappa); \beta_i \in [0{,}1]$

$\alpha \le \beta: \alpha_i \le \beta_i; i \in [1, k]$

$\alpha \le \beta \Rightarrow p(\alpha) \le p(\beta) \Rightarrow$ функция монотонна

Например,

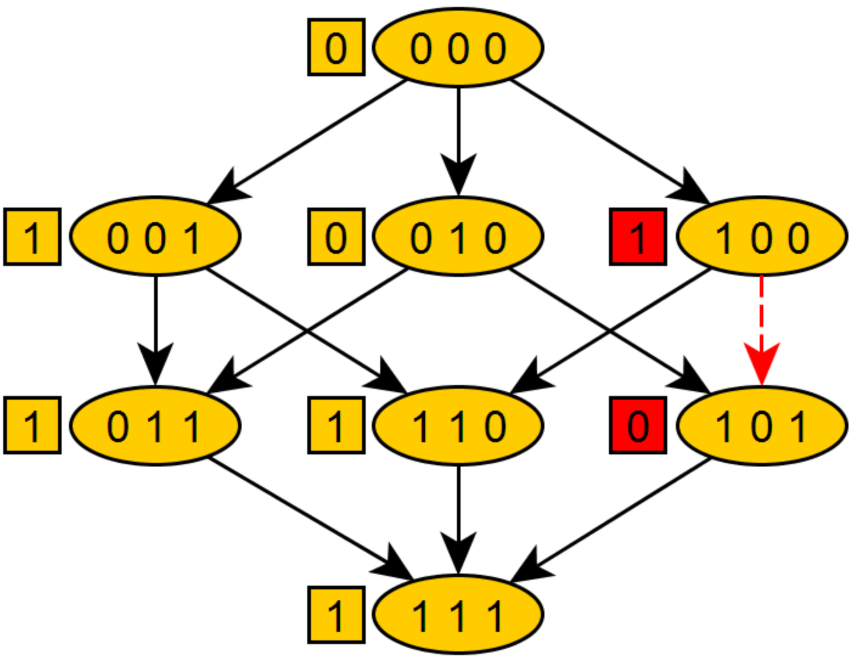
$xy \in M$,

но

$$x \to y \notin M \Leftarrow \begin{array}{cc} (0{,}0) & 1 \\ \wedge & \vee \\ (1{,}0) & 0 \end{array}$$

Для проверки монотонности функции можно использовать диаграмму Хассе

Функция: $p(x_1, x_2, x_3) = (0{,}1{,}0{,}1{,}1{,}0{,}1{,}1)$



Стрелками отмечены отношения "меньше". В данном случае набор $(1{,}0{,}0) < (1{,}0{,}1)$, но $1 > 0$.

Лемма о немонотонных функциях

$f \notin M \Leftarrow \bar{x} \in [\{f, 0{,}1\}]$

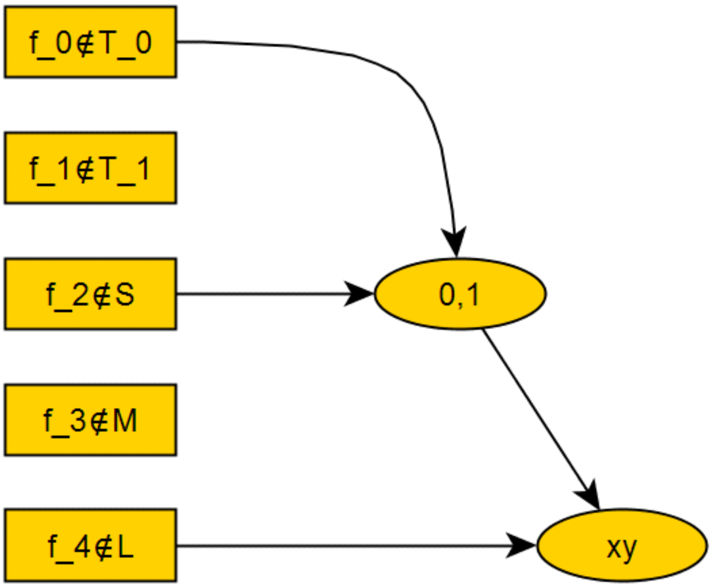
Доказательство:

$f \notin M: \exists \alpha < \beta: p(\alpha) > p(\beta)$
$1 = p(\alpha) > p(\beta) = 0$
$\alpha < \beta \Rightarrow p(\dots, 0, \dots) = 1; p(\dots, 1, \dots) = 0$
$g(x) = f(\dots, x, \dots)$
$p\big(g(0)\big) = 1; p\big(g(1)\big) = 0 \Rightarrow g(x)$ – отрицание ■

Критерий Поста
Множество или класс Поста является полным тогда и только тогда, когда он целиком не содержится ни в одном из замкнутых классов Поста

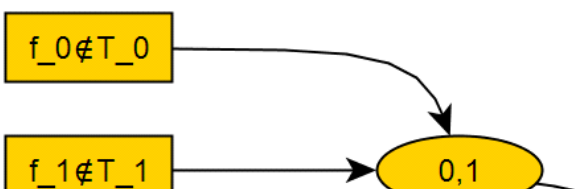
Пример полного класса: $\{xy, x \vee z, \bar{x}\}$
$\{x \vee y, \bar{x}\}, \{xy, \bar{x}\}, \{xy, x \oplus y, 1{,}0\}$ – полный набор

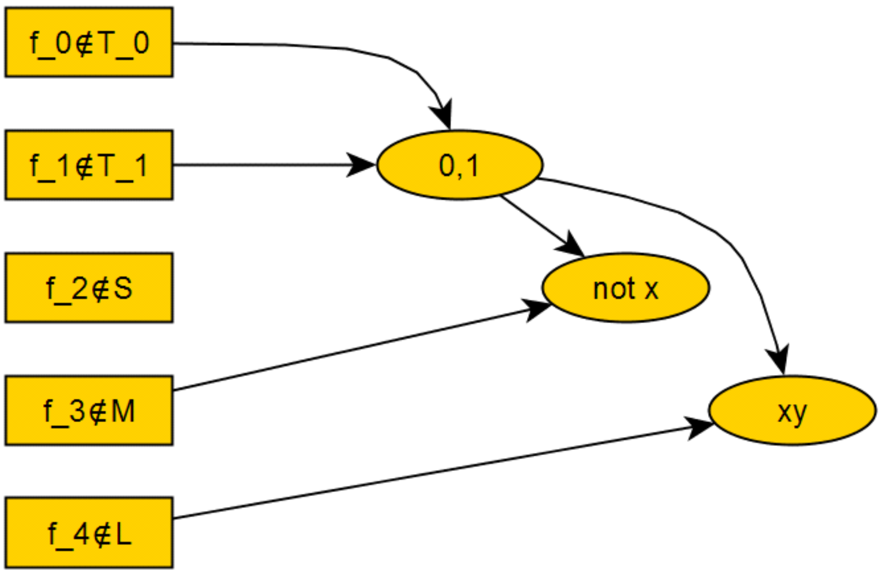
Доказательство:



Пусть полное множество k попало в замкнутый класс. Тогда весь класс - полный. Классы Поста неполны, поэтому множество k не может быть полным.
Пусть k не содержится целиком в T_0, T_1, S, M, L
$k \nsubseteq T_0 \Rightarrow f_0 \notin T_o$
$k \nsubseteq T_1 \Rightarrow f_1 \notin T_1$;
$k \nsubseteq S \Rightarrow f_2 \notin S$
$k \nsubseteq M \Rightarrow f_3 \notin M$
$k \nsubseteq L \Rightarrow f_4 \notin T_o$
$f_{0,1,2,3,4} \in k$
Чтобы доказать, что k – полный, нужно построить ∧ и ¬

1) $f_0 \notin T_0 \Rightarrow f_0(0, \dots, 0) = 1;$ если $f_0(1, \dots, 1) = 0; g(x) = f_0(x, \dots, x)$ – отрицание
2) если же $f_0(1, \dots, 1) = 1; g(x) = f_0(x, \dots, x)$ – константа 1
3) $f_1(1, \dots, 1) = 0; h(x) = f_1\big(g(x), \dots, g(x)\big) = 0 \Rightarrow h(x) = 0$





Следствие: каждый класс содержит полный подкласс не более чем их 4-х функций.

Определение полноты класса
Таблица Поста:

	T_0	T_1	S	M	L
f_1	+	-	+	-	+
f_2	-	+	-	+	-
…					
f_n	+	-	-	+	+

Чтобы система функций была полной, в каждом столбце должен быть хотя бы 1 минус.

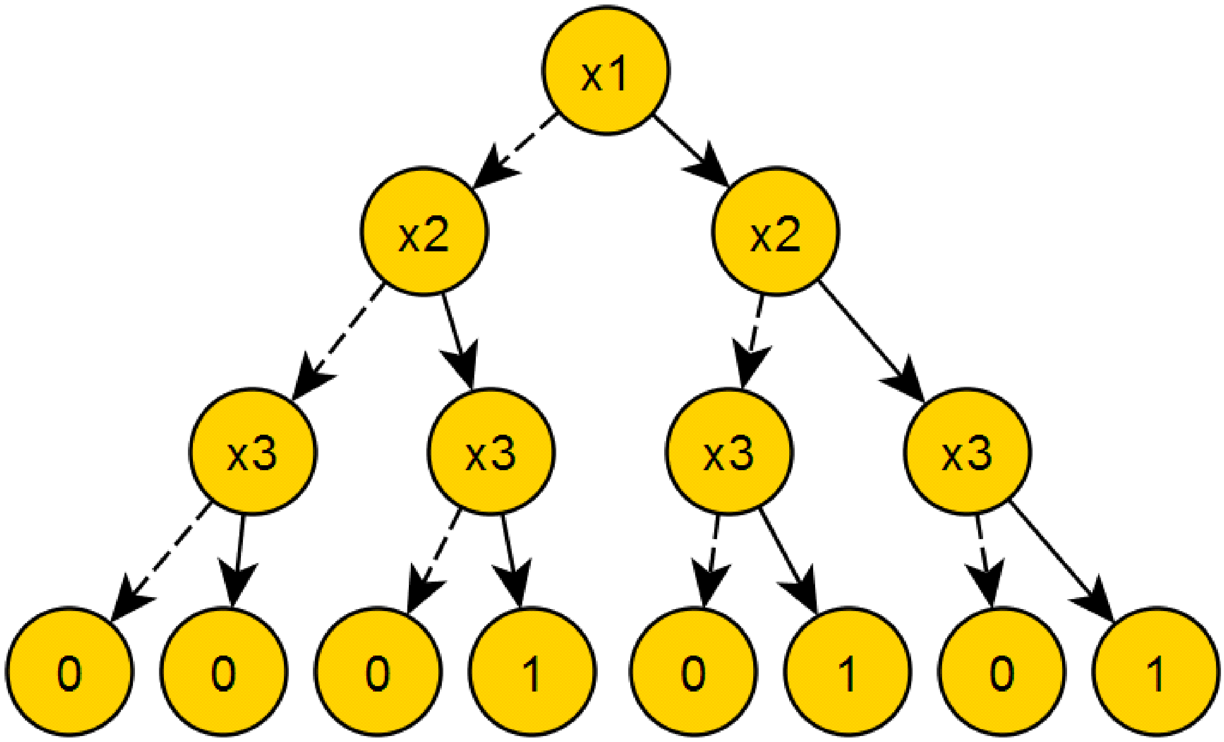
Базис - минимальная полная подсистема. Например, система $\{0,1,xy,x \oplus y\}$ - полная, но базисом не является, т.к. 0 - избыточен: $x \oplus 1 = \bar{x}, \bar{1} = 0$

Разложение Шеннона. OBDD

27 октября 2017 г. 21:32

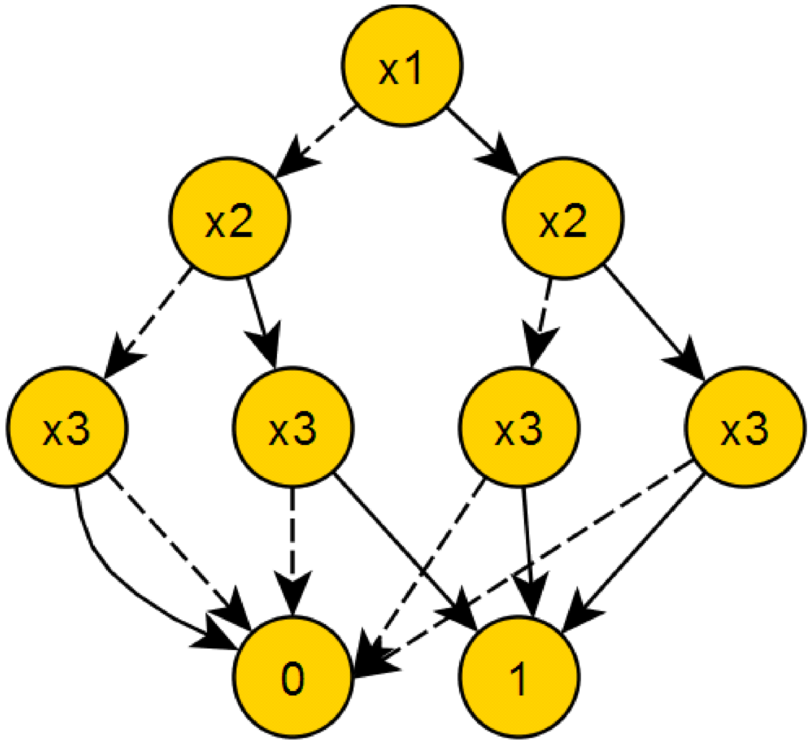
Разложение Шеннона - представление булевой функции N переменных через функции от $N-1$ переменных
Если $\wedge = 0$, то $\vee = \oplus$
$f(x_1, \dots, x_n)$
$= x_i f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \vee \overline{x_i} f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$
Пример:
$f(x_1, x_2, x_3) = x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_3} \vee \overline{x_1} x_2 \overline{x_3} \vee \overline{x_1 x_2} x_3$
$= x_2(x_1 \overline{x_3} \vee \overline{x_1 x_3}) \oplus \overline{x_2}(x_1 x_3 \vee \overline{x_1} x_3) = x_2 f(x, 1, x_3) \oplus \overline{x_2} f(x_1, 0, x_3)$

Бинарная диаграмма решений
Функция $f(x_1, x_2, x_3) = (0,0,0,1,0,1,0,1)$

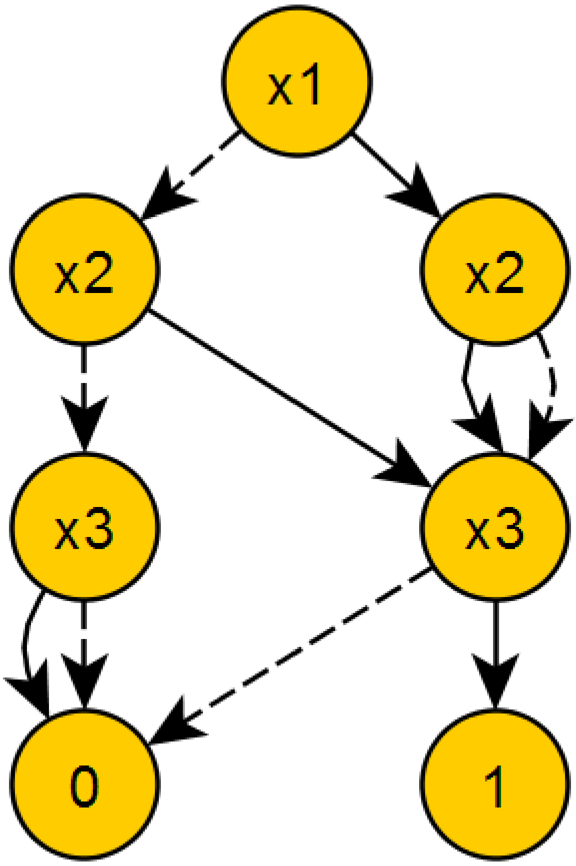


Обычная линия - 1, прерывистая - 0.

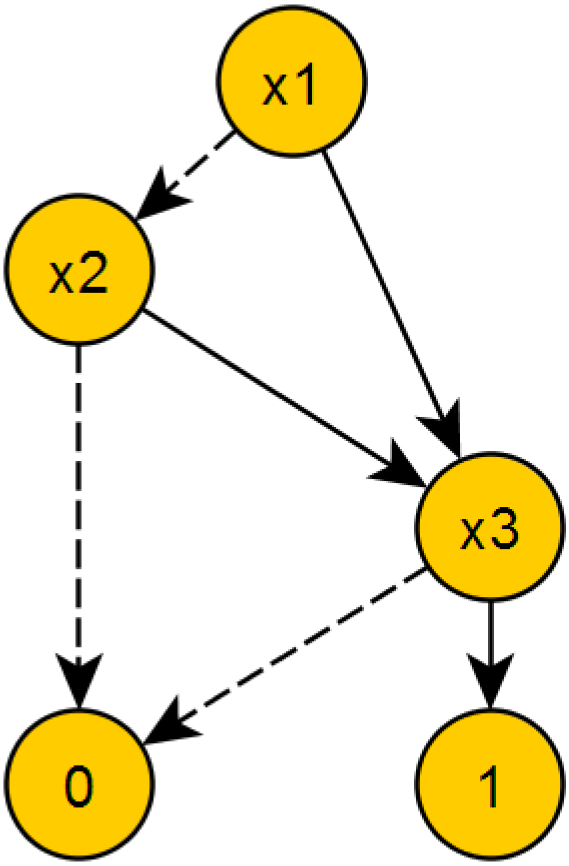
Шаг 1. Объединяются узлы "0" и "1"



Шаг 2. Объединяются вершины одного уровня с одинаковыми выходами



Шаг 3. Удаляются избыточные проверки


x1
x2
x3
0
1

Логика предикатов

29 ноября 2017 г.   16:22

Предикат - высказывание, зависящее от переменной $M -$ предметного множества $p(x_1, \dots, x_n) = M^n \to \{0,1\}$. Предикат - способ, сопоставляющий набору предметов 1 или 0. Для булевой функции предметное множество - двоичные. Предикаты, как и высказывания, могут быть соединены с помощью ∧,∨, ¬ и т.п.

Кванторы
$f(x_1, \dots, x_n, y)$
$G(x_1, \dots, x_n) = (\forall y) P(x_1, \dots, x_n, y)$ - квантор **всеобщности**
$F(\alpha_1, \dots, \alpha_n) = 1 \Leftrightarrow P(\alpha_1, \dots, \alpha_n, y)$ при $\forall y$

$R(x_1, \dots, x_n) = (\exists y) P(x_1, \dots, x_n, y)$ - квантор **существования**
$R(\alpha_1, \dots, \alpha_n) = 1 \Leftrightarrow P(\alpha_1, \dots, \alpha_n, y) = 1$ при некотором y

Сигнатура: $\Sigma = \Sigma_F \cup \Sigma_p \cup \Sigma_c$
$\Sigma f = \{f_i\} -$ функторы, функциональные множество $f(\dots) \to M$

Терм - один предмет, переменная или константа. $f(t_1, \dots t_n)$, t - **терм**, $f -$ **функтор**.

Атомарная формула - $f(t_1, \dots, t_k), t_i -$ термы

Формулы логики предикатов
1. Атомарная формула
2. $(F) \wedge (G)$
   $(F) \vee (G)$
   $(F) \to (G)$
   $\overline{(F)}$
   $(F) \leftrightarrow (G)$
   $(\forall y) F$
   $(\exists y) F$

Вхождение переменной в формулу называется **связным**, если оно стоит в области действия квантора. Если все вхождения связны, то формула - **предложение**

Пример
1. Любой радикал является сторонником общественного прогресса
2. Некоторые консерваторы недолюбливают сторонников общественного прогресса

Значит ли это, что некоторые консерваторы недолюбливают всех

радикалов?
$R(x)$ — радикал, $P(x)$ — сторонник общественного прогресса, $k(x)$ — консерватор. $L(x, y)$ — x недолюбливает y

1. $(\forall x)\big(R(x) \to P(x)\big)$
2. $(\exists x)\Big(k(x) \wedge (\forall y)\big(P(y) \to L(x, y)\big)\Big)$
3. $(\exists x)\Big(k(x) \wedge (\forall y)\big(R(y) \to L(x, y)\big)\Big)$

Равносильность формул

$F = G$, если для любого предметного множества на любых наборах значения формул совпадают. Для предикатов верны все равносильности булевых функций. Кроме того, есть новые формулы:
$(\forall x)\big(F(x) \wedge G(x)\big) = (\forall x)F(x) \wedge (\forall x)G(x)$
$(\exists x)\big(F(x) \vee G(x)\big) = (\exists x)F(x) \vee (\exists x)G(x)$
Однако, следующие формулы неверны:
$(\forall x)\big(F(x) \vee G(x)\big) \neq (\forall x)F(x) \vee (\forall x)G(x)$
$(\exists x)\big(F(x) \wedge G(x)\big) \neq (\exists x)F(x) \wedge (\exists x)G(x)$
Чтобы это понять, достаточно привести контрпример
Также верны формулы:
$(\forall x)(\forall y)\big(R(x, y)\big) = (\forall y)(\forall x)\big(R(x, y)\big)$
$(\exists x)(\exists y)\big(R(x, y)\big) = (\exists y)(\exists x)\big(R(x, y)\big)$
И неверна формула
$(\exists x)(\forall y)\big(R(x, y)\big) \neq (\forall y)(\exists x)\big(R(x, y)\big)$
Верны формулы:
$\overline{(\forall x)F(x)} = (\exists x)\overline{F(x)}$
$\overline{(\exists x)F(x)} = (\forall x)\overline{F(x)}$
$(\forall x)(F(x) \vee G) = (\forall x)F(x) \vee G$
$(\exists x)(F(x) \vee G) = (\exists x)F(x) \wedge G$
Возможно производить **переименование**
$(\forall x)f(x) = (\forall z)f(z)$
$(\exists x)f(x) = (\exists z)f(z)$

Нормальная форма

30 ноября 2017 г.   20:50

Логическое следствие в логике предикатов
$\{F_1, \dots, F_k\}, G$
На любом наборе переменных, на которых $F_{1\dots k} = 1 \Rightarrow G = 1$.
Чтобы показать выполнимость набора, нужно привести пример, в котором показывается выполнение.

Теорема о логическом следствии.
$G -$ логическое следствие набора F, если невыполнимо $\{F_1, \dots, F_n, \bar{G}\}$

Нормальная форма в логике предикатов.
ПНФ - **предварённая/префиксная нормальная форма**. Формула имеет вид ПНФ, если она имеет вид:
$(Q_1 x_1)(Q_2 x_2) \dots (Q_k x_k) F(\dots)$
Где $Q_{1\dots k} -$ кванторы, а $F -$ безкванторная формула - "**матрица**".

Приведение к ПНФ:
1. Убрать $\vee, \wedge, \neg, \rightarrow$ с помощью равносильных булевых преоблазований
2. Занесение отрицаний отдельную в формулу
3. Вынос квантора

Пример
$\overline{(\exists x)\big(P(x)\big)} \vee (\forall x)(Q(x,y) =$
Раскрытие отрицания через равносильную формулу
$= (\forall x)\big(\bar{P}(x)\big) \vee (\forall x)\big(Q(x,y)\big) =$
Переименование x в z в первой части
$= (\forall z)(\bar{P}(z)) \vee (\forall x)(Q(x,y)) =$
Так как первая часть не зависит от x, а вторая - от z, можно вынести
$= (\forall z)(\forall x)(\bar{P}(z) \vee Q(x,y))$

СНФ - Сколемовская нормальная форма
Формула представлена в СНФ, если она - универсальная и её матрица имеет вид КНФ. СНФ неравносильна с исходной, но сохраняет выполнимость.

Приведение к СНФ
0. Привести к ПНФ
1. Удаление квантора $\exists -$ **иллюминация**
   Если квантор $\exists x$ первый, то квантор удаляется, а x заменяется на константу α, которой до этого не было в матрице
   Если же имеем, например, $(\forall x)(\forall y)(\exists z)$, то после удаления $\exists z$ делается замена z на функтор $z = f(x,y)$.

2. Привести матрицу к КНФ

Пример

$(\exists x)(\forall y)(\exists z)\left(\overline{P(x,y)} \vee Q(x,z)G(y)\right) =$

Равносильное преобразование $A \vee BC = (A \vee B)(A \vee C)$

$= (\exists x)(\forall y)(\exists z)\big(\bar{P}(x,y) \vee Q(x,z)\big)\big(\bar{P}(x,y) \vee G(y)\big) \rightarrow$

$x = \alpha$. При иллюминации равносильность теряется, поэтому знак $\rightarrow$.

$(\forall y)(\exists z)\big(\bar{P}(\alpha,y) \vee Q(\alpha,z)\big)\big(\bar{P}(\alpha,y) \vee G(y)\big) \rightarrow$

Замена $z = f(y)$. Это преобразование также неравносильно

$\rightarrow (\forall y)\left(\bar{P}(\alpha,y) \vee Q\big(\alpha,f(y)\big)\right)\left(\bar{P}(\alpha,y) \vee G(y)\right)$

Приведение к КНФ было рассмотрено ранее. В данном случае матрица уже в КНФ

Метод резолюций

30 ноября 2017 г. 21:41

Предварительный алгоритм - создание множества дизъюнктов из формул

1. Каждую формулу привести к СНФ
2. **Замкнутая формула** - все переменные связаны кванторами. Если есть свободные переменные, то они заменяются константами.

В итоге получается множество дизъюнктов, выполнимое одновременно с множеством формул.

Подстановка, унификация

Подстановка - множество равенств $\delta = \{x_1 = t_1, \dots, x_n = t_n\}$, где $x_1, \dots, x_n -$ переменные, $t_1, \dots, t_n -$ термы, причём терм t_i не содержит переменной x_i.

$S -$ множество дизъюнктов.

$\varepsilon -$ пустая подстановка

$\delta_2 \circ \delta_1 -$ композиция, т.е. применение сначала δ_1, потом δ_2.

Пример

$S = \{R(x, y) \vee P(f(y), z), Q(y, z, z)\}$

$\delta = \{x = g(y), y = \alpha, z = h(x)\}$

$\delta(S) = \{R(g(y), \alpha) \vee (P\big(f(\alpha), h(x)\big), Q(\alpha, h(x), h(x)\}$

Подстановка $\delta -$ **унификация**, если для множества литералов $\{L_1, \dots, L_k\}$: $\delta(L_1) = \dots = \delta(L_k)$. Множество литералов **унифицируемо,** если существует унификатор этого множества

Бинарная резольвента

$\bar{Q}(t_1, \dots, t_k) \vee F; Q(r_1, \dots, r_k) \vee G$

$\delta: \{G(t_1, \dots, t_k), Q(r_1, \dots, r_k)\}$

$\delta(F) \vee \delta(G) -$ **бинарная резольвента**.

Пример

$\begin{cases} P(x) \vee Q(x) \\ \bar{P}(\alpha) \vee R(y) \end{cases}$

$\delta: \{x = \alpha\}$

$Q(\alpha) \vee R(y)$

Склейка

$D = L_1 \vee L_2 \vee \cdots$

$\delta(L_1) = \delta(L_2) = \cdots$

$\delta(D) -$ **склейка D**

Пример

$\bar{P}(x) \vee \bar{P}\big(F(y)\big) \vee G(x,y)$

$\delta{:}\,\{x = f(y)\}$

$\delta(D) = P\big(f(y)\big) \vee G(f(y),y)$

Метод резолюций

Дан набор формул:

$f_1 = (\forall x)\big(E(x) \to V(x)R(x)\big)$

$f_2 = (\exists x)\big(E(x) \wedge Q(x)\big)$

$G = (\exists x)\big(Q(x) \wedge F(x)\big)$

Нужно проверить, является ли G логическим следствием f_1, f_2. Для этого нужно доказать невыполнимость множества $\{f_1, f_2, \bar{G}\}$

Сколемизация:

$f_1 \to (\forall x)\left(\overline{E(x)} \vee V(x)\right)\left(\bar{E}(x) \vee R(x)\right)$

$f_2 \to E(\alpha)Q(\alpha)$

$\bar{G} \to (\forall x)\big(\bar{Q}(x) \vee \overline{R(x)}\big)$

Множество дизъюнктов:

$D_1 = \bar{E}(x) \vee V(x); D_2 = \overline{E(x)} \vee R(x); D_3 = E(\alpha); D_4 = Q(\alpha); D_5 = \left(\bar{Q}(x) \vee \bar{R}(x)\right)$

$D_6 = [D_2, D_3{:}\,x = \alpha] = R(\alpha); D_7 = [D_5, D_6{:}\,x = \alpha] = \bar{Q}(\alpha); D_8 = [D_4, D_7] = \square$

Общего алгоритма перебора нет, т.к. число перестановок бесконечно. Чтобы этого избежать, упрощается входное множество.

Элементы ПРОЛОГа

Дизъюнкт, в котором не более одного литерала входит без отрицания - **предложение**.

Равносильными преобразованиями такое предложение может быть преобразовано к виду:

$L \vee \overline{L_1} \vee \overline{L_2} \vee \cdots \vee \overline{L_n} = L_1 L_2 \dots L_n \to L$ $-$ **правило**,

где $L -$ **заголовок**.

При $n = 0$ правило называтся **фактом**

Если ни один литерал не входит в дизъюнкт без отрицания, то его можно преобразовать к виду

$\overline{L_1} \vee \overline{L_2} \dots \vee \overline{L_k} = \overline{L_1 \dots L_n} \to L$

Это записывается так:

$? - \mathrm{L_1 L_2 \dots L_k} -$ **запрос**

Чаще всего для проверки используется линейная резолюция.

Пример

```
 1. Программирует(Иванов)
 2. Программирует(Петров)
 3. Читал(Иванов, Ире Пол)
 4. Читал(Петров, Смешарики)
 5. Книга(Ире Пол, программирование)
 6. Книга(Смешарики, детская)
 7. Сдавал(Иванов)
 8. Сдавал(Петров)
 9. Сдавал(x),программирует(x),знает(x) → экзамен(x,5)
10. Сдавал(x),программирует(x) → экзамен(x,4)
11. Сдавал(x) → экзамен(x,3)
12. Читал(x,y),книга(y,программирование) → знает(x)
13. ?-Экзамен(Петров,z)
```

Здесь 1-8 - факты, 9-12 - правила 13 - запрос

Сначала унифицируется с фактами, если не выходит - с правилами.

```
14. ?-Сдавал(Петров),программирует(Петров),
    знает(Петров)
15. ?-Читал(Петров,y),книга(y,программирование
```

Ни с чем не унифицируется. Шаг назад

```
16. Сдавал(Петров),программирует(Петров)
    z=4
```

Машина Тьюринга

1 декабря 2017 г. 11:22

Имеется бесконечная лента, нарезанная на ячейки. $A = \{\alpha_0, \alpha_1, \dots, \alpha_n\}$ – алфавит, $Q = \{q_0, q_1, \dots, q_n\}$ - множество состояний машины Тьюринга.
q_1 – стартовое состояние, q_0 – заключительное

$$q_j\alpha_i \to q_s\alpha_j \left\{ \begin{array}{l} R - \text{вправо} \\ L - \text{влево} \\ C - \text{стоп} \end{array} \right.$$

… | α_i | …
q_j

Конфигурация - все, что есть на ленте, текущая ячейка и состояние.
Применимость конфигурации - возможность перейти в конечное состояние за конечное число шагов.

Вычисление функций на машине Тьюринга
Унарный алфавит - $\{\alpha_0, 1\}$. α_0 – пустой символ
$f\colon \mathbb{N}^+ \to \mathbb{N}^+$ $(\mathbb{N}^+ = \{0\} \cup \mathbb{N})$

$k \to |\ 1\ |\ 1\ |\dots|\ 1\ |\ - 1^{k+1}$
Ф-я f, вычисляемая на данной машине Тьюринга: начальная конфигурация 1^{k+1}, конечная конфигурация $1^{f(k)+1}$, число шагов конечно
Для $f(x,y)$ в алфавит вводится разделитель
$1^{x+1} + 1^{y+1} \to 1^{f(x,y)}$

Пример 1
$f(x) = x + 1$
$\alpha_0\ |\ 1\ |\ 1\ |\dots|\ 1\ |\ \alpha_0$
$\uparrow$
q_1
$q_1 1 \to q_1 1 R$ – идем вправо, пока встречаем 1.
$q_1\alpha_0 \to q_2 1 L$ – заменяем встреченный пустой символ на 1 и делаем шаг влево
$q_2 1 \to q_2 1 L$ – идем влево, пока встречаем 1
$q_2\alpha_0 \to q_0\alpha_0 R.$ – Делаем шаг вправо, когда встречаем пустой символ. Конец.

Пример 2
$f(x,y) = x + y$
Начальное состояние: $1 \dots 1 \times 1 \dots 1$
$q_1 1 \to q_1 1 R$
$q_1 \times \to q_2 1 R$
$q_2 1 \to q_2 1 R$
$q_2\alpha_0 \to q_3\alpha_0 L$

$q_3 1 \to q_4 \alpha_0 L$
$q_4 1L \to q_5 \alpha_0 L$
$q_5 1 \to q_5 1L$
$q_5 \alpha_0 \to q_0 \alpha_0 R$.
Сначала идем вправо, заменяем $\times$ на 1, идем в конец, отрубаем две единицы (т.к. число k представлено $k+1$ единицей, чтобы различать 0) и идем в начало.

Пример 3
Алфавит - $\{0,1,\dots,9\}$
α_0 2 0 1 7 α_0
$\downarrow$
α_0 2 0 1 8 α_0
$q_1\{0,\dots,9\} \to q_1\{0,\dots 9\}R$
$q_1\alpha_0 \to q_2\alpha_0 L$
$q_2\{0,\dots,8\} \to q_3\{1,\dots,9\}L$
$q_3\{0,\dots,9\} \to q_3\{0,\dots,9\}L$
$q_3\alpha_0 \to q_0\alpha_0 R$
$q_2 9 \to q_2 0L$
$q_2\alpha_0 \to q_0 1C$.

Работает следующим образом: сначала перемещается в конец. Если встречает цифры от 0 до 8, то меняет их на цифры от 1 до 9 и идет назад (состояние q_3). Если же встречает 9, до меняет цифру на 0, и делает шаг влево. Если при этом встречается цифра от 0 до 8, то $+1$ и переход в состояние q_3, а если дошли до конца в состоянии q_2, то в начало числа приписываем 1.

Суперпозиция - соединение машин Тьюринга.
$T_1, T_2 -$ машины Тьюринга
$q_k^1, q_k^2 -$ состояния
$T = T_2 \circ T_1 = T_2(T_1)$
$q_1 = q_1^1$
$q_1^2 = q_0^1$
Возможна реализация условных переходов:
$q_1 0k \to T_0$
$q_1 1k \to T_1$

Проблема останова - есть ли алгоритм, который определяет применимость машины Тьюринга.
Тезис Тьюринга - любой алгоритм может быть представлен в виде машины Тьюринга. Значит, чтобы доказать, что алгоритма нет, нужно доказать, что не существует такой машины Тьюринга. Однако, тезис Тьюринга <u>не доказан</u>, поэтому лежит в области философии.

Гёделева нумерация - абстрактный способ запомнить машину Тьюринга. Этот способ неприменим на практике из-за того, что не существует эффективного алгоритма разложения на простые множители. Однако, способ используется при доказательствах.

A_0 — внешний алфавит

Q_0 — внутренний алфавит

$\{p_i\}$ — последовательность всех простых чисел в порядке возрастания

Пусть машина Тьюринга:

$q_{i_1}\alpha_{j_1} \to q_{k_1}\alpha_{l_1}M_{s_1}$

$q_{i_2}\alpha_{j_2} \to q_{k_2}\alpha_{l_2}M_{s_2}$

...

$q_{i_m}\alpha_{j_m} \to q_{k_m}\alpha_{l_m}M_{s_m}$

Тогда номер машины:

$$n(T) = \prod_{i=1}^{m} p_{5t-4}^{i_t} p_{5t-3}^{j_t} p_{5t-2}^{k_t}\, p_{(5t-1)}^{l_t} p_{5t}^{s_t}$$

Где $i_t, k_t \in Q_0$; $j_t, l_t \in A_0$; $s_t \in \{1,2,3\}$

Нормальные алгорифмы Маркова

1 декабря 2017 г. 19:07

Есть алфавит.

$$\begin{cases} P_1 \to (\cdot) Q_1 \\ P_2 \to (\cdot) Q_2 \\ \quad\dots \\ P_k \to (\cdot) Q_k \end{cases}$$

$P_i -$ **слово**, $Q_i -$ **подстановка**.

Ищется первое вхождение P_i, которое заменяется на Q_i. После удачной замены $i = 1$. Алгоритм останавливается при команде с $(\cdot)$, или если ни одно правило не применимо.

Пример 1

$\Lambda -$ пустой символ

$\Lambda \to \cdot\, \Lambda$ - машина применима ко всему

$\Lambda \to \Lambda$ - машина не применима ни к чему

Пример 2 - инвертация булевого набора

$* 0 \to 1 *$

$* 1 \to 0 *$

$* \to \cdot\, \Lambda$

$\Lambda \to *$

Пример 3 - сортировка

$10 \to 01$

$20 \to 02$

$21 \to 12$

Пример 4 - проверка правильности скобок

$() \to \Lambda$

Если все правильно, то останется пустая строка

Подход порождающих грамматик

1 декабря 2017 г. 19:36

$A = \{\alpha_1, \dots, \varepsilon\}, \varepsilon -$ пустой символ
$A^* -$ все возможные цепоки, которые можно получить из A.
$A^+ \in A^* -$ множество непустых цепочек
Порождающая грамматика $G = \{V_N, V_T, p, S\}$
$V_N -$ **нетерминальные символы**
$V_T -$ **терминальные символы**, $V_N \cap V_T = \emptyset$
$s \in V_N -$ начальный символ
$P -$ **правило**, $P \subset (V_T \cup V_N)^+ \times (V_N \cup V_T)^*$
Любую цепочку из терминальных и нетерминальных символов можно заменить на другую (может быть, и простую)

$$\begin{cases} \alpha \to \beta_1 \\ \quad \dots \\ \alpha \to \beta_к \end{cases} = \alpha \to \beta_1 \,|\dots|\, \beta_к$$

- Несколько правил с одинаковыми правыми частями.

Из цепочки $\alpha \in (V_T \cup V_N)^+$ **непосредственное выводится** $\beta \in (V_T \cup V_N)^*$, если α представима в виде
$\alpha = \xi_1\gamma\xi_2, \beta = \xi_1\delta\xi_2$
$\xi_1, \xi_2, \delta \in (V_T \cup V_N)^*, \gamma \in (V_T \cup V_N)^+$
И правило вывода $\gamma \to \delta$ содержится в P
$\alpha \to \beta_1 \to \beta_2 \to \dots \to \beta = \alpha \Rightarrow \beta$ - из α выводится β.

Пример:
$G = \{\{a, b\}, \{A, S\}, p, S\}, \{a, b\} \in V_t, \{A, S\} \in V_n$
Обычно нетерминальные символы обозначаются большими буквами

$$P: \begin{cases} S \to aAb \\ aA \to aaAb \\ A \to \varepsilon \end{cases}$$

$S \to aAb \to aaAbb \to aaaAbbb \to aaabbb$
Здесь цепочка $aaAbb$ непосредственно выводима из aAb

$L(G) -$ **язык**, **порождаемый грамматикой G** $-$ множество цепочек терминальных символов, которые можно вывести из S. Так, язык из предыдущего примера:
$L(G) = \{a^n, b^n, n > 0\}$
$L(G_1) = L(G_2) -$ **эквивалентные грамматики**.
Пример эквивалентной грамматики:
$\tilde{G} = \{\{a, b\}, \{S\}, p, s\}$;
$P: S \to aSb|ab$
$L(G) = L(\tilde{G})$.

Классификация грамматик по Хомскому

Происходит по правилам вывода.

1. G_0 – **свободная/неограниченная**. Любая цепочка меняется на любую
2. G_1 – **контекстно-зависимая грамматика**
   $\xi_1 A \xi_2 \to \xi_1 \gamma \xi_2$
   A – нетерминальный $A \in V_N$
   $\gamma \in (V_N \cup V_T)^+; \xi_1, \xi_2 \in (V_N \cup V_T)^*$ - **контекст**
   Меняется нетерминальный символ А на цепочку, но только в условия контекста.
   Пример
   $G = \{\{a,b,c\},\{S,A,B\},P,s\}$
   $$P: \begin{cases} S \to aSBA|abA \\ AB \to BA \\ bb \to bb \\ bA \to bc \\ cA \to cc \end{cases} ; L(G) = \{a^n, b^n, c^n, n > 0\}$$
3. G_2 – **контекстно-свободная грамматика**
   $A \to \gamma$
   $A \in V_N; \gamma \in (V_N \cup V_T)^+$
   Нетерминальный символ меняется на цепочку.
   Пример
   $G = \{\{a_1, \dots, a_k, b_1, \dots, b_k\},\{s\},P,s\}$
   Правило:
   $s \to \varepsilon$
   $S \to a_1Sb_1 \mid a_2Sb_2 \mid \dots \mid a_kSb_k|$ – язык правильных скобочных выражений
4. G_3 – **регулярная/автоматная грамматика**
   a. $A \to a|Ba$ – левая рекурсивная/линейная
   b. $A \to a|aB$ – правая рекурсивная/линейная
   Пример
   $G = (\{a,b\},\{S,A,B\},P,S\}$
   $P: S \to aA|bB, A \to aA|a, B \to b$
   $L(G) = \{a^n b^2 | n > 1\}$

Контекстно-свободные грамматики. Форма Бэкуса-Наура(BNF)

$\alpha \to \beta_1|\beta_2| \dots$
$\alpha ::= \beta_1|\beta_2| \dots$
$a + b/c * d - e \dots$
$V_T = \{a, b, \dots, z, +, -, *, /\}$
$V_N = \{<\text{бесскобочное выражение}>, <\text{буква}>, <\text{знак операций}>\}$
$S = <\text{БВ}>$
Пример
```
<БВ>::=<БВ><знак операции><БВ>|<буква>
<буква>::=<a,b,…,z>
<знак операции>::=<+,-,*,/>
```
Грамматический разбор

$s \to \beta_1 \to \beta_2 \to \cdots \to \alpha$. На каждом шаге один нетерминальный символ заменяется на цепочку

Пример на основе предыдущего

$a + b * d$

```
<БВ>
<БВ><з.о.><БВ>
<буква><з.о.><БВ>
a<з.о.><БВ>
a+<БВ>
a+<БВ><з.о.><БВ>
```

…

Грамматика **неоднозначна**, т.е. на одном шаге можно применить несколько правил. Это приводит к тупикам и бэктрэкингу.

Введем понятие $\boldsymbol{L(B)}$ – множество терминальных символов, которых можно получить после применения правила.

Условие однозначности - $L(B_i) \cap L(B_j) = \emptyset$

Пример

В рассмотренном примере есть пересечение:

```
L(<БВ>)={a,b,…
L(<буква>)={a,b,…
```

Модификация:

```
<БВ>::=<буква><окончание>
<окончание>::=<з.о.><БВ>|ε
```

Алгоритмы преобразования G_2.

Алгоритм **непродуктивный**, если он не порождает ни одной терминальной цепочки

Поиск непродуктивного

$\mathrm{A} \Rightarrow \alpha; \alpha \in V_T^+$

$w_0 = \emptyset$

$w_1 = \{A | A \to \alpha \in P, \alpha \in V_T^+\}$

$k = 1$

While $(W_k \neq W_{k-1})${

$\quad W_{k+1} = W_k \cup \{B | B \to \beta \in \rho, \beta \in (V_T \cup W_k)^+\}$

$\quad$ k=k+1

}

- Цепочка на выходе даст или терминал, или продуктивные нетерминальные символы

Пример

$s \to sA|Bsb|bAb$

$A \to aSa|bb$

$B \to bBb|BaA$

$w_1 = \{A\}$

$w_2 = \{A, S\} \Rightarrow$ B - непродуктивный

Недостижимые символы

$S \to aSb|ba$
$A \to aAa$
Символ A недостижим, хоть и продуктивен. Алгоритм удаления:
$w_0 = \emptyset$
$w_1 = S$
$k = 1$
While $(w_k \neq w_{k-1})$
$$w_{k+1} = w_k \cup \{B|a \to \alpha B\beta, a \in W_k\}$$

Т.е. к множеству W_{k+1} добавляются те нетерминалы, в которые можно попасть из K.

Пример:
$S \to abS|Asa|ab$
$A \to abAa|ab$
$B \to bAab|bB$
$w_1 = \{S\}$
$w_2 = \{S, A\}$
$w_3 = w_2 \Rightarrow B -$ лишний

Польская запись

```
<постфикс>::=<постфикс><постфикс><знак операции>|<буква>
<буква>::={a,b,…,z}
<знак операции>::= {+,-,*,/}
<префикс>::=<знак операции><префикс><преФикс>|<буква>
```

Автоматы. Автоматы Мили и Мура

14 января 2018 г. 15:00

Автоматы Мили (1-го рода)

$\mathbb{A}::=\langle Q,\Sigma,\Omega,\delta,\Lambda,q_0\rangle$

Q - **множество состояний автомата**

Σ - **входной алфавит**

Ω - **выходной алфавит**

$\delta: Q \vee E \rightarrow Q$ – **функция перехода**

$\Lambda: Q \vee \Sigma \rightarrow Q$ – **функция выхода**

$q_0 \in Q$ – **начальное состояние**

Обычно автомат представляется графом или таблицей.

Пример 1

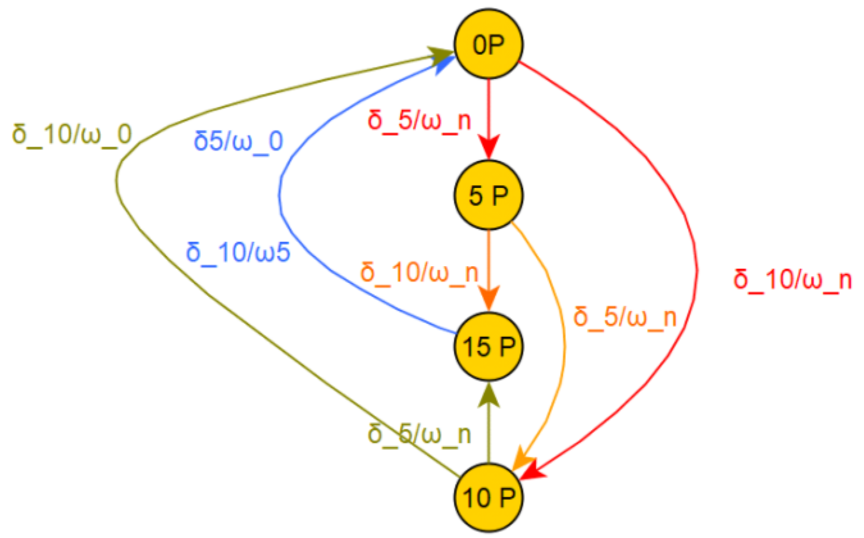
Автомат принимает монеты по 5,10₽. Шоколадка стоит 20₽.

$Q: q_0, q_5, q_{10}, q_{15}$ – множество состояний

δ_5, δ_{10} – функции перехода

$\Omega: \omega_0, \omega_5, \omega_n$

ω_n – ничего не дать, ω_0 – дать шоколадку, ω_5 – дать шоколадку и 5₽



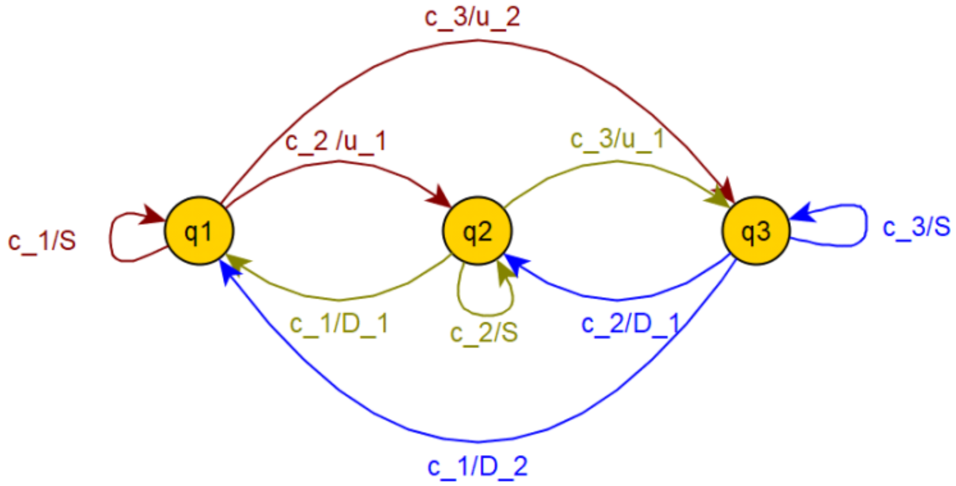
	δ_5	δ_{10}
q_0	q_5/ω_n	q_{10}/ω_n
q_5	q_{10}/ω_n	q_{15}/ω_n
q_{10}	q_{15}/ω_n	q_0/ω_0
q_{15}	q_0/ω_0	q_0/ω_5

Пример 2

Есть 3-х этажный дом с лифтом

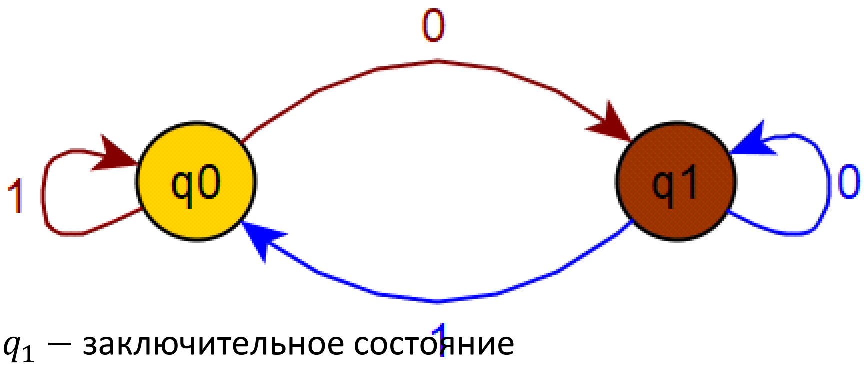
$\Sigma: c_1, c_2, c_3$

$\Omega: u_1, u_2, d_1, d_2, S$
$Q: q_1, q_2, q_3$



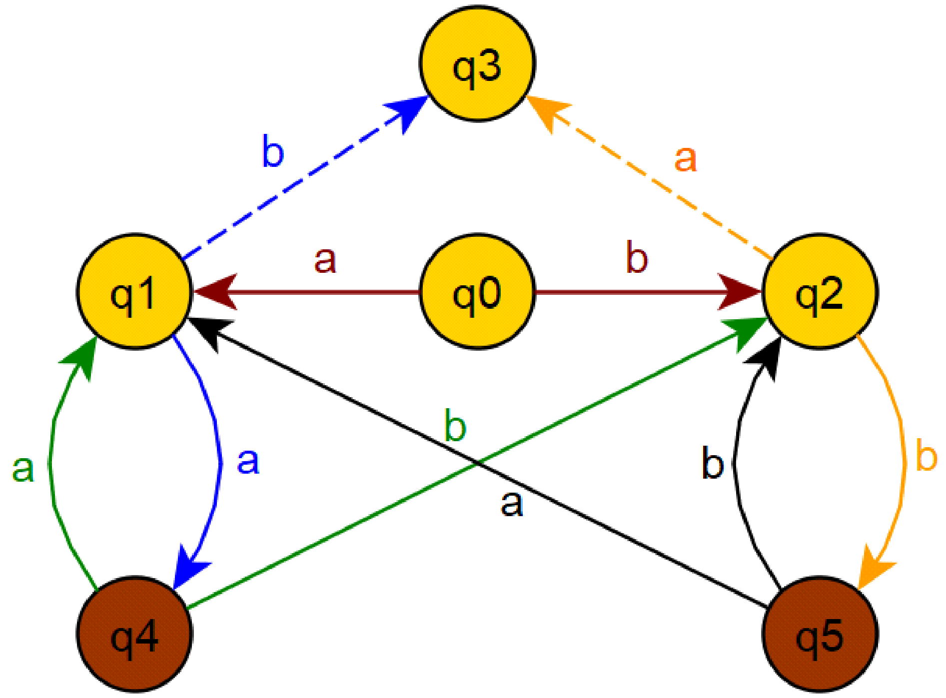
Автоматы Мура (2-го рода)
$\Lambda: Q \rightarrow \Omega.$
В этом автомате выходной символ определяется только состоянием.
Автомат-распознаватель - частный случай автомата Мура, если выходное множество содержит 2 элемента. Такой автомат используется для распознавания принадлежности цепочки символов к языку
A::=<Q, Σ, δ, q_0, F>
$F \subseteq Q$ — заключительное состояние.

Пример 1
Язык состоит из четных двоичных чисел.



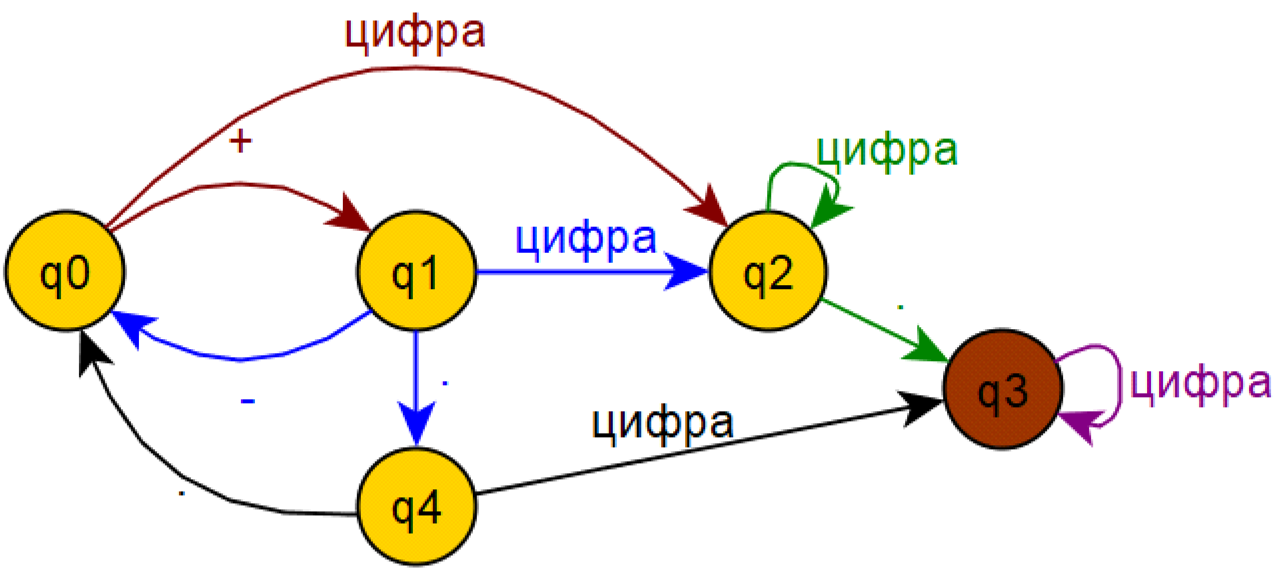
q_1 — заключительное состояние

Пример 2
$\Sigma: a, b$. Все символы должны встречаться по 2 раза подряд

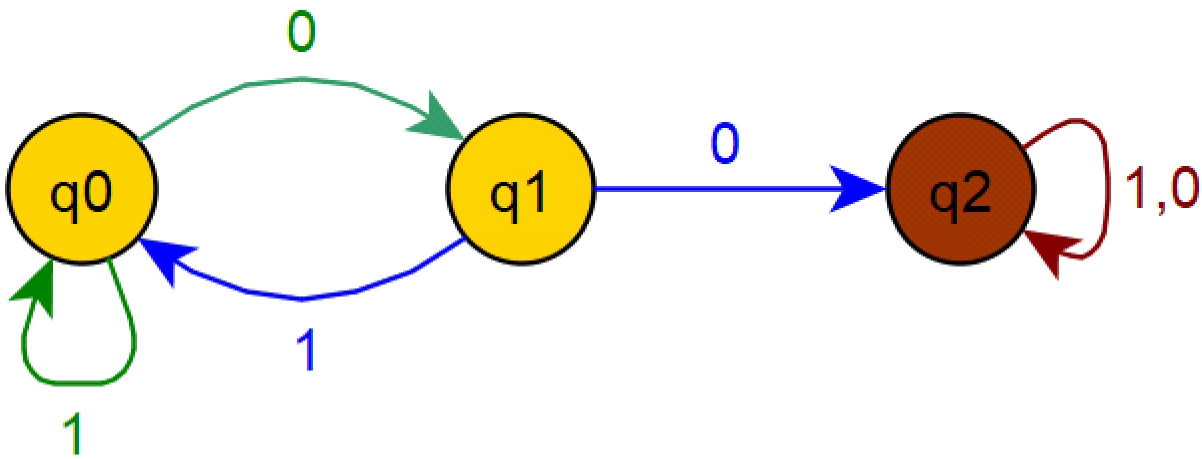


q_3 – **невозвратное состояние**. Обычно невозвратные состояния не рисуются, а автомат прекращает работу, если нет выхода.

Пример 3
$\Sigma = \{+, -, 0,1,2,3,4,5,6,7,8,9, .\}$
Слова вида $\mathrm{AA}.\,|\ldots.\ldots|\,.\mathrm{BB}\ldots$



Пример 4
Двоичная цепочка должна содержать два нуля подряд


0
q0
q1
0
q2
1,0
1
1

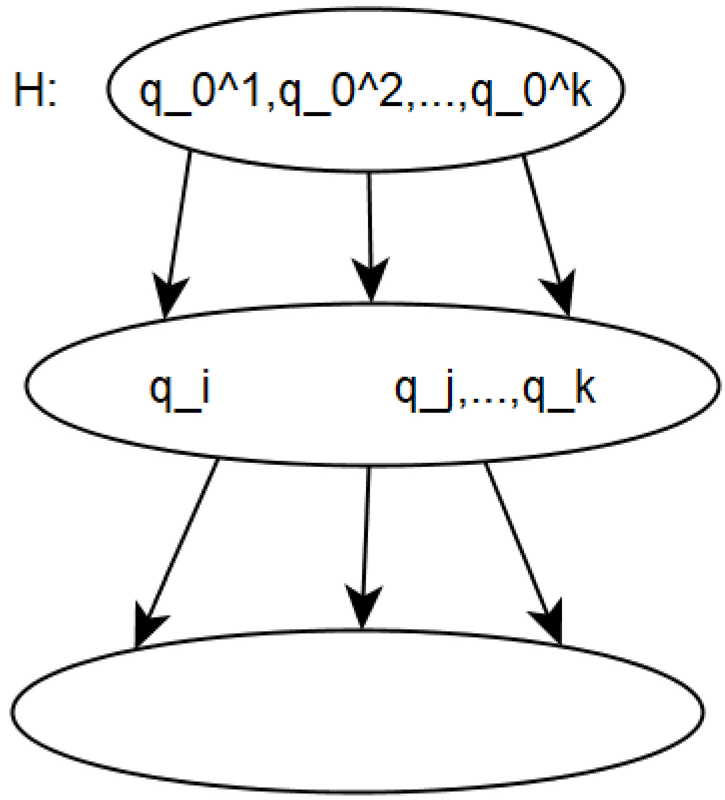
Детерминизация

14 января 2018 г. 16:41

A::=<Q,Σ,δ,H,F>
Q — **множество состояний**
δ — функция перехода $Q \times \Sigma \to 2^{Q}$
$H \subseteq Q$ — **подмножество начальных состояний**
$F \subseteq Q$ — **заключительные состояния**
В отличие от детерминированного автомата, в этом переходы неоднозначны.

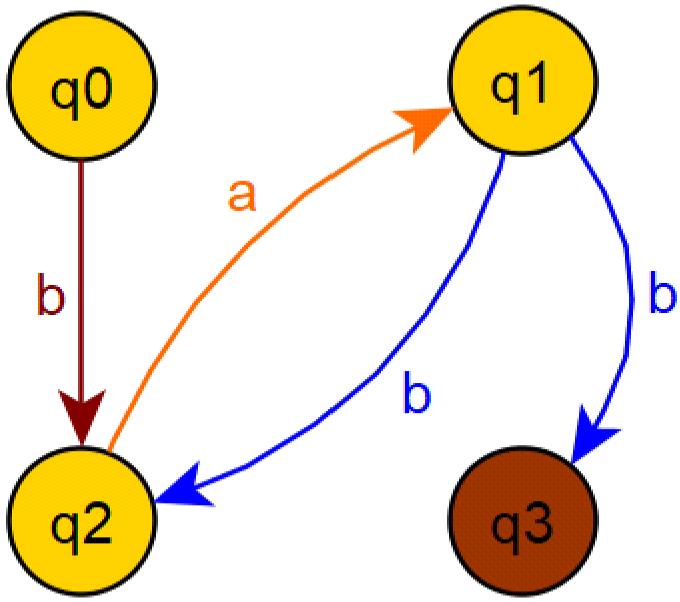
Нахождение недостижимых состояний
$L \leftarrow q_0$
$L = \{\dots\}$ — перебор всех состояний и определение, куда можно попасть. Если встречается новое состояние, оно добавляется в L. Когда L перестает расширяется, алгоритм останавливается.

Детерминизация
$\widetilde{q_0} = H$ — все начальные состояния объединяются в одно
$q_0^1, q_0^2, \dots, q_0^k$



Новое состояние - подмножество старых. Алгоритм работает до тех пор, пока появляются новые состояния. У нового графа может быть 2^{Q} состояний, что приводит к серьезному разрастанию графа

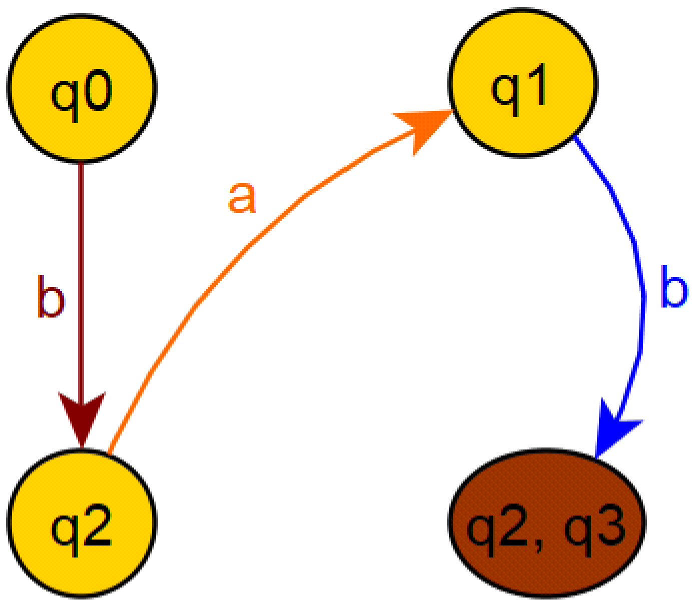
Пример



Данный автомат недетерминирован - из состояния q_1 по b можно перейти в q_2, q_3.

$$\delta(q_0, b) = \begin{matrix} \{q_0\} \\ \{q_2\} \\ \{q_1\} \\ \{q_2, q_3\} \end{matrix}$$

Новый автомат:



Теорема о разрастании для автоматных языков (лемма о накачке)

$L -$ бекспонечный автоматный язык

$L \Rightarrow \exists n$

$\omega \in L; |\omega| > n$

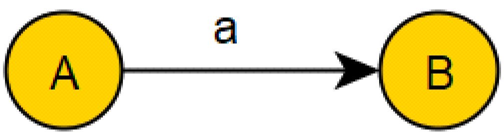
$\omega = \delta_1\delta_2\delta_3$

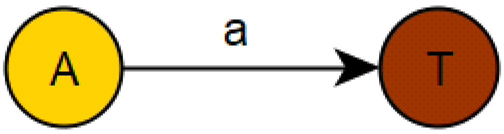
$|\delta_1\delta_2| \leq n$

$|\delta_2| > 0$

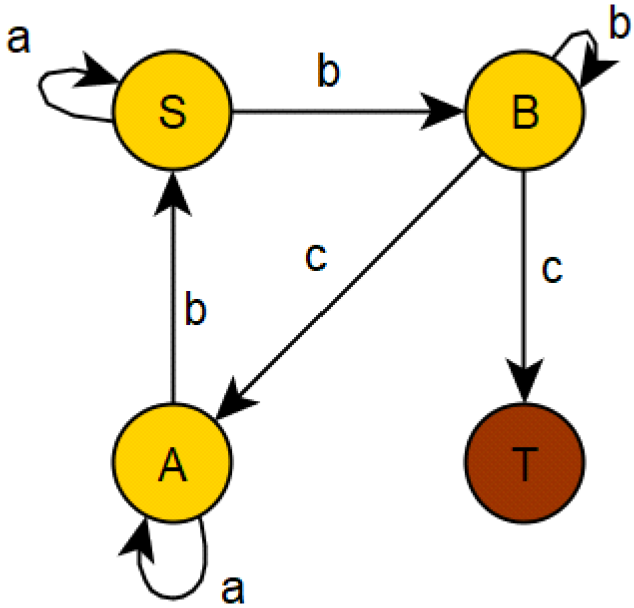
$m \geq 0 \; \delta_1 \delta_2^m \delta_3 \in L$
Любое слово языка длиннее n может быть представлено как сумма трех частей
$|\mathrm{G}| > \mathrm{n}$
$\omega = \omega_1, \omega_2 \ldots \omega_k, k > n -$ слово длиннее n.
Так как $k > n$, одно слово стретиться хотя бы 2 раза

Связь автоматов и автоматной грамматики
Порождающая грамматика G называется эквивалетной автомату A, если $L(G) = L(A)$, т.е. грамматика распознается автоматом.
Для каждого нетерминального символа есть вершина
$A \to aB|a$





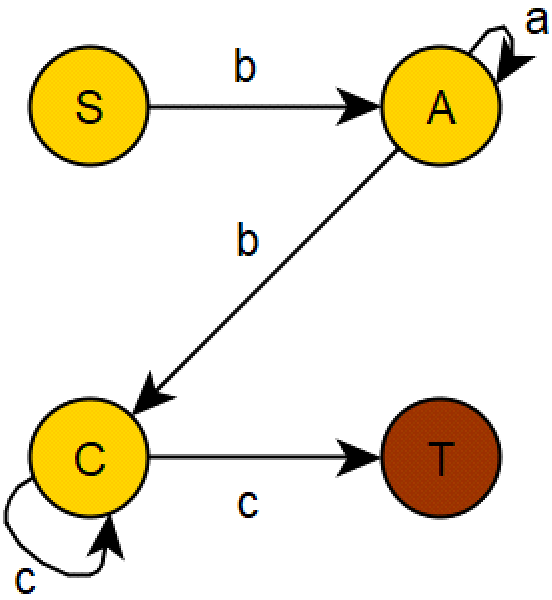
Пример
$S \to aS|bB$
$A \to aA|bS$
$B \to bB|c|cA$
Задача - построить эквивалентный автомат



Основной недостаток - как правило, получаются недетерминированные автоматы

Пример 2
$S \to aA$

$A \to aA|bC$
$C \to cC|c$



Возможна и обратная задача - построение грамматики по автомату

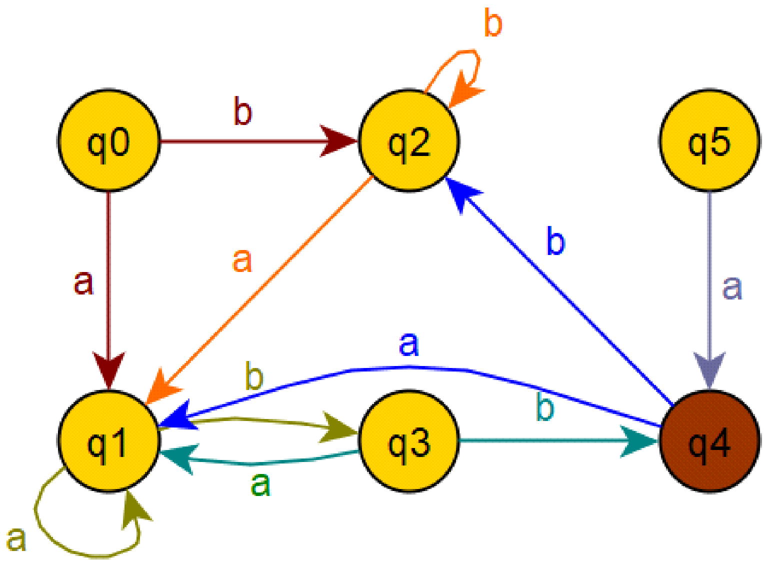
Минимизация. Конечные автоматы

14 января 2018 г. 19:06

Q — множество состояний
A — алфавит
$q_i \equiv_0 q_j$ — элементы **0-эквиваленты,** если они оба заключительны или оба не заключительны
$q_i \equiv_k q_j$ — элементы **k-эквиваленты,** если они оба были эквивалентны на предыдущем шаге или $\left(q_i \equiv_{k-1} q_j\right); \delta(q_i, \alpha) \equiv_{k-1} \delta(q_j, \alpha)$

Инициализация - из автоматов удаляются все недостижимые состояния
Шаг 1. $S_1 = f; S_2 = Q - f$
S_1 — финальные состояния, S_2 — нефинальные
Шаг n. $S_{i1}, S_{i2}, \dots, S_{ik}$ — классы эквивалетности $n-1$.
Попытка разбить по n — эквивалетности. Если хотя бы одно множество разбивается, процесс продолжается.

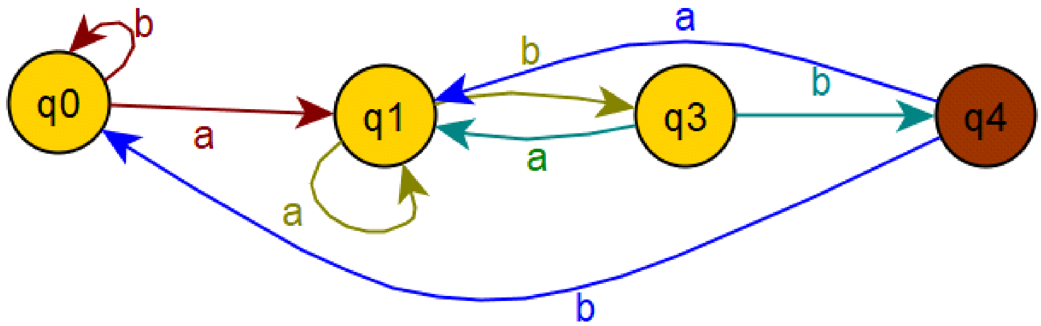
Пример



-1. $\{q_0\}, \{q_0, q_1, q_2\}, \{q_0, q_1, q_2, q_3\}, \{q_0, q_1, q_2, q_3, q_4\} \Rightarrow q_5$ — недостижимое состояние

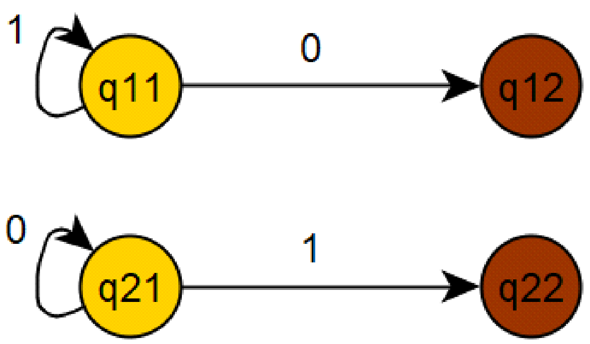
0. $\{q_4\}, \{q_0, q_1, q_2, q_3\}$
1. $\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}$
2. $\{q_4\}, \{q_3\}, \{q_0, q_2\}, \{q_1\}$

Сокращенный автомат:

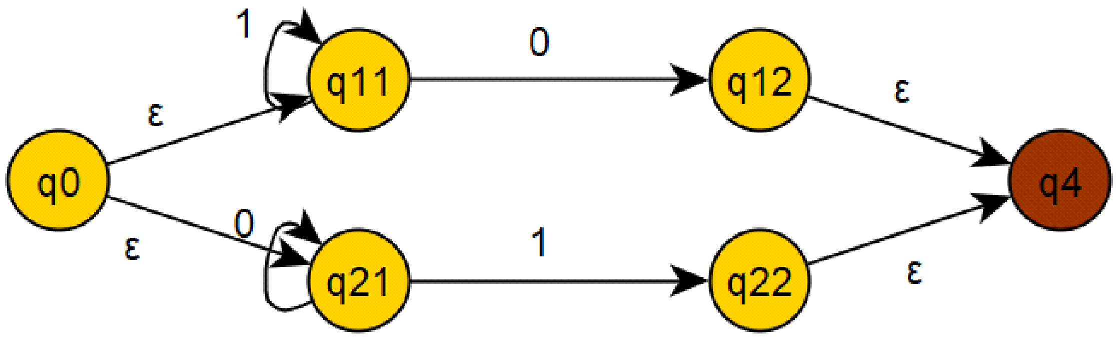


Конечные автоматы с ε —переходами
ε —переход - переход, которыйможет быть выполнен без входного

символа. Они обычно используются для объединения автоматов



Объединение:



Регулярные выражения

$L_1 + L_2$ — **объединения** языков. Содержит все слова из L_1 и L_2

$L_1 \wedge L_2$ — **конкатенация/сцепление** языков. Содержит все слова вида $\omega_1\omega_2$.

$L_1 \wedge L_2 = \{\omega_1\omega_2 | w_1 \in L_1, \omega_2 \in L_2\}$

L^* — **итерация** языка. Содержит все слова, которые можно разбить на несколько подряд идущих слов этого языка

$(L)^* = \{\varepsilon| \cup \{\omega | (\exists n \geq 0)(\omega = \omega_1\omega_2 \dots \omega_k), \omega_i \in L\}$

Приоритеты в порядке убывания: итерация, конкатенация, объединение.

Так $(\left((1 \wedge 0) \wedge \left((1)^* + 0\right)\right) \equiv 10(1^* + 0)$

Определение регулярного выражения

1. ε (пустой символ) - регулярное выражение
2. $\forall a \in \Sigma$, a - регулярное выражение
3. Если r, p — регулярные выражения, то $(r + p), (rp), r^*$ - тоже.

Пример: $11(0 + 1)^*001$. В этом примере слово начинается на 11, в середине любая последовательность из 0 и 1, в конце - 001.

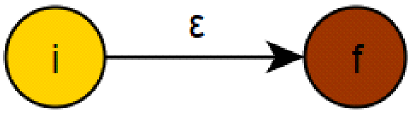
Свойства:

1. $r + p = p + r$
2. $(r + p) + q = r + (p + q)$
3. $(rp)q = r(pq)$
4. $(r^*)^* = r^*$
5. $(r + p)q = rq + pq$

Построение автомата

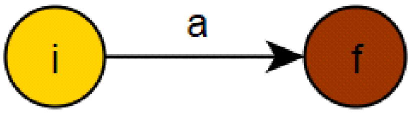
Для каждого регулярного выражения можно построить автомат.

1. Автомат, распознающий пустой символ, т.е. язык $L = \{\varepsilon\}$

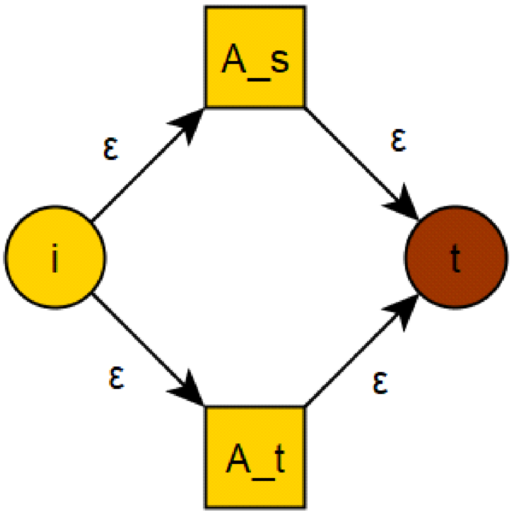


i, f — начальное и заключительное состояние

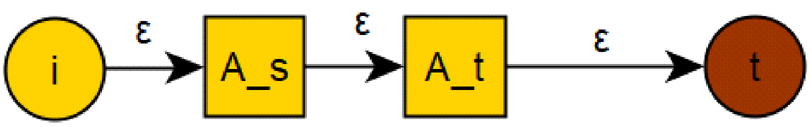
2. Автомат, распознающий один символ, т.е. язык $L = \{a\}$



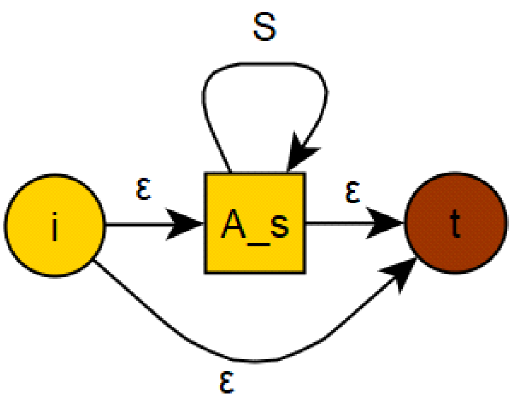
3. Пусть есть автоматы, распознающие языки s и t. Тогда автомат, распознающий язык $s + t$:



4. Автомат, распознающий язык st



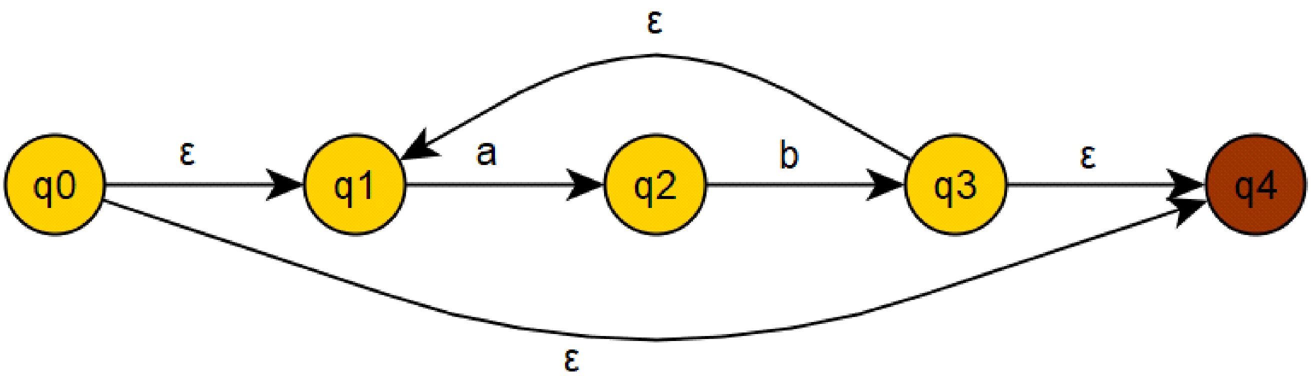
5. Автомат, распознающий язык s^*



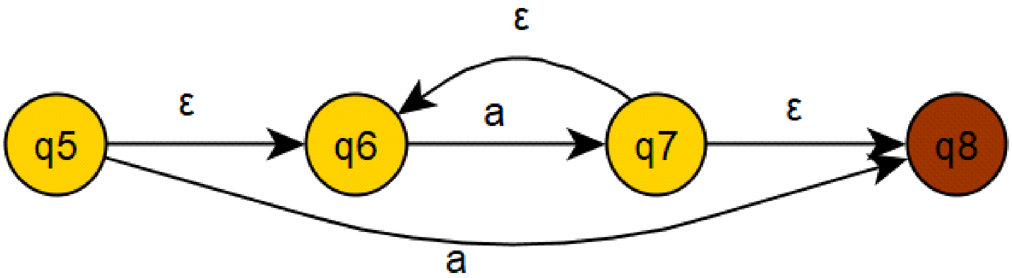
Пример

$r = (ab)^* a^* ba$

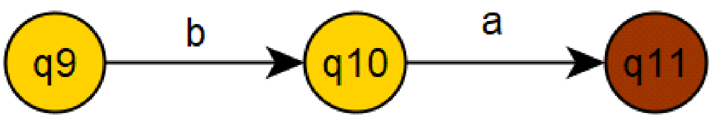
1. Итерация ab



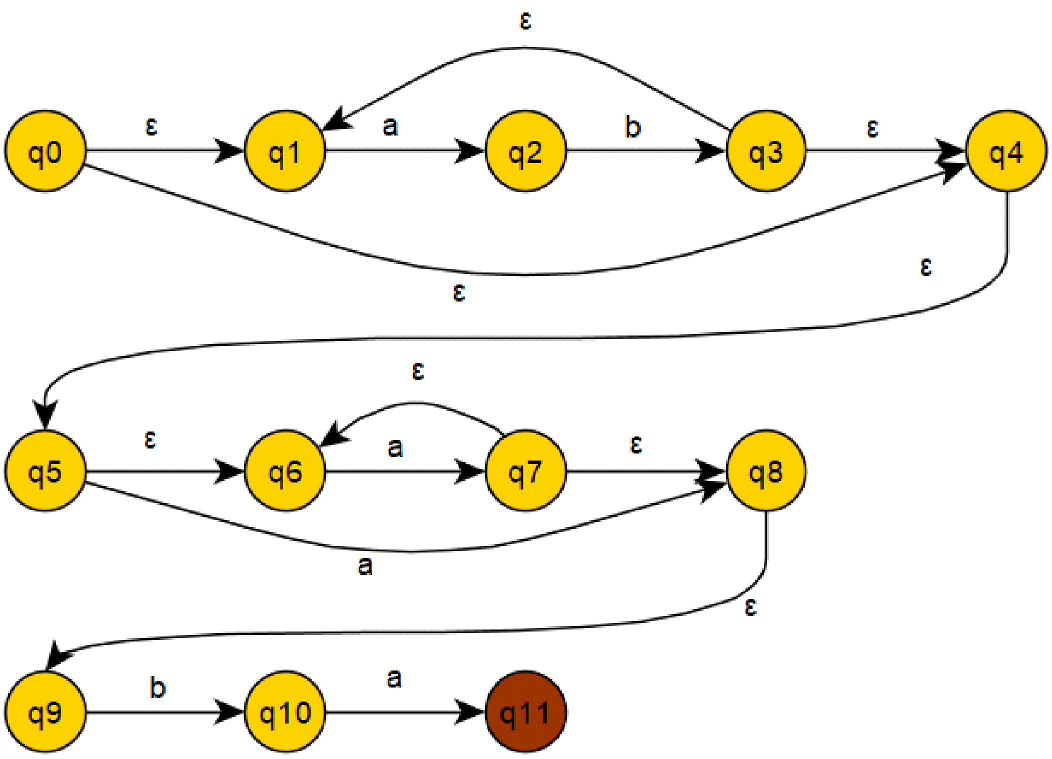
2. Итерация a



3. ba



И объединение этих автоматов:



Сложность алгоритмов

1 декабря 2017 г. 19:27

Подход 1. Классификация по сложности в зависимости от входных данных

$T(n) -$ функция, определяющая количество элементарных операций в зависимости от количества элементов

$T(n) \leq \alpha f(n), \alpha > 0$
$T(n) = O(f)$
$O(f) -$ верхняя оценка

1. $O(1) -$ не зависит от входных данных
2. $O(\log(n)) -$ например, быстрое возведение в степень. Характерно, когда алгоритм разбивается на несколько.
3. $O(n)$ - линейная сложность
4. $O(n\log(n)) \ -$ например, быстрое преобразование Фурье
5. $O(n^2), O(n^3) \dots, O(n^6) -$ полиномиальная сложность
6. $O(2^n) -$ экспоненциальная
7. $O(n!) -$ факториальная

1-5 считаются "хорошими", а 6-7 - "плохими".

Интересное. Пусть есть алгоритм со сложностью 7^n. Если увеличить производительность компьютера в 10 раз, то прирост решаемого $n - 4$. Если же улучшить алгоритм - 3^n - то производительность удвоится. Поэтому обычно эффективнее улучшать алгоритмы.

Подход 2. Классификация задач по сложности

$P -$ задача

1. $O\left(n^k\right); k \leq 6 -$ есть полиномиальный алгоритм, решающий задачу
2. $NP -$ есть полиномиальный алгоритм, проверяющий решение
3. $NPC - NP$-полная задача - задача, к которой сводятся все задачи класса NP.

"Задача тысячелетия" - $P = NP - ?$

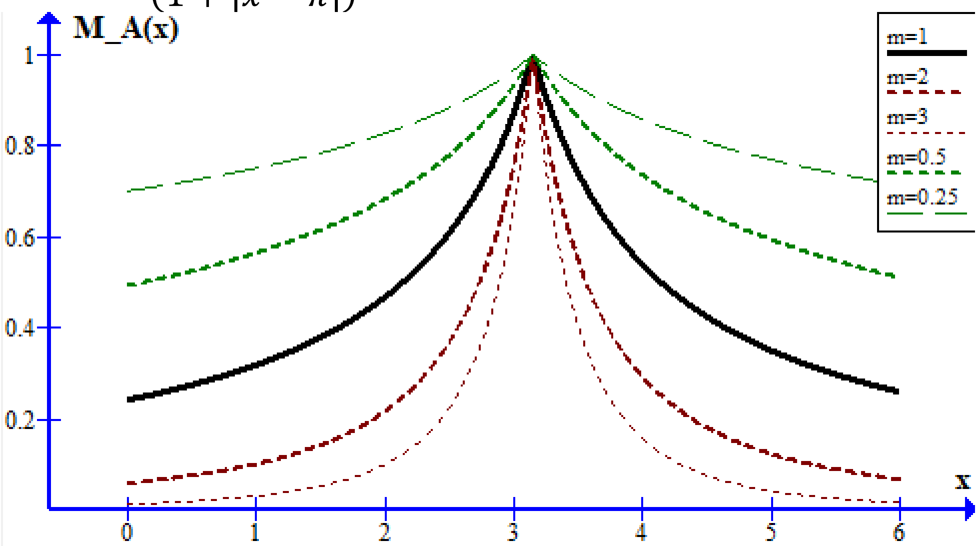
Нечёткие множества

14 января 2018 г. 19:26

Парадокс кучи. 1 зерно - не куча, 2 - не куча ⇒ кучи не существует?
$S(x)$ – зерно не образует кучи. $S(x) \to S(x+1) \Rightarrow \forall x\big(S(x)\big)$

M – **универсум**, множество, содержащее все $A \subseteq M$ – подмножества.
Для **чёткого** множества $M_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases}$
Для **нечёткого** множества $M_A(x)$ означает степень достоверности
$M_A(x) = \{(x, M_A(X)) | \forall x \in M\}$
Носитель - множество элементов x: $M_A(x) > 0$

Операции:
$M_{A \vee B} = \max\{M_A(x), M_B(x)\}$
$M_{A \wedge B} = \min\{M_A(x), M_B(x)\}$
$\bar{A} = M - A$
$M_{\bar{A}} = 1 - M_A(x)$
Для нечетких множеств неверны некоторые утверждения, верные для четких
$A \cap \bar{A} \neq \emptyset$
$A \cup \bar{A} \neq M$

Пример
$x \in [0,6]$
$$M_A(x) = \frac{1}{(1 + |x - \pi|)^m}; m \geq 0$$


Возведение в степень > 1 – **концентрация**
Возведение в степень < 1 – **размывание**

Мера нечеткости - чем больше непустоты в $A \cap \bar{A}$, тем более нечёткое множество

$$FUZ_p(A) = 1 - \frac{D_p(A, \bar{A})}{n^{\frac{1}{p}}}; p = 1,2, \ldots$$

$D_p -$ расстояние

Метрика Хеминга:

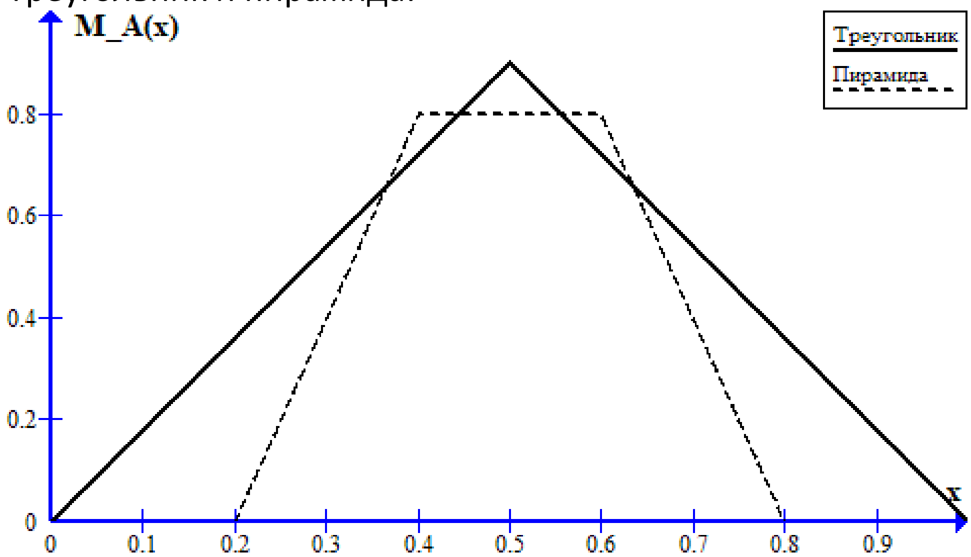
$$\mathrm{D}_1 = \sum_{i=1}^{n} |M_A(x_1) - M_A(x_2)|$$

Метрика Евклида

$$D_2 = \sqrt{\sum_{i=1}^{n} \left(M_A^2(x_1) - M_A^2(x_2)\right)}$$

Наиболее используемые функции принадлежности

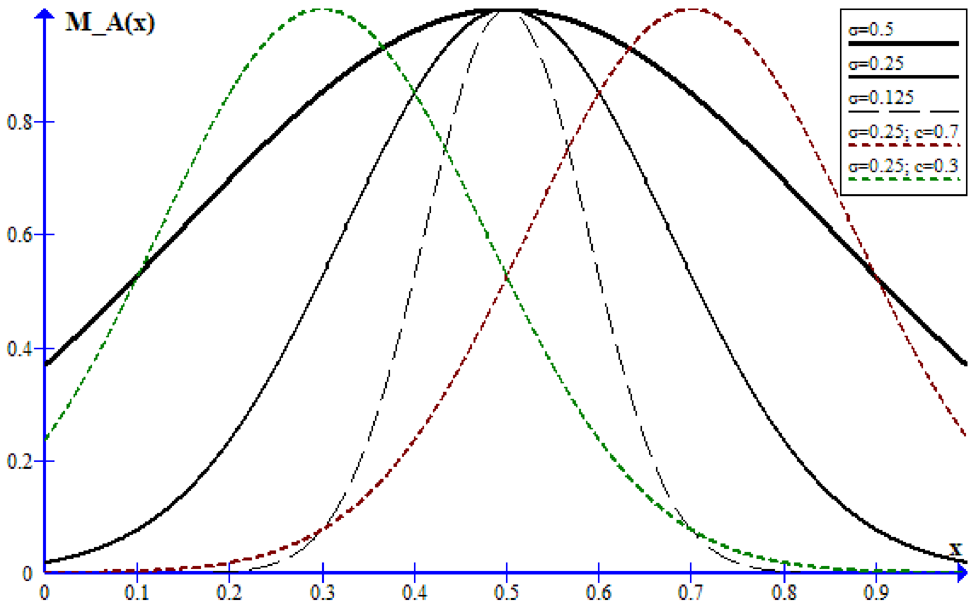
Треугольник и пирамида:



Экспонента:

$$M_A(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2}$$

$c -$ центр, $\sigma -$ крутизна


M_A(x)
σ=0.5
σ=0.25
σ=0.125
σ=0.25; c=0.7
σ=0.25; c=0.3
0.8
0.6
0.4
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
x