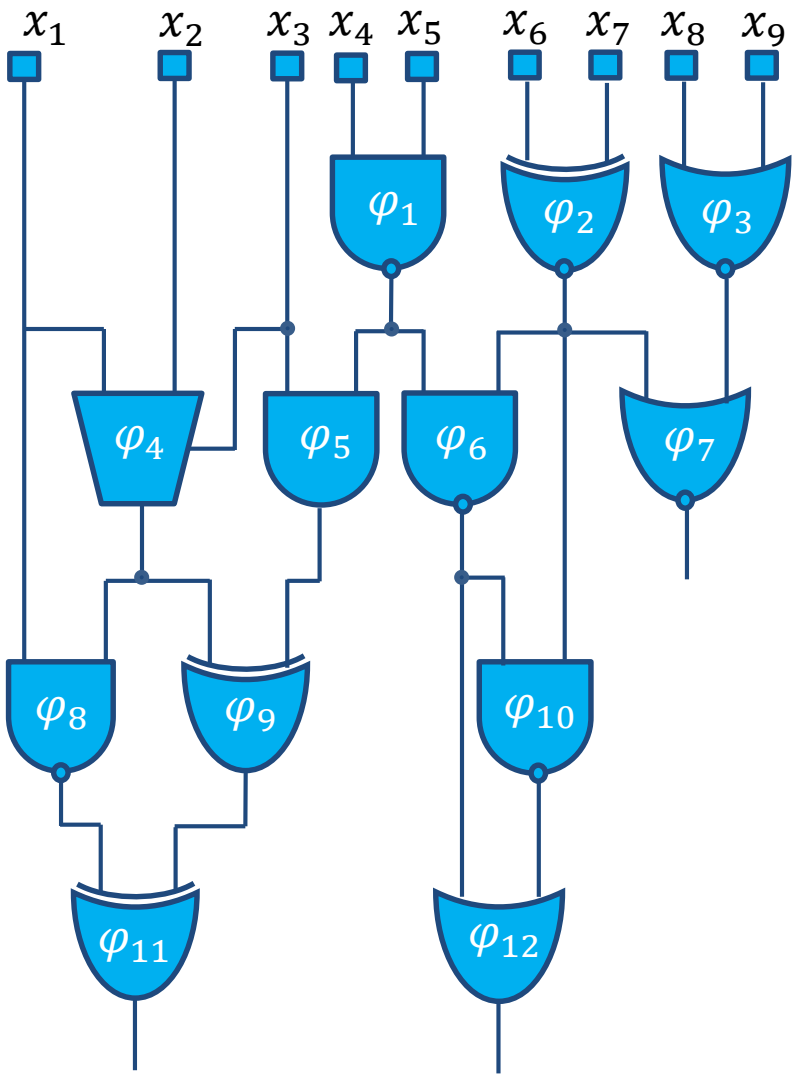
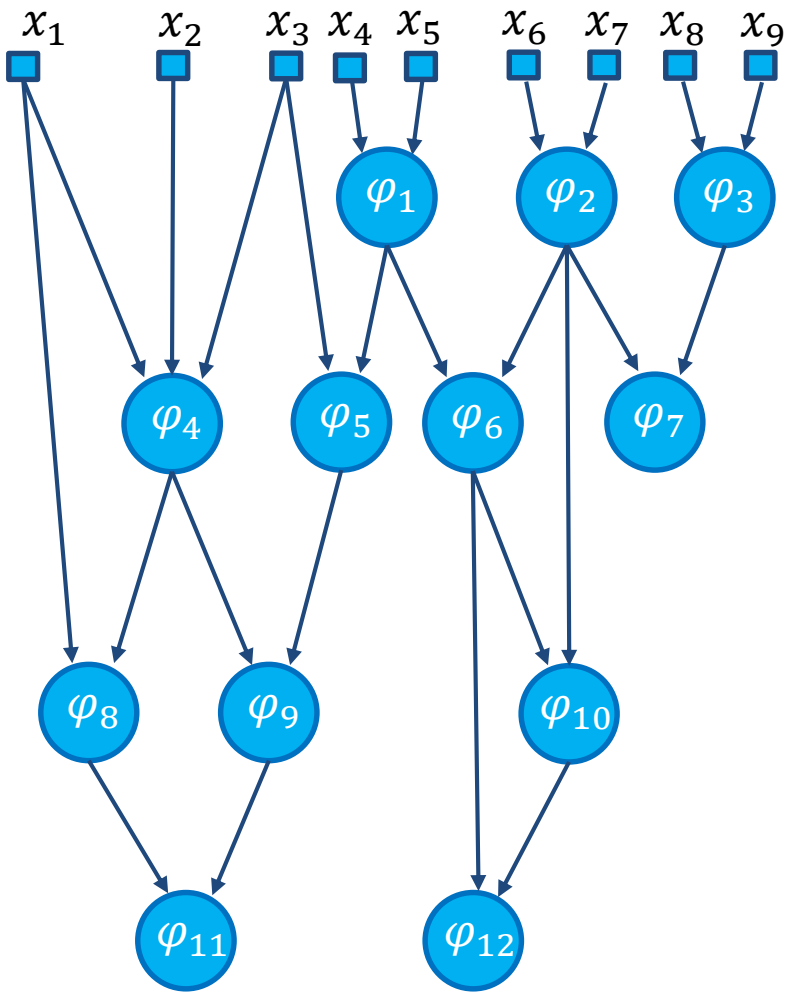
Введение в задачу привязки к библиотеке

Математические модели и методы логического синтеза СБИС

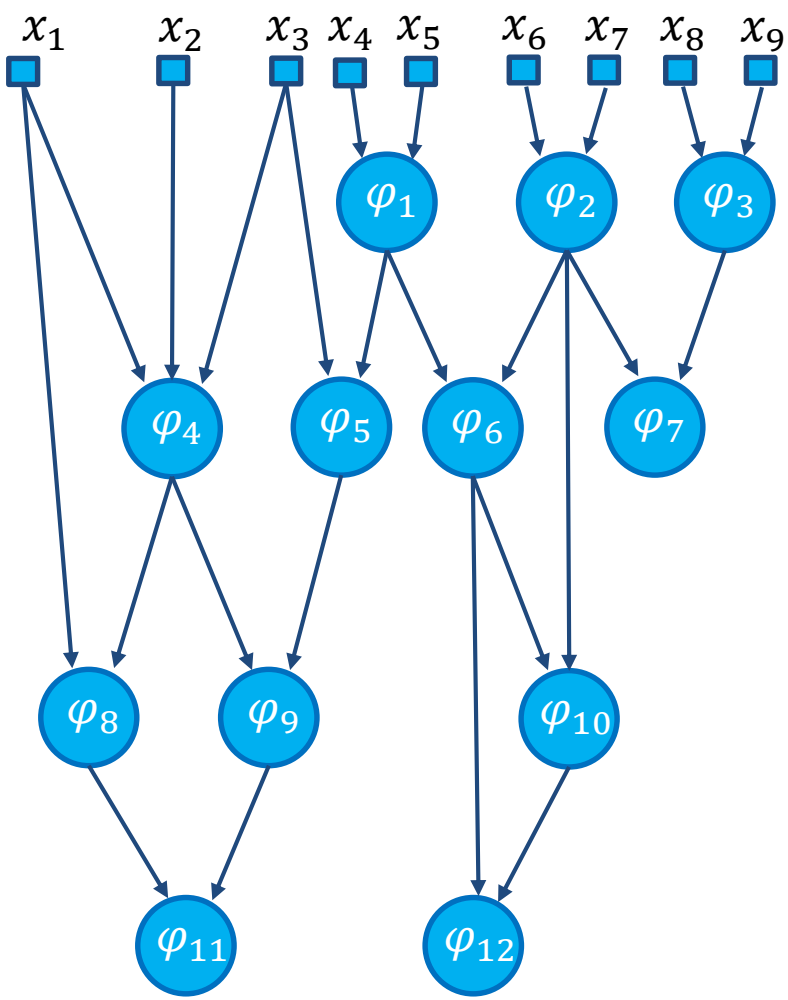
Логическая схема и ее граф





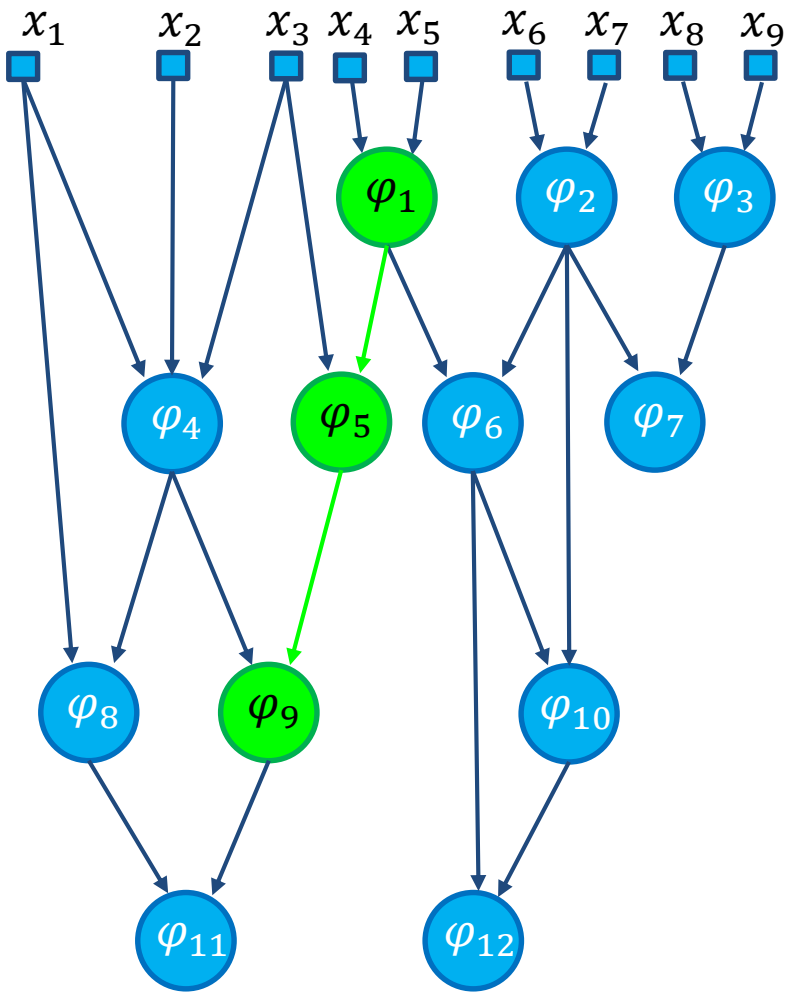
Конус вершины

- Конус C_φ вершины φ СФЭ Σ – произвольное множество отличных от входов Σ вершин, такое что для любой вершины $\varphi' \in C_\varphi$ существует ориентированный путь из φ' в φ, состоящий только из вершин, принадлежащих C_φ
- Вершина φ – корень конуса
- Множество i_{C_φ}, состоящее из всех вершин, имеющих хотя бы одно ребро, которое идет в вершину конуса C_φ, - множество входов C_φ
- $\left|C_\varphi\right|$ - вес конуса
- $\left|i_{C_\varphi}\right|$ - ширина конуса



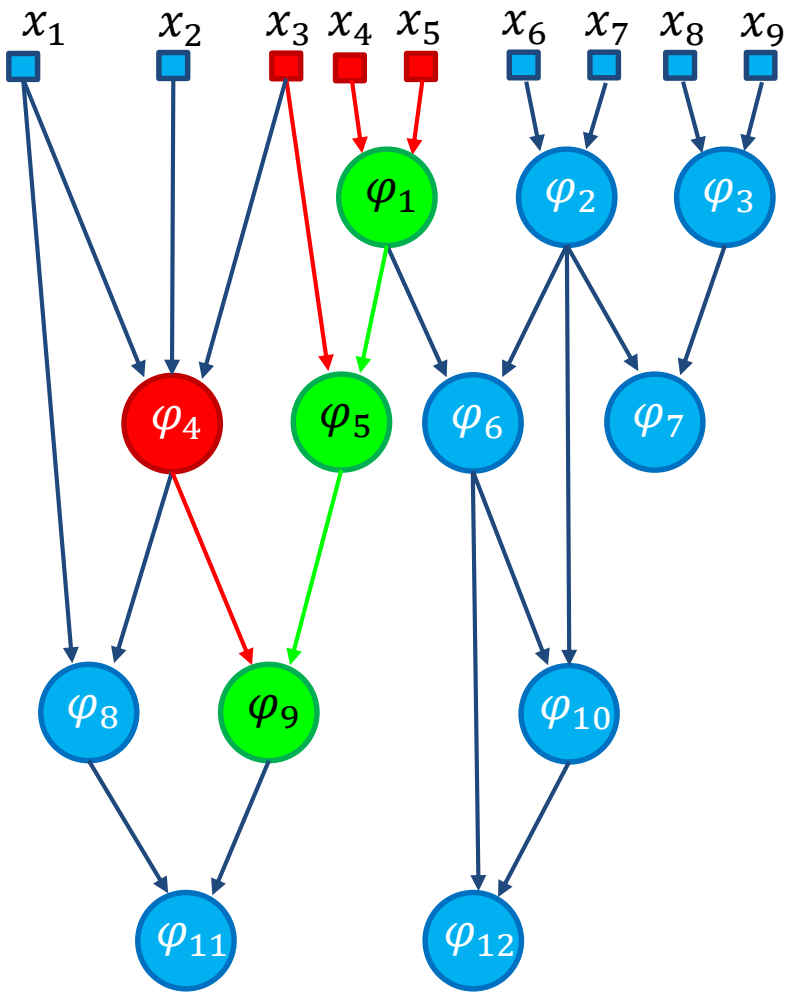
Конус вершины - пример

- Конус $C_{\varphi_9} = \{\varphi_1, \varphi_5, \varphi_9\}$ вершины φ_9
- $|C_\varphi| = 3$



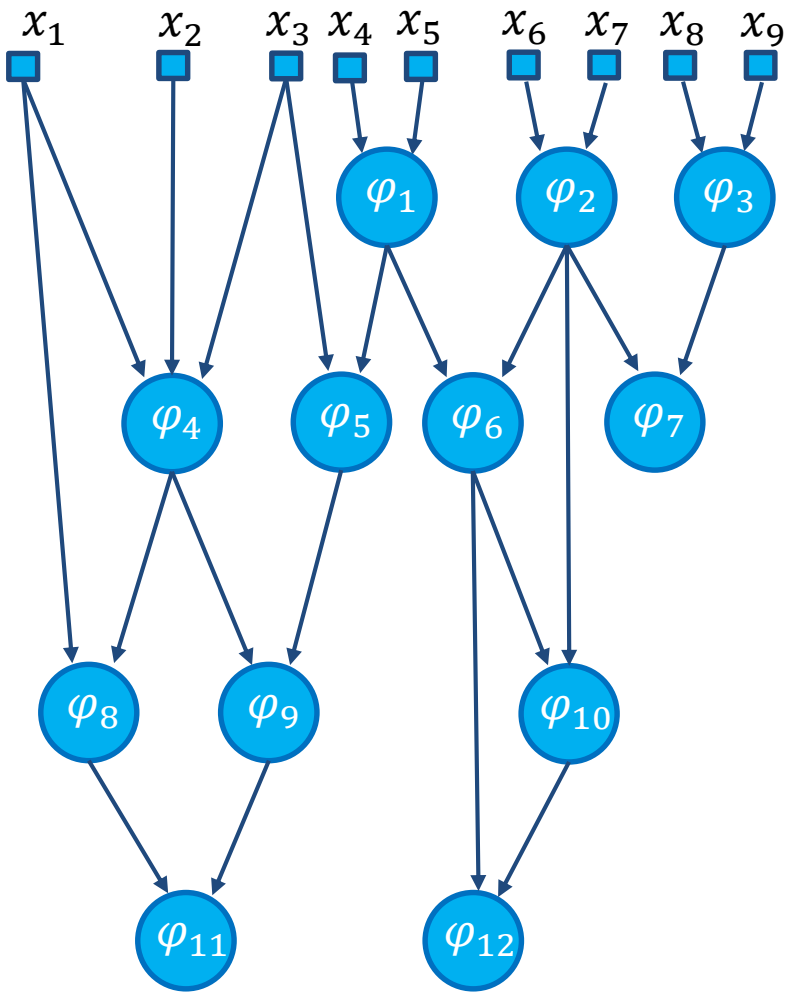
Конус вершины - пример

- Конус $C_{\varphi_9} = \{\varphi_1, \varphi_5, \varphi_9\}$ вершины φ_9
- $\left|C_{\varphi}\right| = 3$
- $i_{C_{\varphi}} = \{\varphi_4, x_3, x_4, x_5\}$
- $\left|i_{C_{\varphi}}\right| = 4$



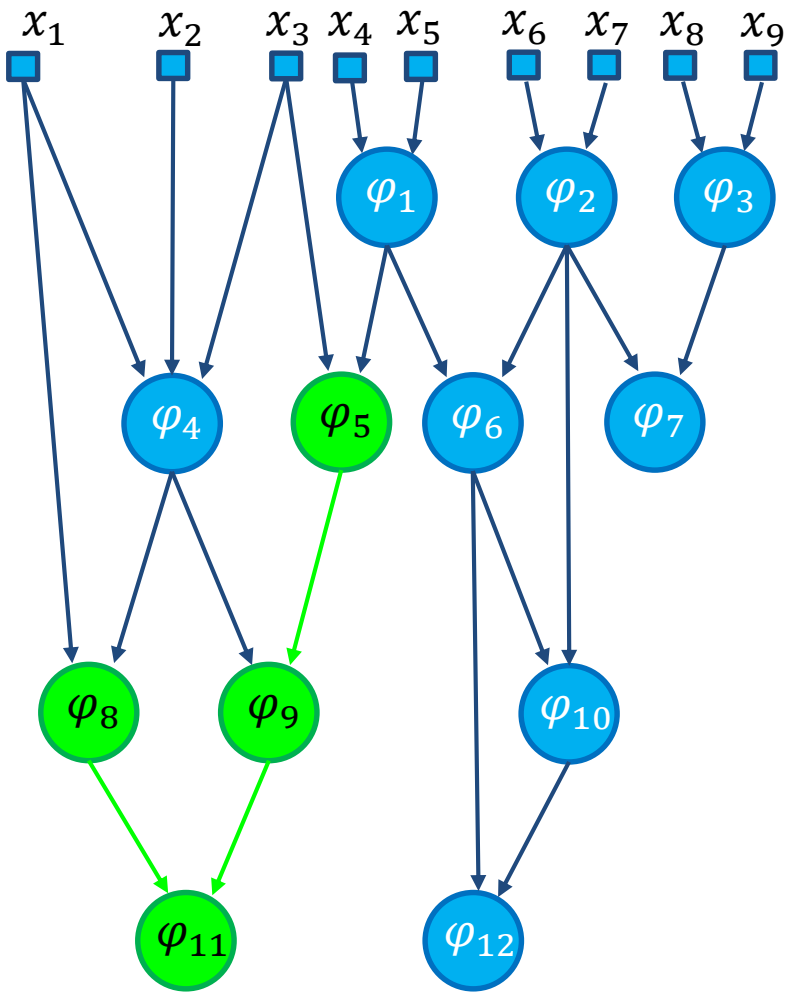
Специальные классы конусов вершин

- *Максимальное дерево* MT_φ вершины φ СФЭ Σ – максимальный по включению конус вершины φ, содержащий только вершины степень ветвления выхода которых равна 1.



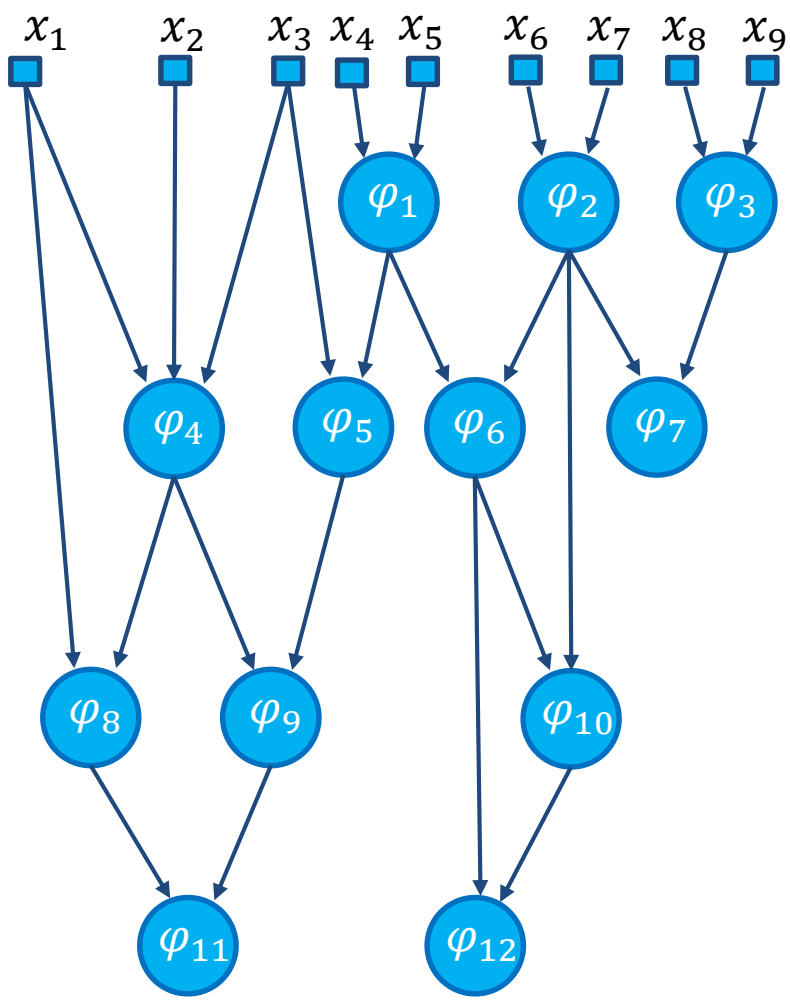
Специальные классы конусов вершин

- *Максимальное дерево* MT_φ вершины φ СФЭ Σ – максимальный по включению конус вершины φ, содержащий только вершины степень ветвления выхода которых равна 1.
- Пример: $MT_{\varphi_{11}} = \{\varphi_5, \varphi_8, \varphi_9, \varphi_{11}\}$



Специальные классы конусов вершин

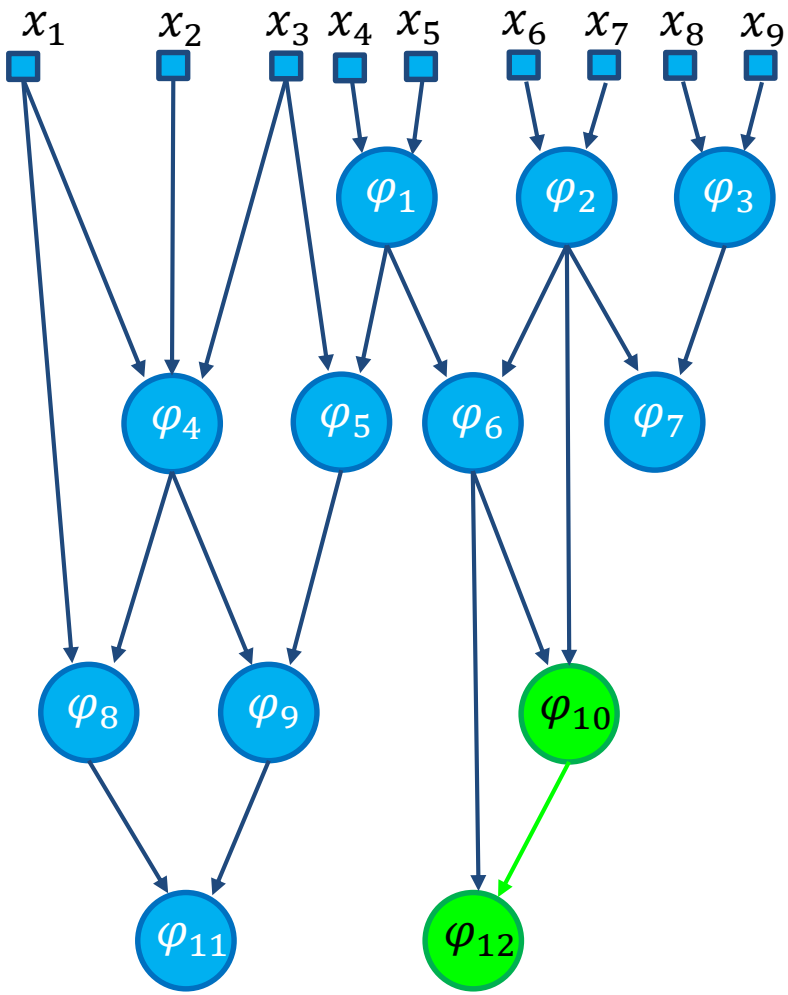
- *Замкнутый конус* FFC_φ вершины φ СФЭ Σ – конус вершины φ, такой что любое ребро, исходящее из вершины конуса, ведет в вершину, принадлежащую конусу.
- *Максимальный замкнутый конус* $MFFC_\varphi$ вершины φ СФЭ Σ – максимальный по включению замкнутый конус вершины φ



Специальные классы конусов вершин

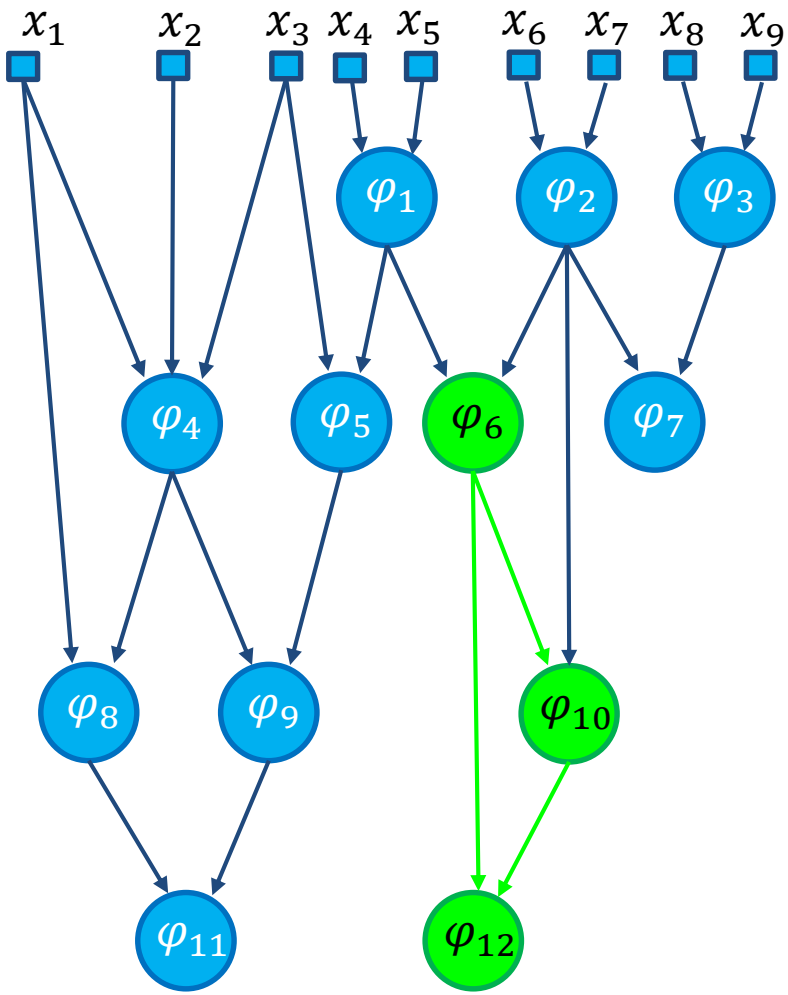
- Примеры:

$$FFC_{\varphi_{12}} = \{\varphi_{10}, \varphi_{12}\}$$



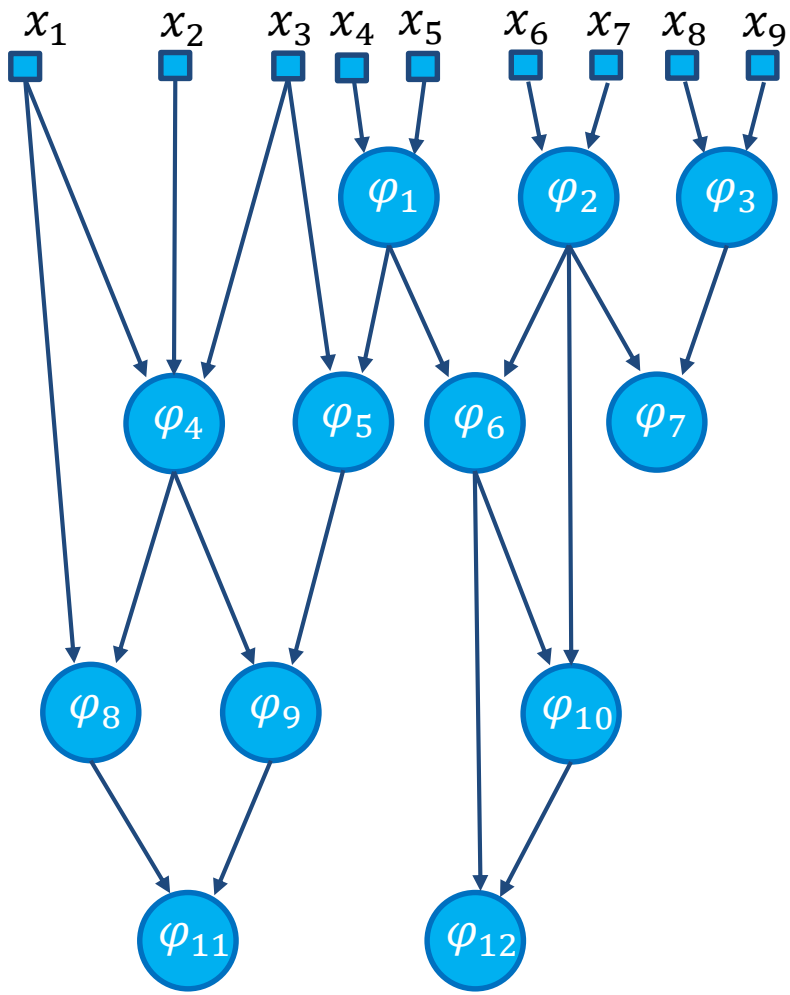
Специальные классы конусов вершин

- Примеры:
  $FFC_{\varphi_{12}} = \{\varphi_{10}, \varphi_{12}\}$
  $MFFC_{\varphi_{12}} = \{\varphi_6, \varphi_{10}, \varphi_{12}\}$



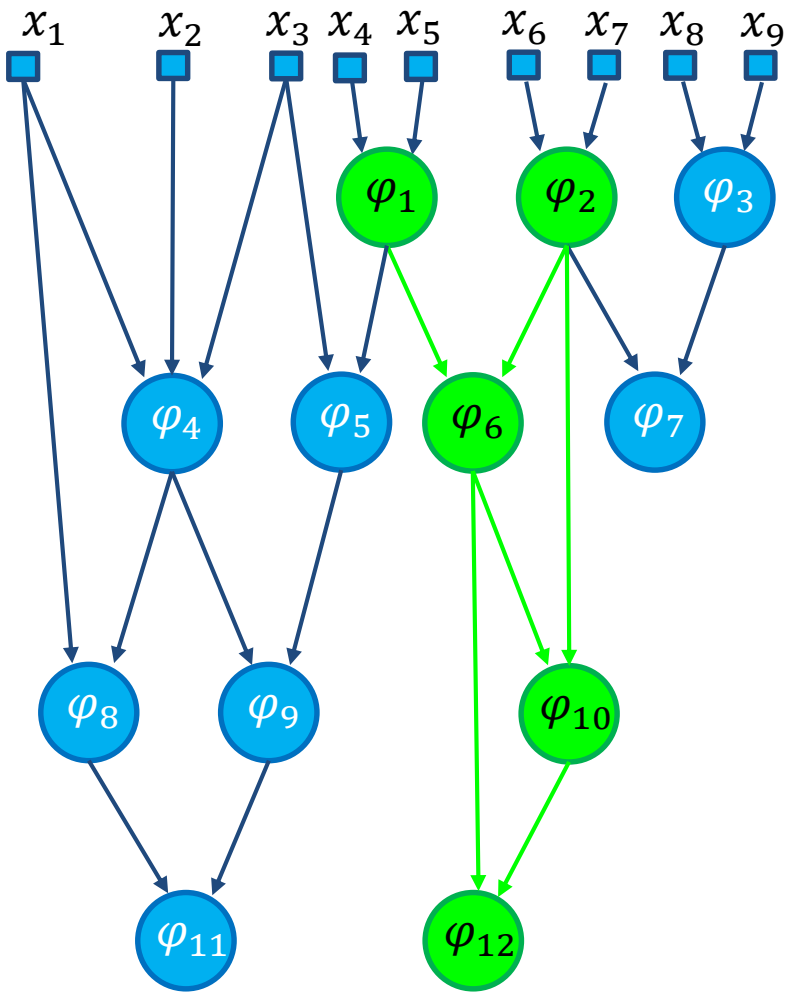
Специальные классы конусов вершин

- *Максимальный конус* MC_φ вершины φ СФЭ Σ – максимальный по включению вершин конус вершины φ
- Максимальный входной конус N_φ вершины φ СФЭ Σ – конус MC_φ, объединенный со всеми входами Σ, которые содержат хотя бы одно ребро, ведущее в MC_φ



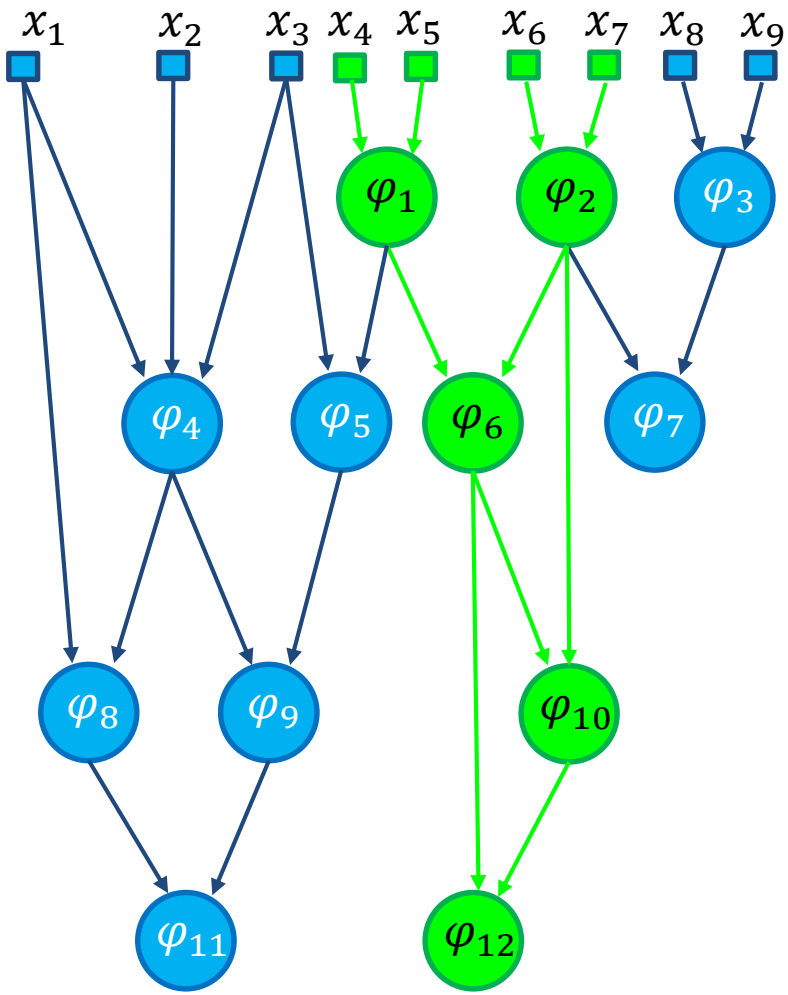
Специальные классы конусов вершин

- Примеры:
  $MC_{\varphi_{12}} = \{\varphi_1, \varphi_2, \varphi_6, \varphi_{10}, \varphi_{12}\}$



Специальные классы конусов вершин

- Примеры:
  $MC_{\varphi_{12}} = \{\varphi_1, \varphi_2, \varphi_6, \varphi_{10}, \varphi_{12}\}$
  $N_{\varphi_{12}} = MC_{\varphi_{12}} \cup \{x_4, x_5, x_6, x_7\}$



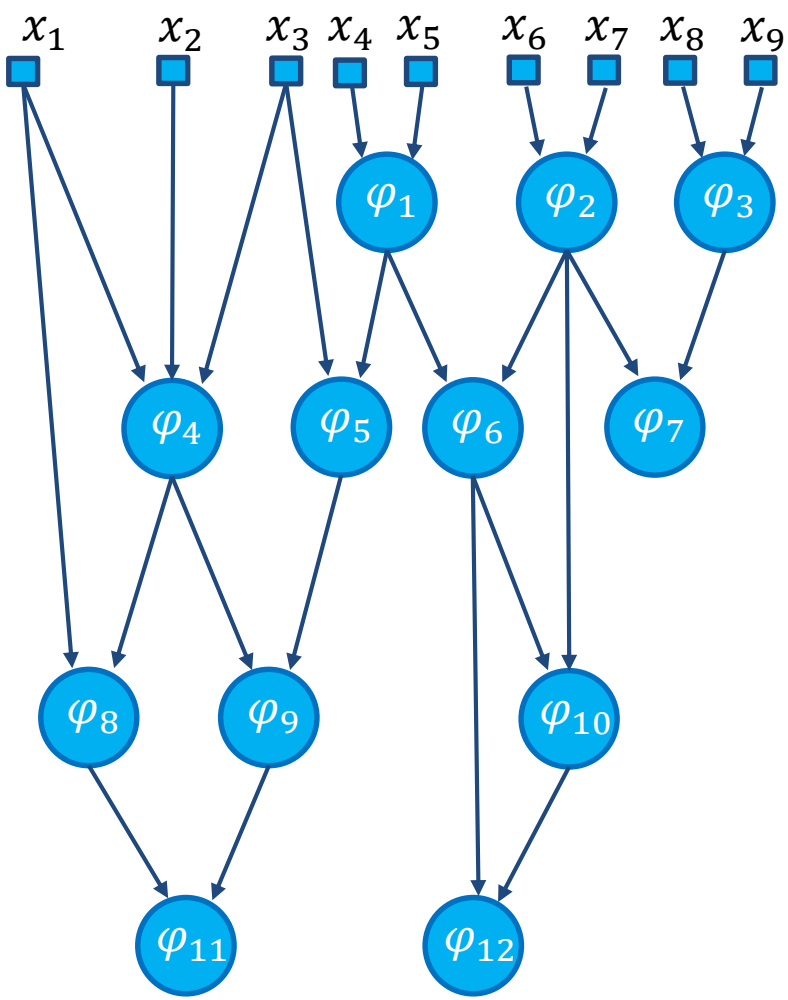
Свойства конусов вершин

- Для любой вершины φ СФЭ Σ существует единственное MT_φ и единственный $MFFC_\varphi$
- Для любой вершины φ СФЭ Σ выполняются соотношения:

$$MT_\varphi \subseteq MFFC_\varphi \subseteq MC_\varphi \subseteq N_\varphi$$

Покрытие СФЭ при помощи конусов

- Множество конусов $CS_\Sigma = \{C_{\varphi_{i_1}}, \dots, C_{\varphi_{i_k}}\}$ называется *покрытием* СФЭ Σ, если
  - $C_{\varphi_{i_1}} \cup \dots \cup C_{\varphi_{i_k}}$ совпадает с множеством всех вершин Σ, отличных от входов
  - Для любого $j, = 1, \dots, k$, каждый элемент $i_{C_{\varphi_j}}$ является либо входом Σ, либо корнем некоторого конуса из CS_Σ

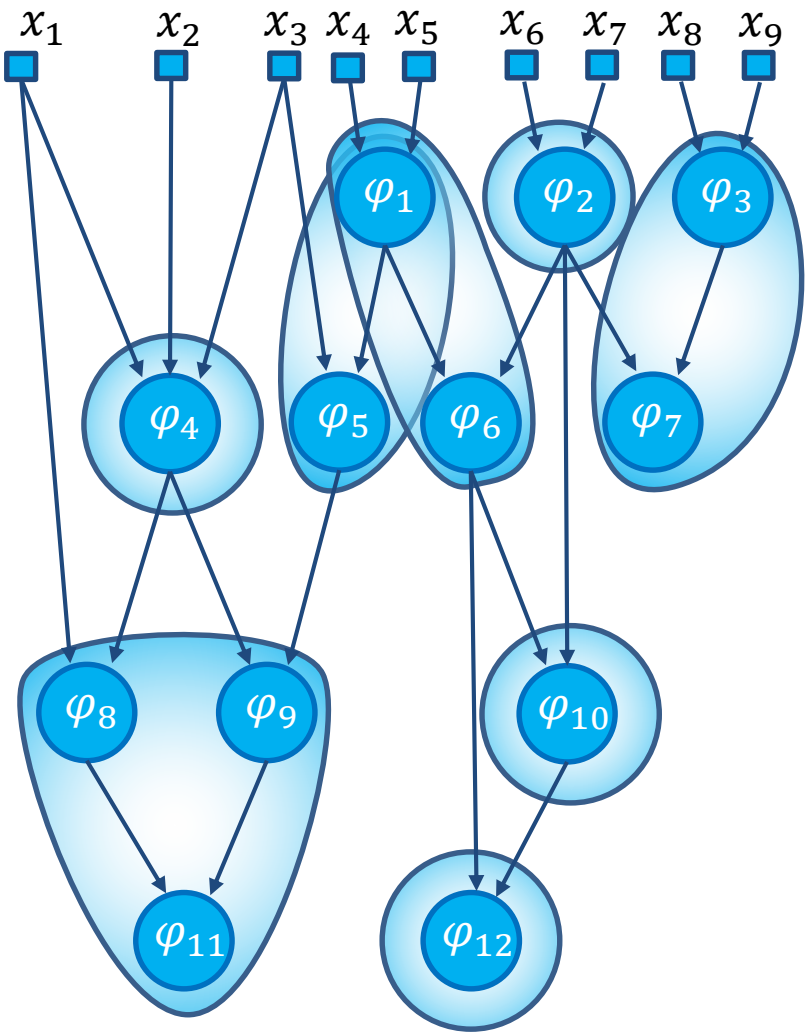


Покрытие СФЭ при помощи конусов

- Пример:

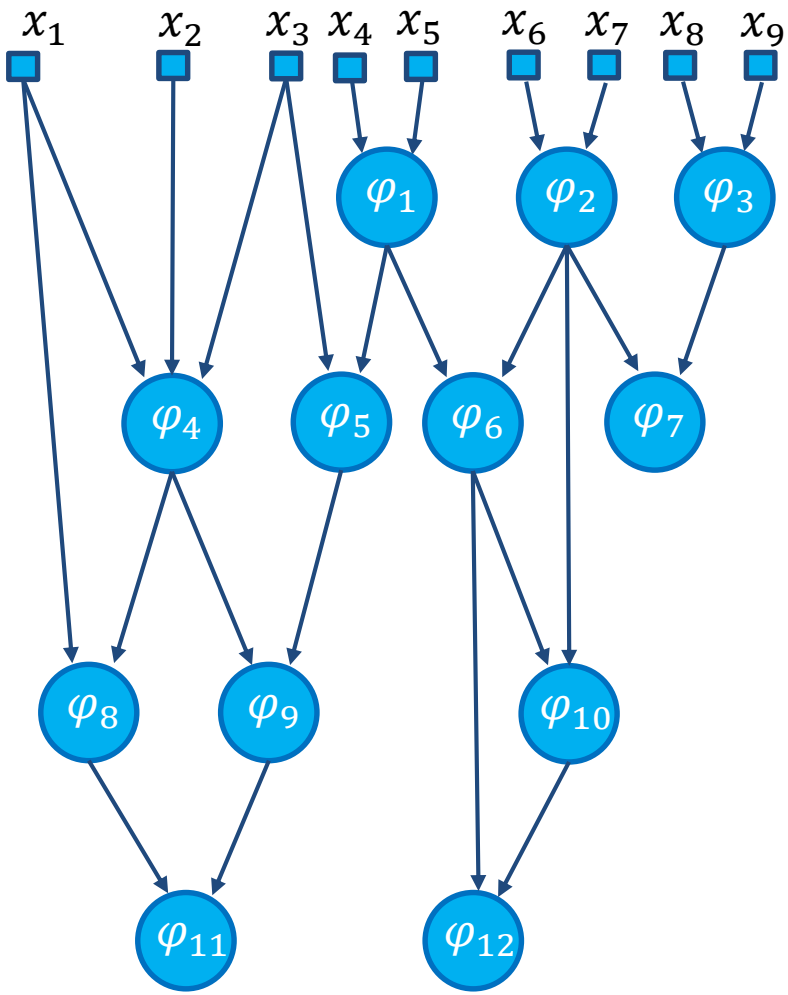
$$CS_{\Sigma} = \begin{Bmatrix} C_{\varphi_2}, C_{\varphi_4}, C_{\varphi_5}, C_{\varphi_6}, \\ C_{\varphi_7}, C_{\varphi_{10}} C_{\varphi_{11}}, C_{\varphi_{12}} \end{Bmatrix}$$

$$C_{\varphi_2} = \{\varphi_2\},$$
$$C_{\varphi_4} = \{\varphi_4\},$$
$$C_{\varphi_5} = \{\varphi_5, \varphi_1\},$$
$$C_{\varphi_6} = \{\varphi_6, \varphi_1\},$$
$$C_{\varphi_7} = \{\varphi_7, \varphi_3\},$$
$$C_{\varphi_{11}} = \{\varphi_{11}, \varphi_8, \varphi_9\},$$
$$C_{\varphi_{10}} = \{\varphi_{10}\},$$
$$C_{\varphi_{12}} = \{\varphi_{12}\}.$$



Разбиение СФЭ при помощи конусов

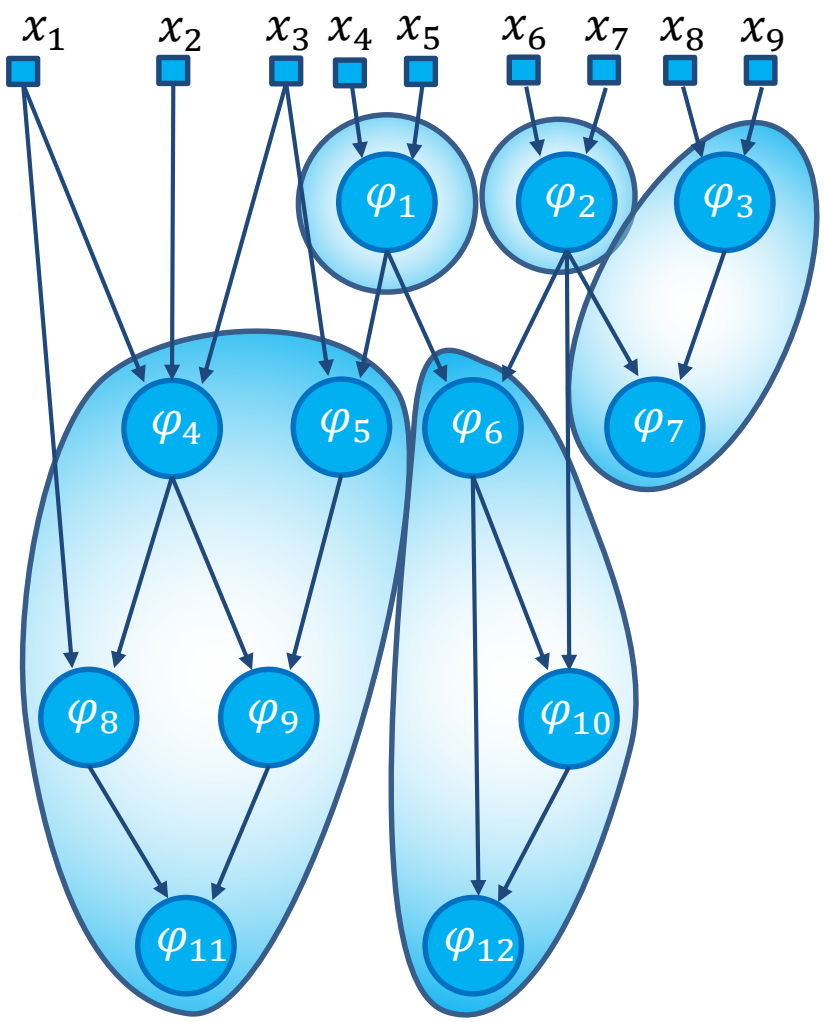
- Покрытие CS_Σ СФЭ Σ называется разбиением, если каждая вершина СФЭ Σ, отличная от входа Σ, входит только в один конус покрытия CS_Σ



Разбиение СФЭ при помощи конусов

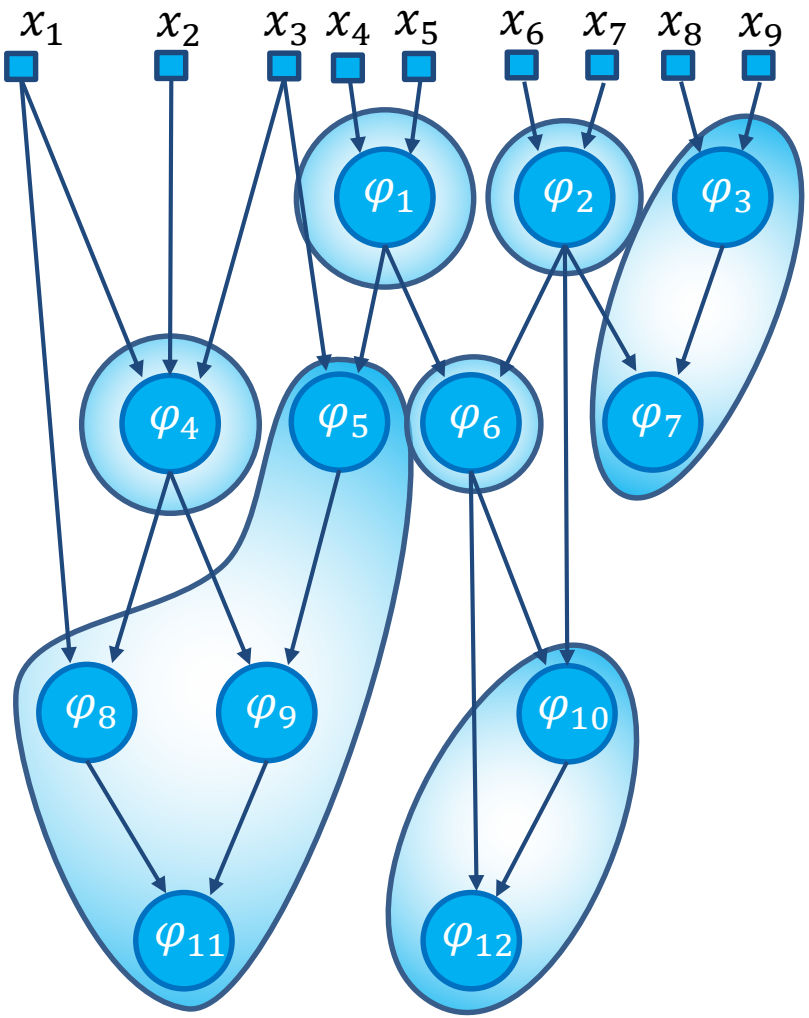
- Покрытие CS_Σ СФЭ Σ называется разбиением, если каждая вершина СФЭ Σ, отличная от входа Σ, входит только в один конус покрытия CS_Σ
- Пример:
  $CS_\Sigma = \{C_{\varphi_1}, C_{\varphi_2}, C_{\varphi_7}, C_{\varphi_{11}}, C_{\varphi_{12}}\}$

$$C_{\varphi_1} = \{\varphi_1\},$$
$$C_{\varphi_2} = \{\varphi_2\},$$
$$C_{\varphi_7} = \{\varphi_7, \varphi_3\},$$
$$C_{\varphi_{11}} = \{\varphi_{11}, \varphi_8, \varphi_9, \varphi_4, \varphi_5\},$$
$$C_{\varphi_7} = \{\varphi_{12}, \varphi_{10}, \varphi_6\}.$$



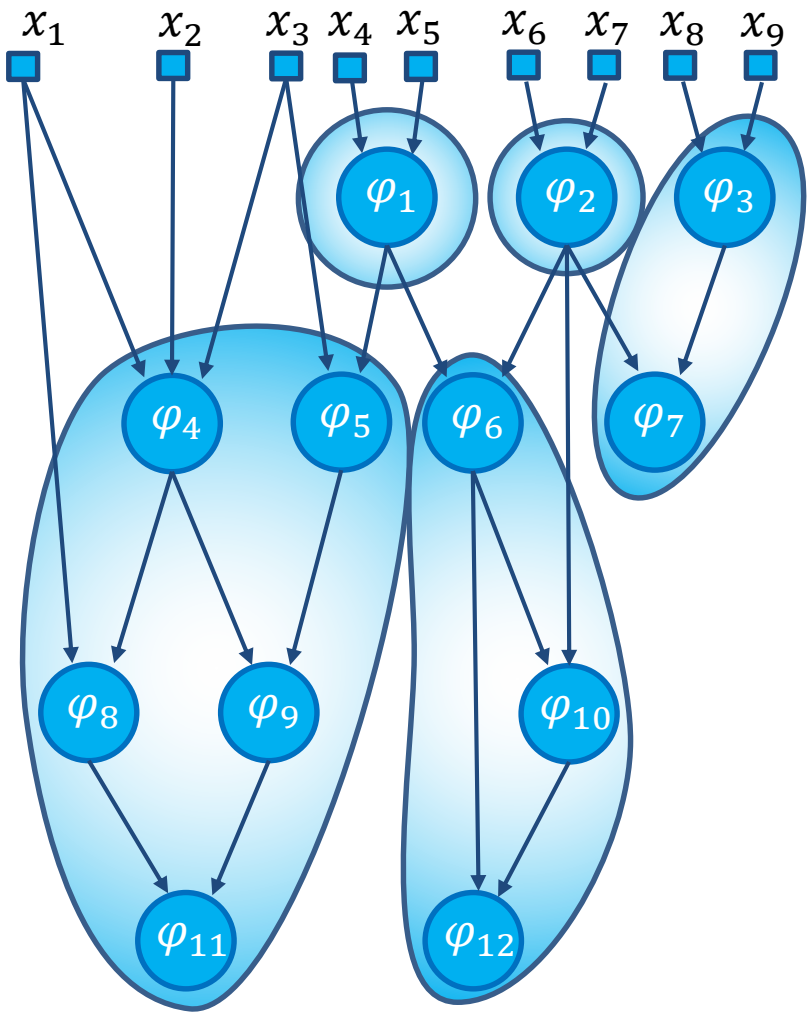
Специальные разбиения СФЭ

- Существует и единственное разбиение СФЭ Σ на
  - Максимальные деревья



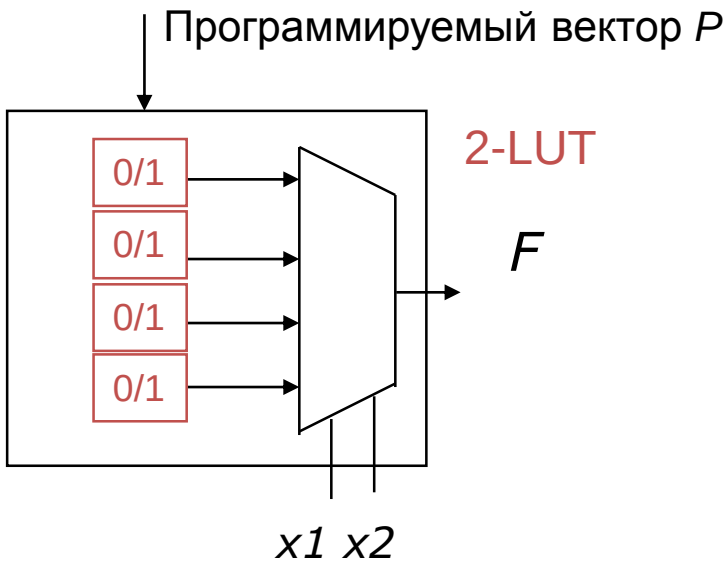
Специальные разбиения СФЭ

- Существует и единственное разбиение СФЭ Σ на
  - максимальные деревья
  - максимальные замкнутые конусы



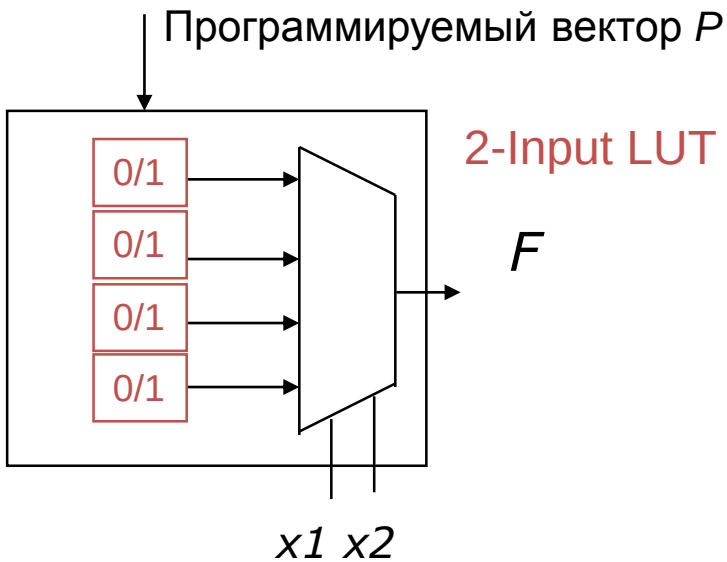
Логические элементы ПЛИС

- k-входовой универсальный элементы (англ. Look-Up Table, LUT)
- k-LUT представляет 2^k-битовую ячейку памяти и мультиплексор, информационные входы которого присоединены к соответствующим ячейкам памяти, а информационные входы являются входами универсального элемента



Логические элементы ПЛИС

- k-входовой универсальный элементы (англ. Look-Up Table, LUT)
- k-LUT представляет 2^k-битовую ячейку памяти и мультиплексор, информационные входы которого присоединены к соответствующим ячейкам памяти, а информационные входы являются входами универсального элемента

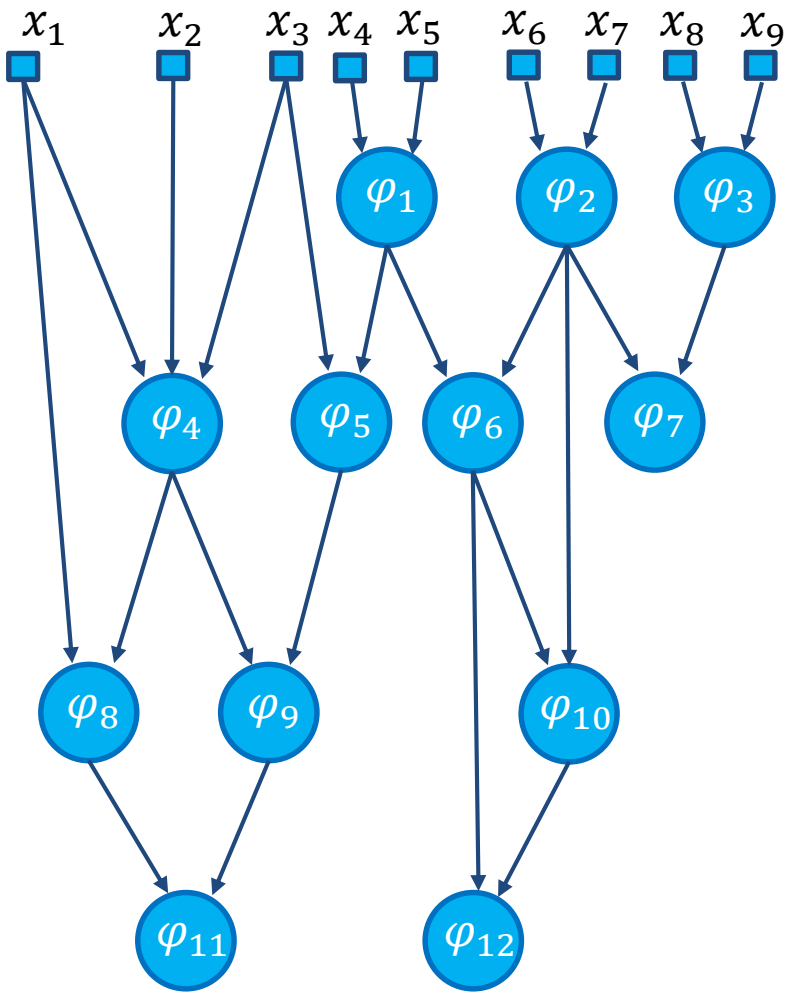


$f = \overline{x}_1\overline{x}_2 \vee x_1x_2$

x1	x2	F
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

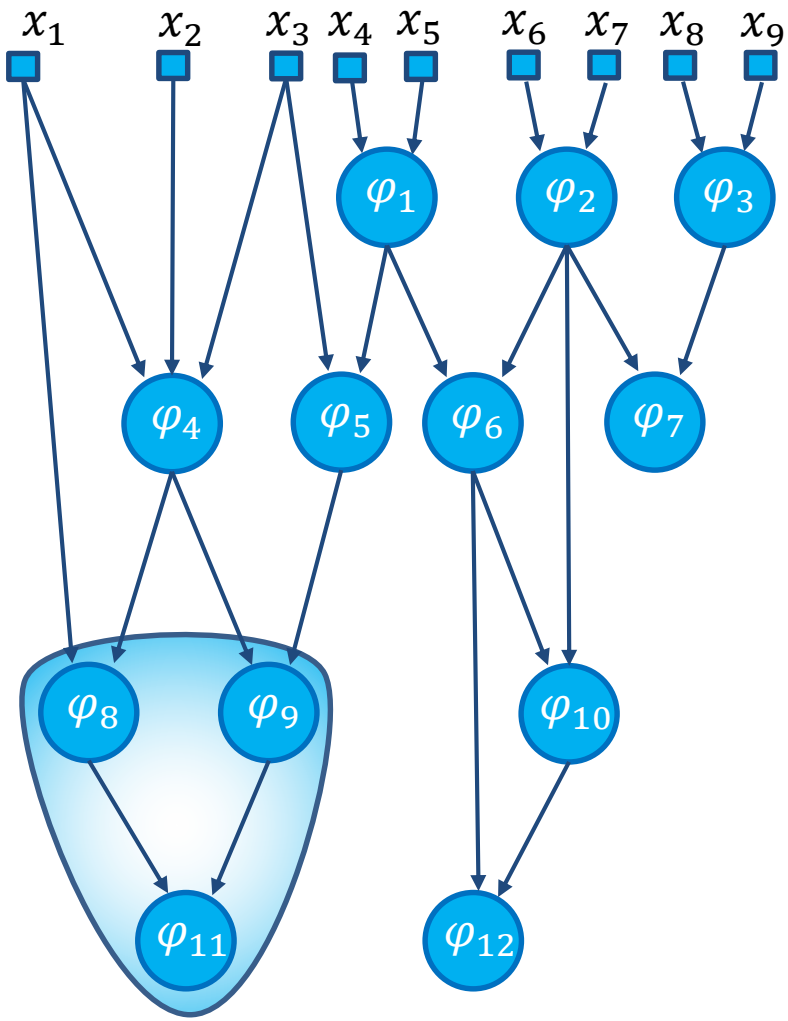
Покрытие СФЭ при помощи LUT

- k-LUT LUT_φ покрывает конус C_φ, если
  - выход конуса связан с выходом LUT
  - Каждое ребро, исходящее из элементов множества i_{C_φ} соответствует некоторому входу LUT_φ



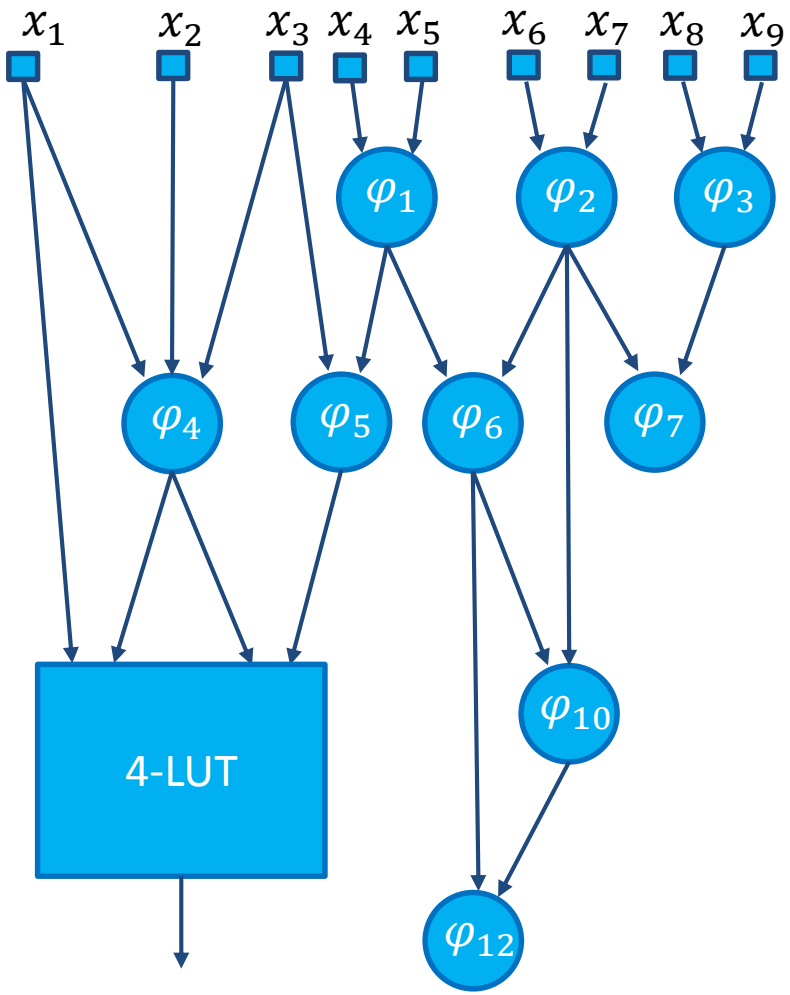
Покрытие СФЭ при помощи LUT

- k-LUT LUT_φ покрывает конус C_φ, если
  - выход конуса связан с выходом LUT
  - Каждое ребро, исходящее из элементов множества i_{C_φ} соответствует некоторому входу LUT_φ
- Пример:



Покрытие СФЭ при помощи LUT

- k-LUT LUT_φ покрывает конус C_φ, если
  - выход конуса связан с выходом LUT
  - Каждое ребро, исходящее из элементов множества i_{C_φ} соответствует некоторому входу LUT_φ
- Пример:



Задача привязки к библиотеке для ПЛИС

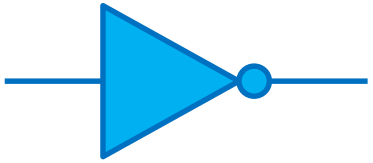
- Задача привязки к библиотеке для ПЛИС заключается в нахождении для СФЭ Σ такого покрытия при помощи конусов, что
  - Каждый конус покрывается соответствующим LUT-ом
  - Достигается минимум выбранного функционала качества
- Функционалы качества:
  - Сложность (площадь) – число LUT.
  - Глубина – максимальное число LUT-ов по всем возможным путям от входов до выходов Σ

Общая задача привязки к библиотеке

- Задана библиотека $B = \{\phi_i | \phi_i = f_i(X), L(\phi_i) = L_i, D(\phi_i) = D_i\}$
- Общая задача привязки к библиотеке заключается в нахождении для СФЭ Σ такого покрытия при помощи конусов, что
  - Каждый конус покрывается соответствующим элементом ϕ_i библиотеки B
  - Достигается минимум выбранного функционала качества
- Функционалы качества:
  - Сложность (площадь) – суммарная сложность всех используемых элементов библиотеки B.
  - Глубина – максимальная взвешенная глубина по всем возможным путям от входов до выходов Σ

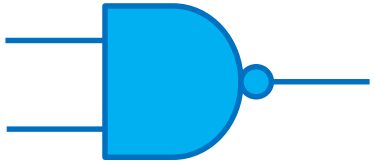
Представление библиотеки

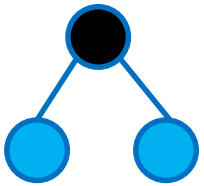
INV



I1v t1.1

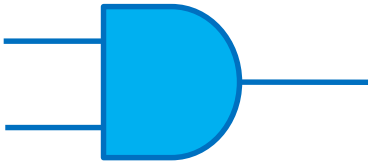
NAND2

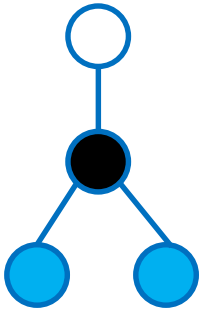




N1v t2.1
N2v t2.2

AND2



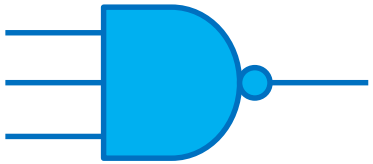


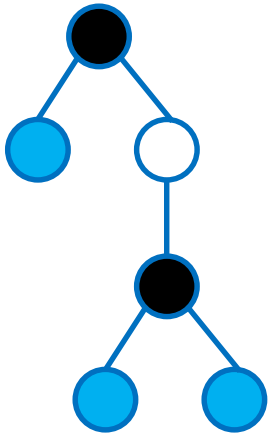
I1N1v t3.1
I1N2v t3.2

INV(I) NAND2(N) Terminal(v)

Представление библиотеки

NAND3





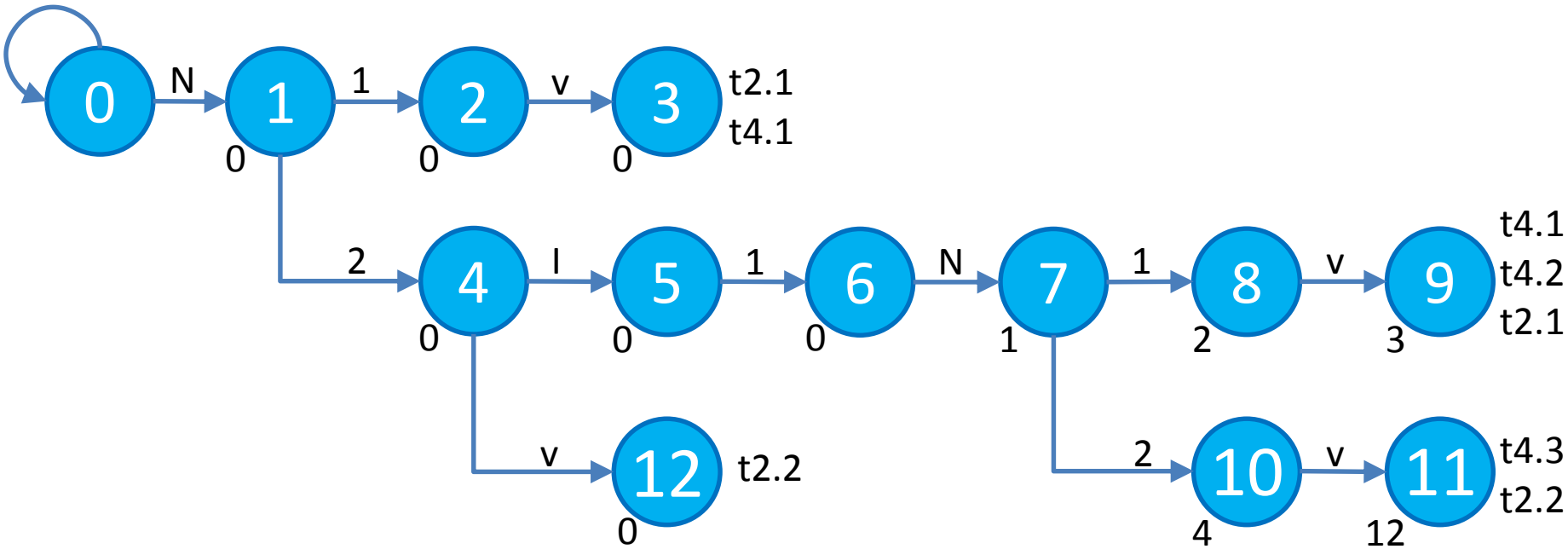
N1v	t4.1
N2I1N1v	t4.2
N2I1N2v	t4.3

INV(I) ○ NAND2(N) ● Terminal(v) ●

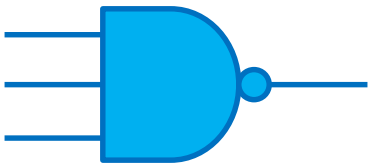
Рекурсивный алгоритм поиска структурных соответствий

```
Match (u, v)
        if u.isLeaf() return True
        else
                if v.isLeaf() return False
                if u.degree() != v.degree() return False
                if v.degree() == 1
                        u_c := u.get_child()
                        v_c := v.get_child()
                        return Match(u_c,v_c)
                else
                        u_l := u.get_left_child()
                        u_r := u.get_right_child()
                        v_l := v.get_left_child()
                        v_r := v.get_right_child()
                        m_1 := Match(u_l,v_l) and Match(u_r,v_r)
                        m_2 := Match(u_l,v_r) and Match(u_r,v_l)
                        return m_1 or m_2
```

Построение автомата

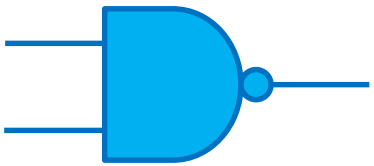


NAND3



N1v	t4.1
N2I1N1v	t4.2
N2I1N2v	t4.3

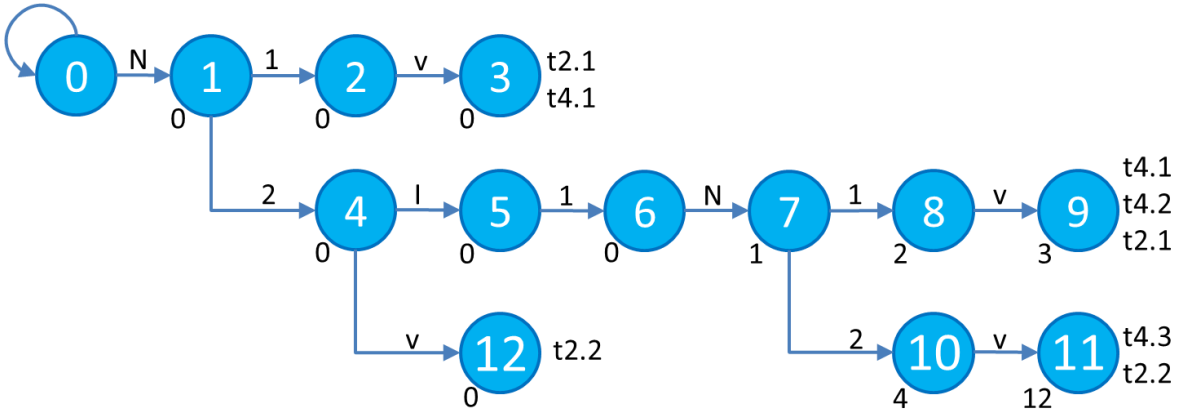
NAND2

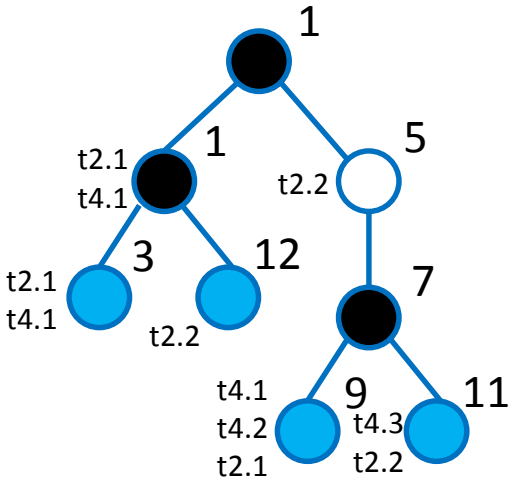


N1v	t2.1
N2v	t2.2

Обход целевого дерева

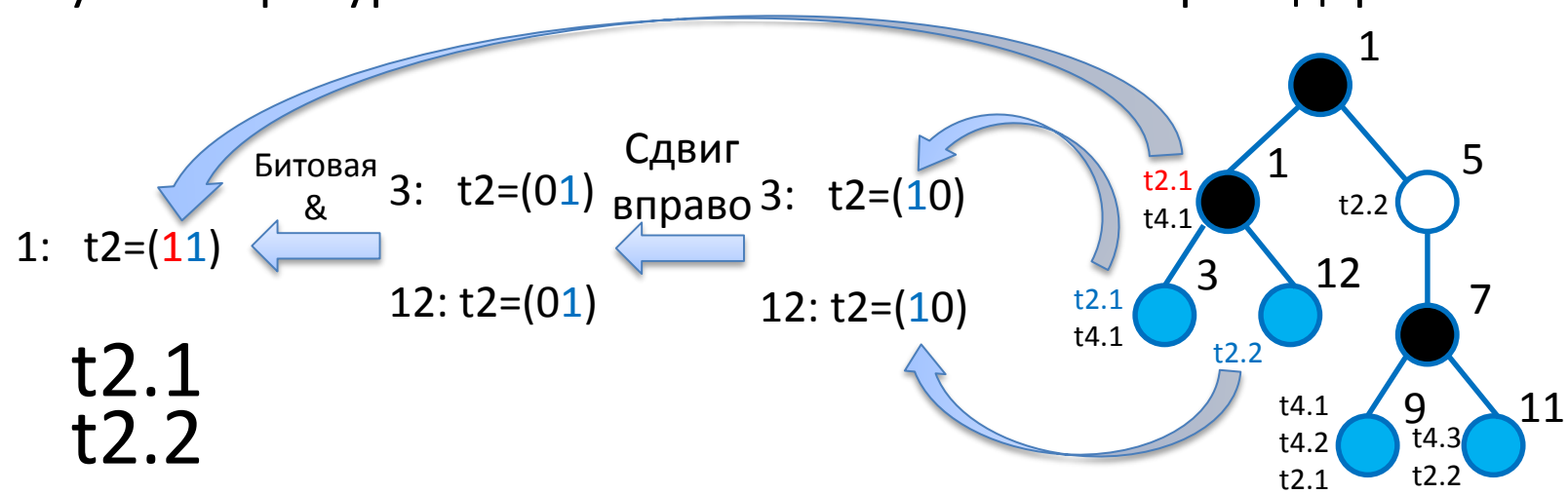
```
Visit_Node (G, n)
        if n – корень дерева
                n.state:=Next(0, n.symbol)
        else
                n.state:=Next(Next(n.parent.state,k),n.symbol)
                //вершина n является k-ым потомком родительской
                //вершины n.parent
        foreach потомка c вершины n
                Visit_Node(G, c)
        Post_Process(G,n)
```





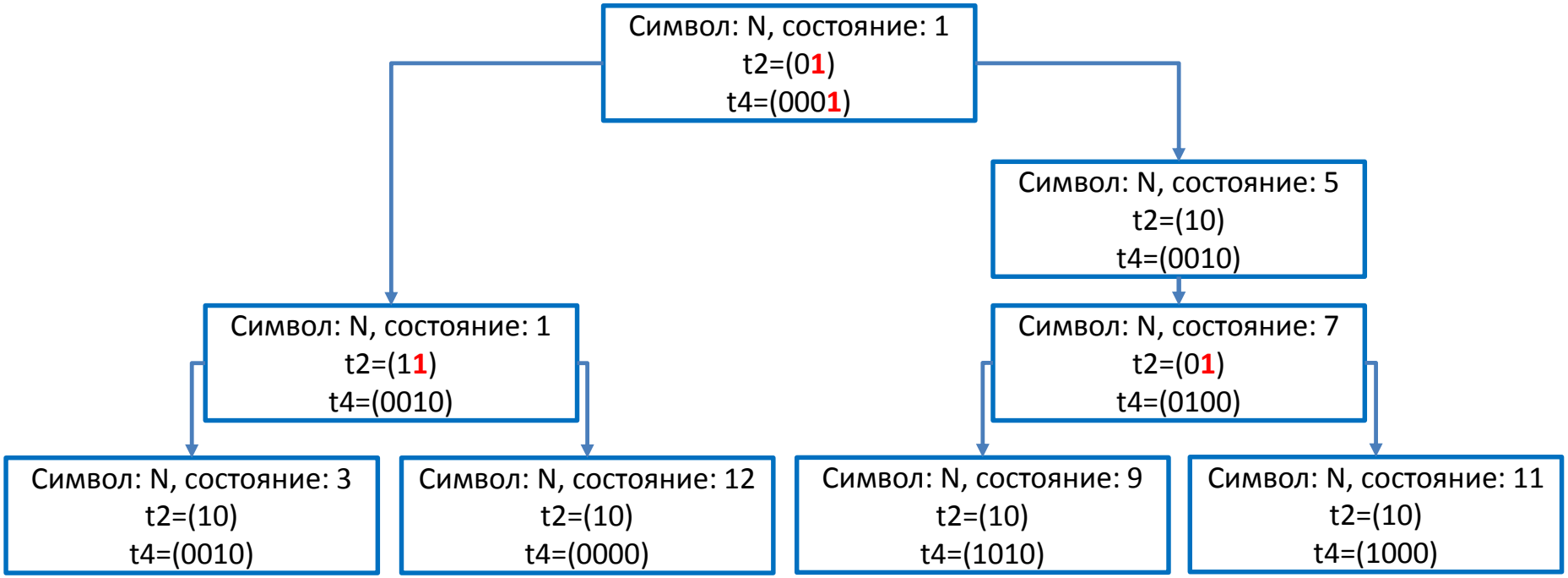
Нахождение соответствий

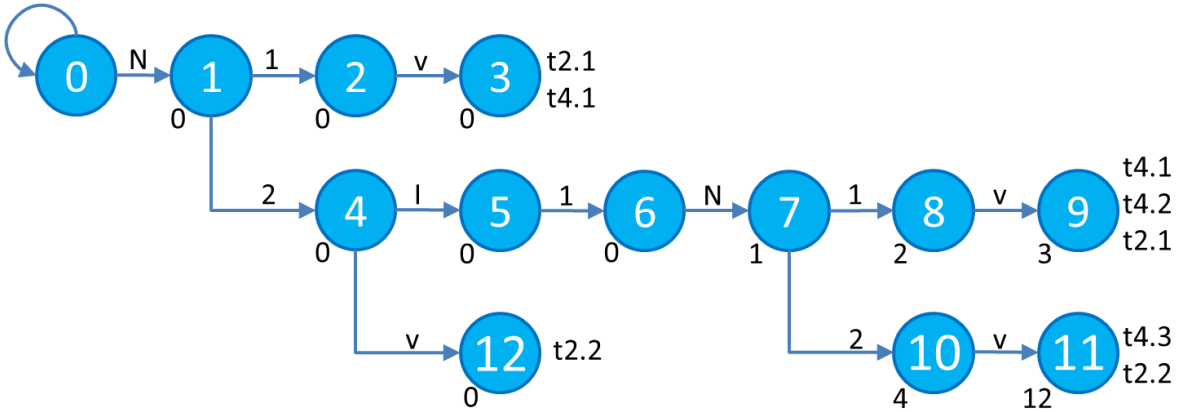
- Кодирование найденных соответствий при помощи битовых векторов:
  - Для каждого шаблона t_i в каждой вершине дерева задается битовый вектор длины $\left\lceil \frac{n_i}{2} \right\rceil + 1$
- k-ый элемент вектора в вершине принимает значение 1:
  - если существует строка $t_{i.j}$ длины $2k - 1$ шаблона t_i, сопоставленная в данной вершине
  - если во всех потомках данной вершины $k + 1$ элемент вектора шаблона t_i равен 1
- Могут быть рекурсивно вычислены от листов к корню дерева

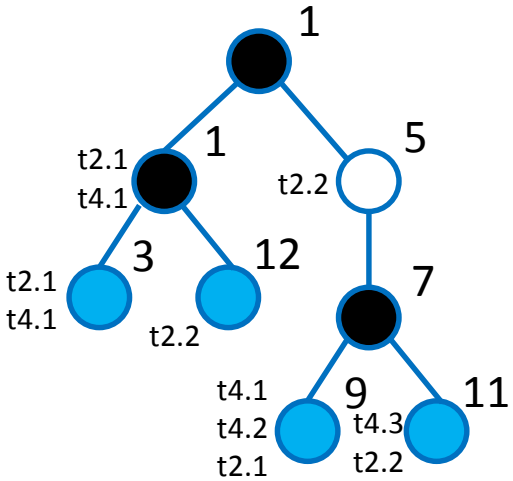


N1v t2.1
N2v t2.2

Нахождение соответствий







Нахождение соответствий

```
Post_Process (G, n)
        n.b(i):=0
        if n.state == “ACCEPT”
                Set_Partial(G, n, n.state)
        foreach шаблона t_i
                n.b(i):=n.b(i) or ∏_{c∈C(n)} c.b_i/2
                //C(n) – множество всех дочерних вершин вершины n
        //Do_Reduce(G, n)

Set_Partial(G, n, s)
        foreach строки t_i длины 2j+1 обнаруженных в s
                n.b(i):=n.b(i) or 2^j
```