Математические модели и методы логического синтеза СБИС

Осень 2022

Лекция 4

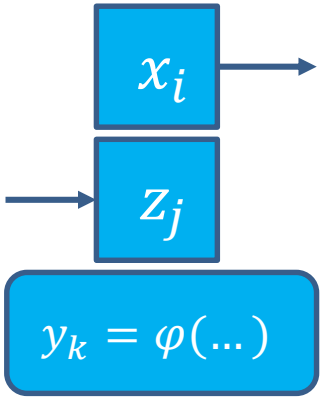
План лекции

- Комбинационные логические сети (КЛС)
  - Структура и функционирование КЛС.
  - Задача оптимизации КЛС (различные постановки задач, функционалы качества при оптимизации КЛС).
  - Основные типы преобразований КЛС: исключение, разложение, экстракция, упрощение и подстановка.
  - Разложение на основе «деления» – поиск общих подфункций.

Комбинационные логические сети

Структура комбинационных логических сетей

- Комбинационная логическая сеть (КЛС) – ациклический ориентированный помеченный граф $\Sigma = (V, E)$ специальной структуры
- Множество вершин $V = (V_{in}, V_{out}, V_\Sigma)$ образуют разбиение:
  - V_{in} - множество основных входов
  - V_{out} - множество основных выходов
  - V_Σ - множество внутренних вершин



Внутренние вершины комбинационных логических сетей

- Каждой внутренней вершине КЛС приписаны:
  - внутренняя переменная $y_k \in Y$
  - ФАЛ φ, зависящая от переменных из множества $X_{y_k} \subseteq X \times Y$, и соответствующая этой ФАЛ представление в некоторой модели

$y_k = (01101001)$

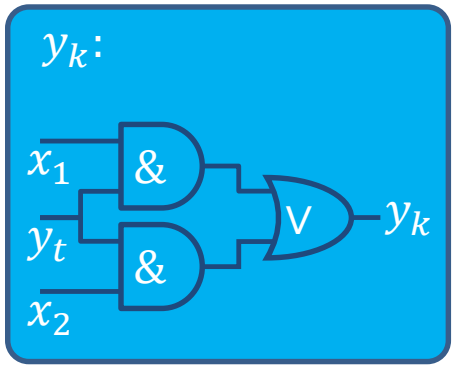
Таблица истинности

$y_k = x_1 y_t \vee x_2 y_s$

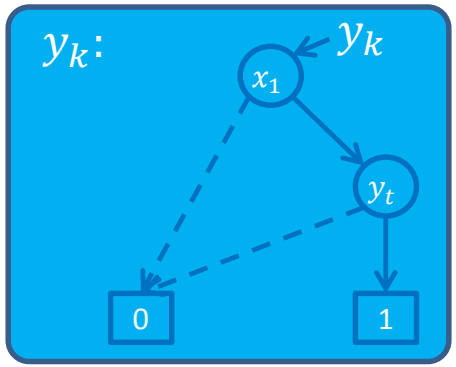
ДНФ

$y_k = (x_1 | x_2) \sim y_s$

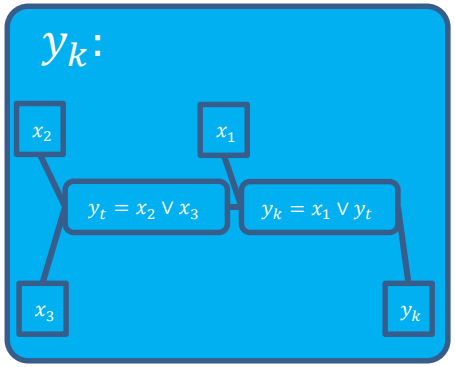
Формула



СФЭ и AIG



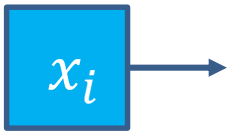
BDD



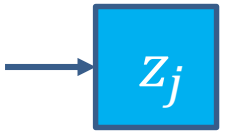
КЛС

Дуги комбинационных логических сетей

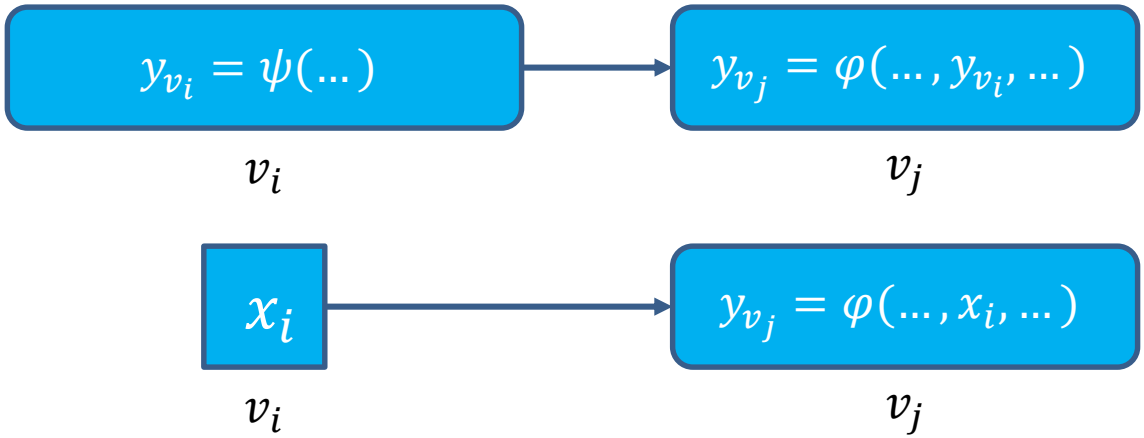
- Основные входы являются истоками



- Основные выходы являются стоками



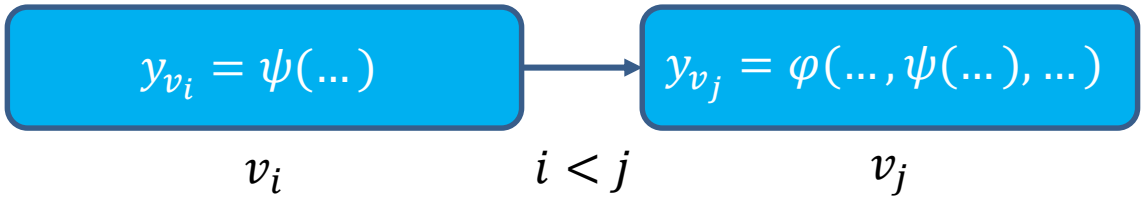
- Дуга $e = (v_i, v_j) \in E$ существует в КЛС $\Sigma = (V, E)$, если ФАЛ, реализуемая в v_j зависит от ФАЛ, реализуемой в v_i



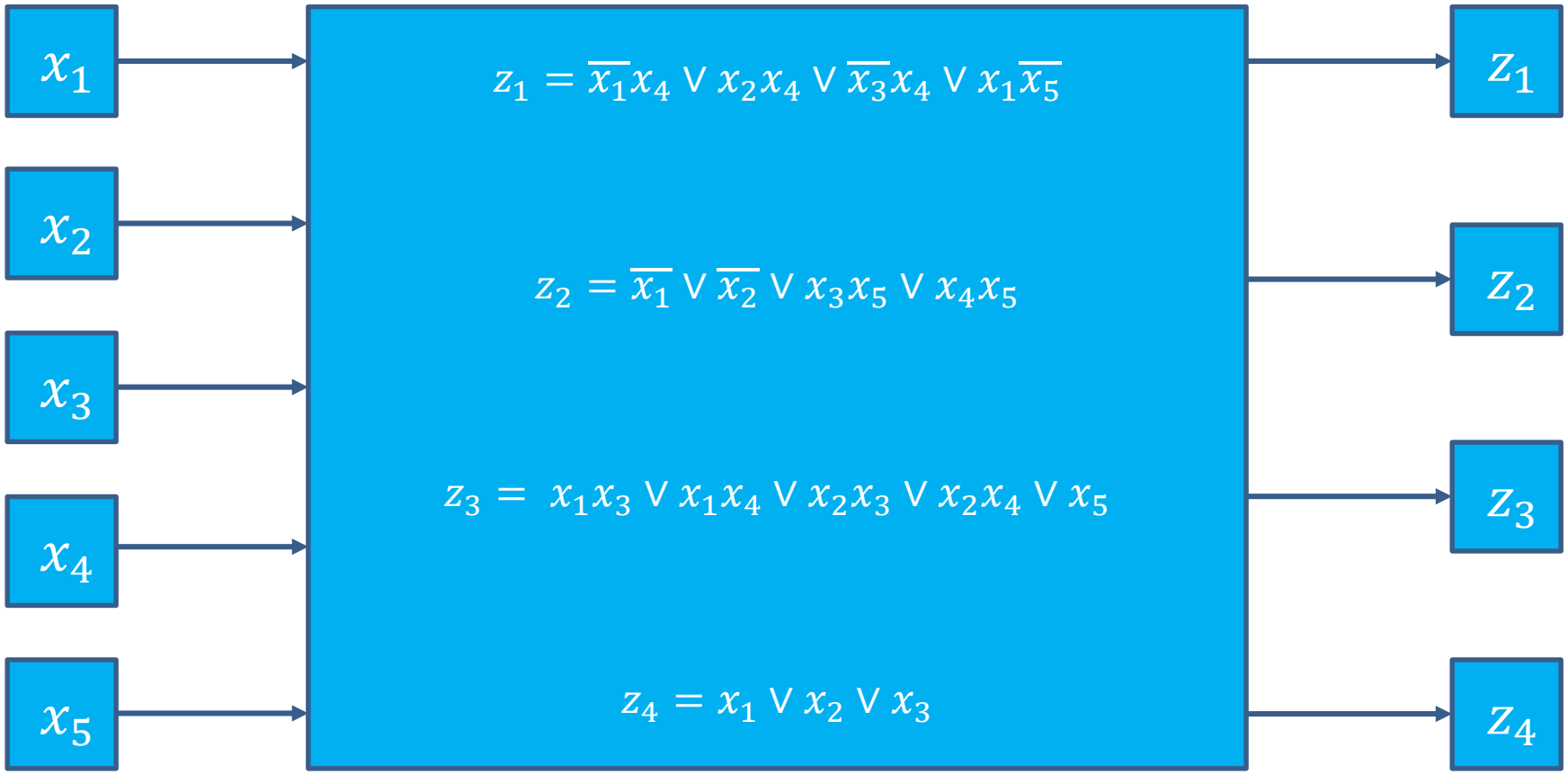
Функционирование комбинационных логических сетей

- Основные входы реализуют тождественные ФАЛ

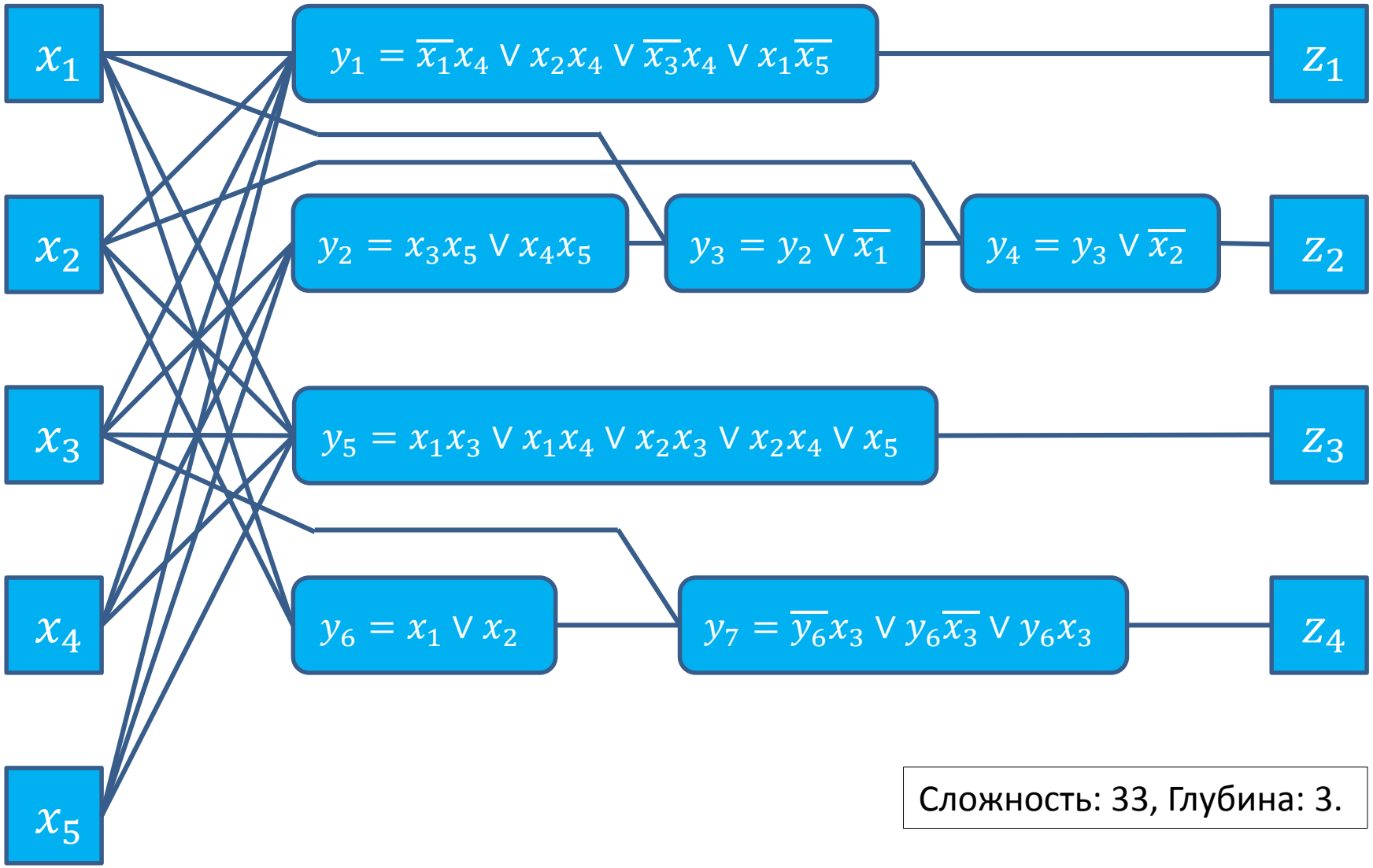
- Так как граф КЛС является ациклическим, то ФАЛ (как функции основных входов), определяются последовательно на основе топологического порядка внутренних вершин



Комбинационная логическая сеть - пример



Комбинационная логическая сеть - пример



Сложность: 33, Глубина: 3.

Оптимизация комбинационных логических сетей

- Преобразование КЛС за счет специальных операций
- Все операции должны сохранять эквивалентность исходной и преобразованной схемы с точки зрения реализуемых выходных функций
- Преобразования можно разделить на два класса:
  - преобразования вершин (локальные)
  - преобразования структуры схемы (глобальные)

Функционалы качества при оптимизация комбинационных логических сетей

- Сначала определяются функционалы качества для внутренних вершин КЛС, а потом на их основе определяются функционалы качества для КЛС в целом
- Функционалы качества во внутренней вершине определяется на основе представления, которое используется для задания ФАЛ, реализуемой в этой вершине
- Для иерархических КЛС функционалы качества определяются рекурсивно

Функционалы качества комбинационных логических сетей

- Для ДНФ A
  - Длина $\lambda(\mathrm{A})$ – число элементарных конъюнкций в A
  - Сложность(ранг) $R(A)$ - число букв в A

$$y_k = x_1 y_t \vee x_2 y_s$$

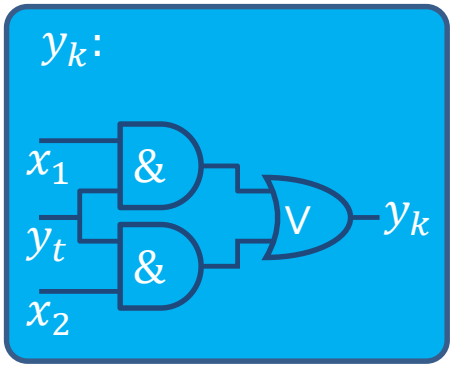
ДНФ

- Для формулы F
  - Ранг $R(F)$ - число букв в F
  - Сложность $L(F)$ – число операций в F
  - Глубина $D(F)$ - глубина корня формулы F

$$y_k = (x_1 | x_2) \sim y_s$$

Формула

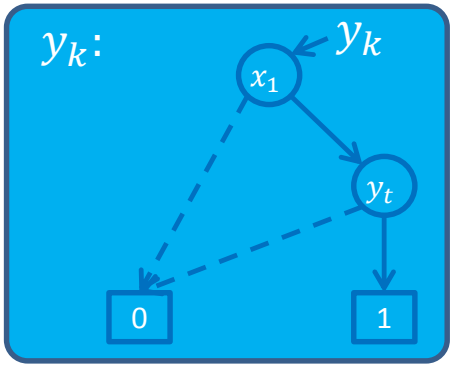
- Для СФЭ Σ
  - Сложность $L(F)$ – число функциональных элементов в Σ
  - Глубина $D(F)$ - максимальная длина пути от входов до выходов СФЭ Σ



СФЭ и AIG

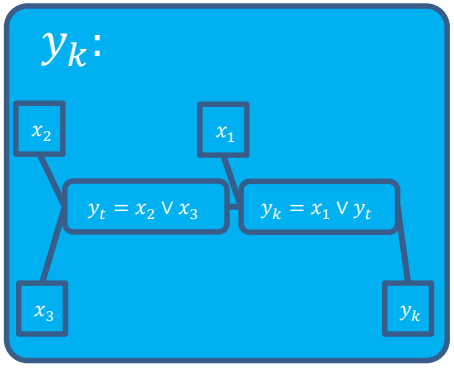
Функционалы качества комбинационных логических сетей

- Для BDD A
  - Сложность $L(A)$ - число нетерминальных вершин в BDD A
  - Глубина $D(F)$ - глубина корня BDD A



BDD

- Для КЛС Σ
  - Сложность $L(F)$ – сумма сложностей во всех внутренних вершинах КЛС Σ
  - Глубина $D(F)$ - максимальная взвешенная глубина пути от входов до выходов КЛС Σ

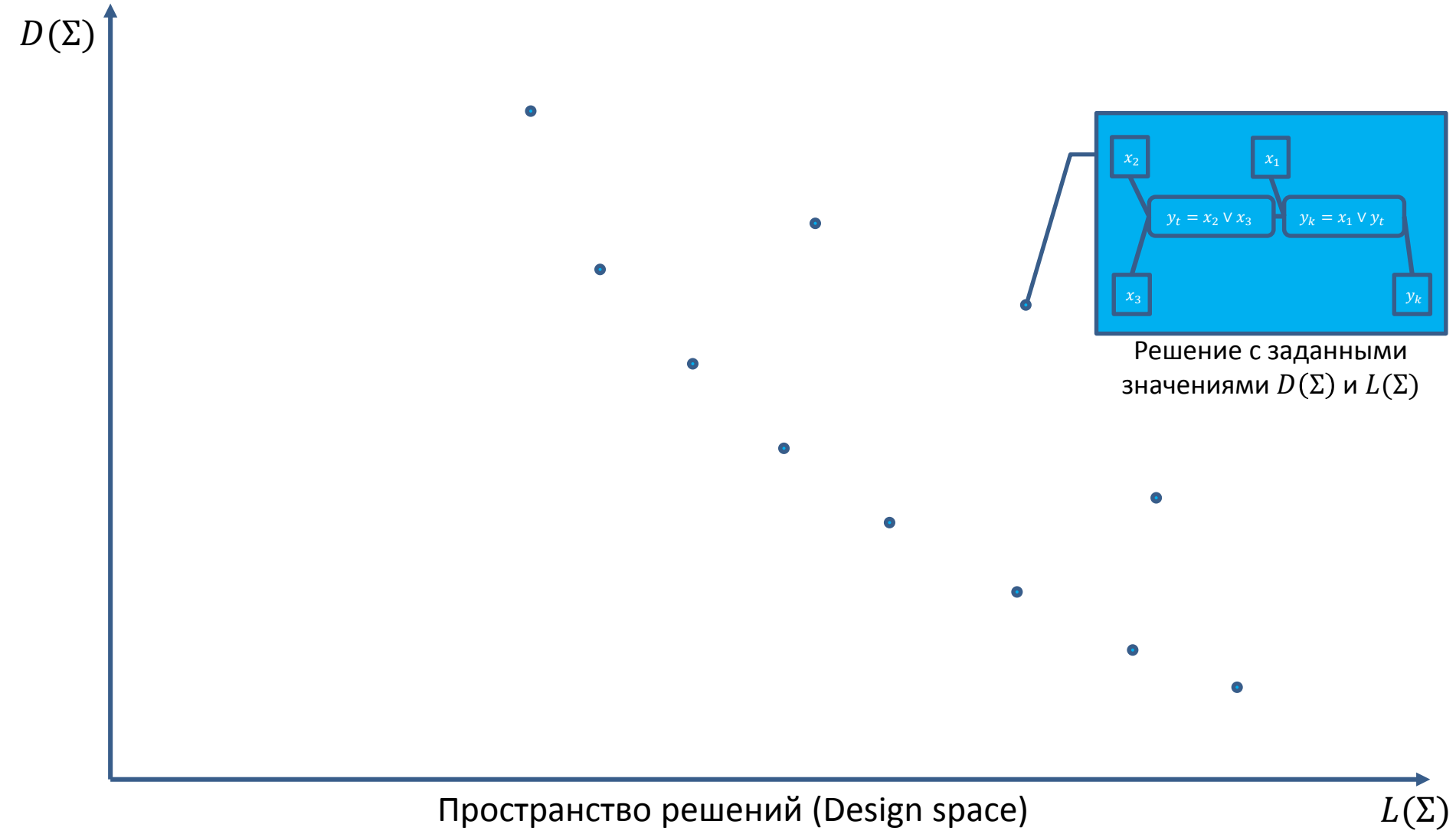


КЛС

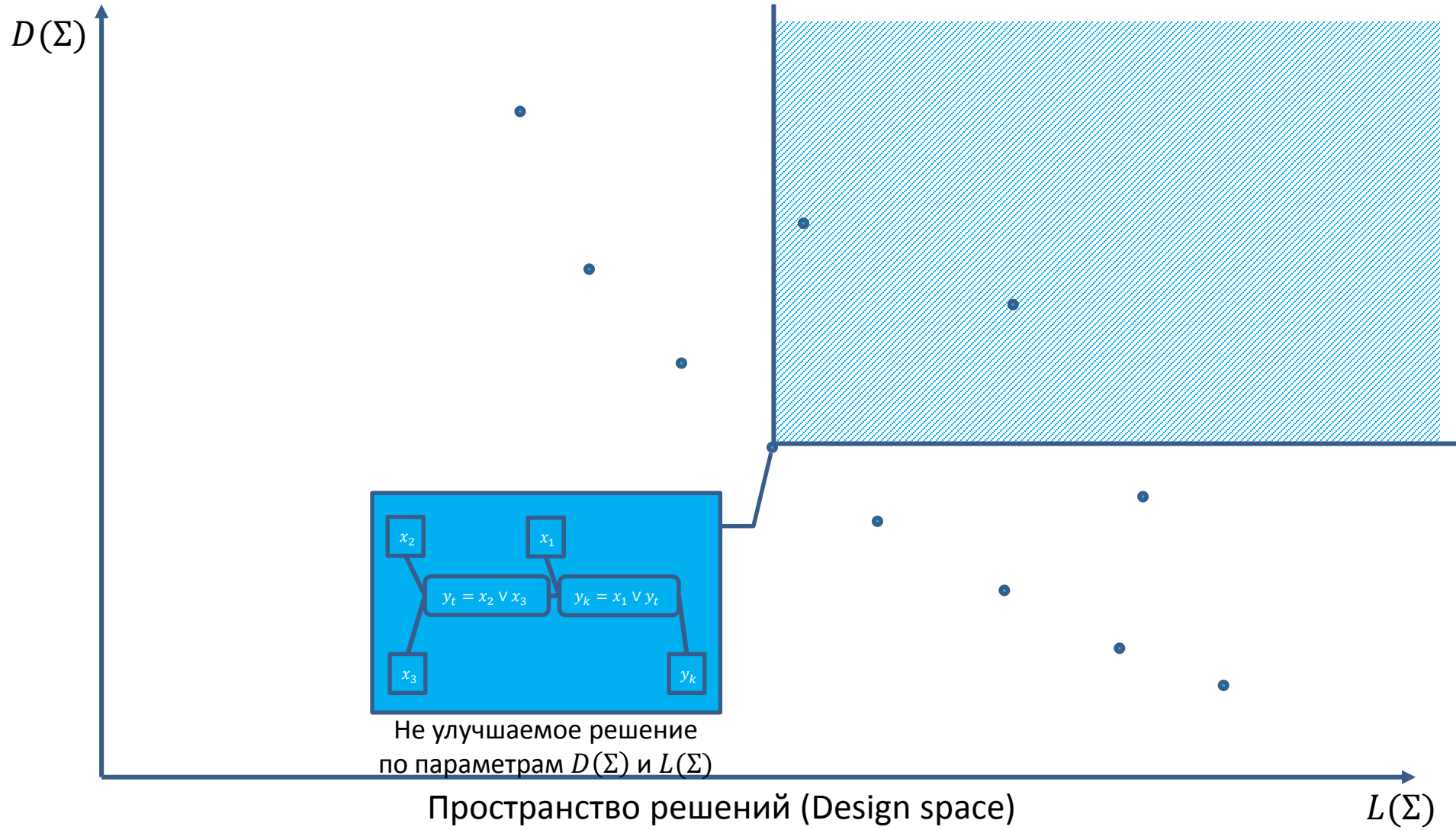
Задача оптимизации комбинационных логических сетей

- Построение КЛС для заданной системы ФАЛ с наилучшими значения выбранных функционалов качества
- Многокритериальная задача оптимизации выбранных функционалов качества
- Однокритериальная задача оптимизации выбранного функционала качества с дополнительными ограничениями на другие функционалы качества

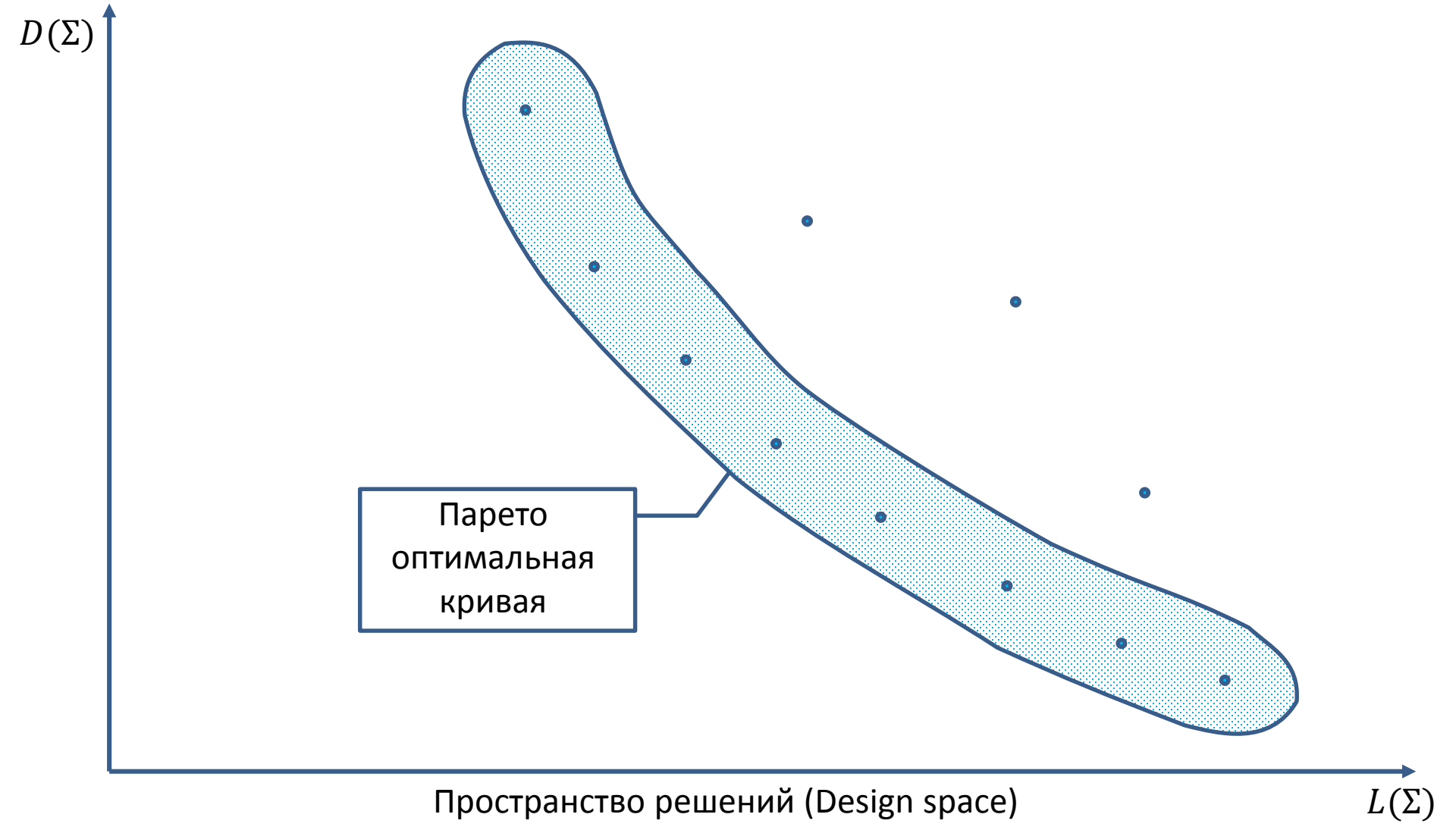
Пространство решений задачи оптимизации комбинационных логических сетей



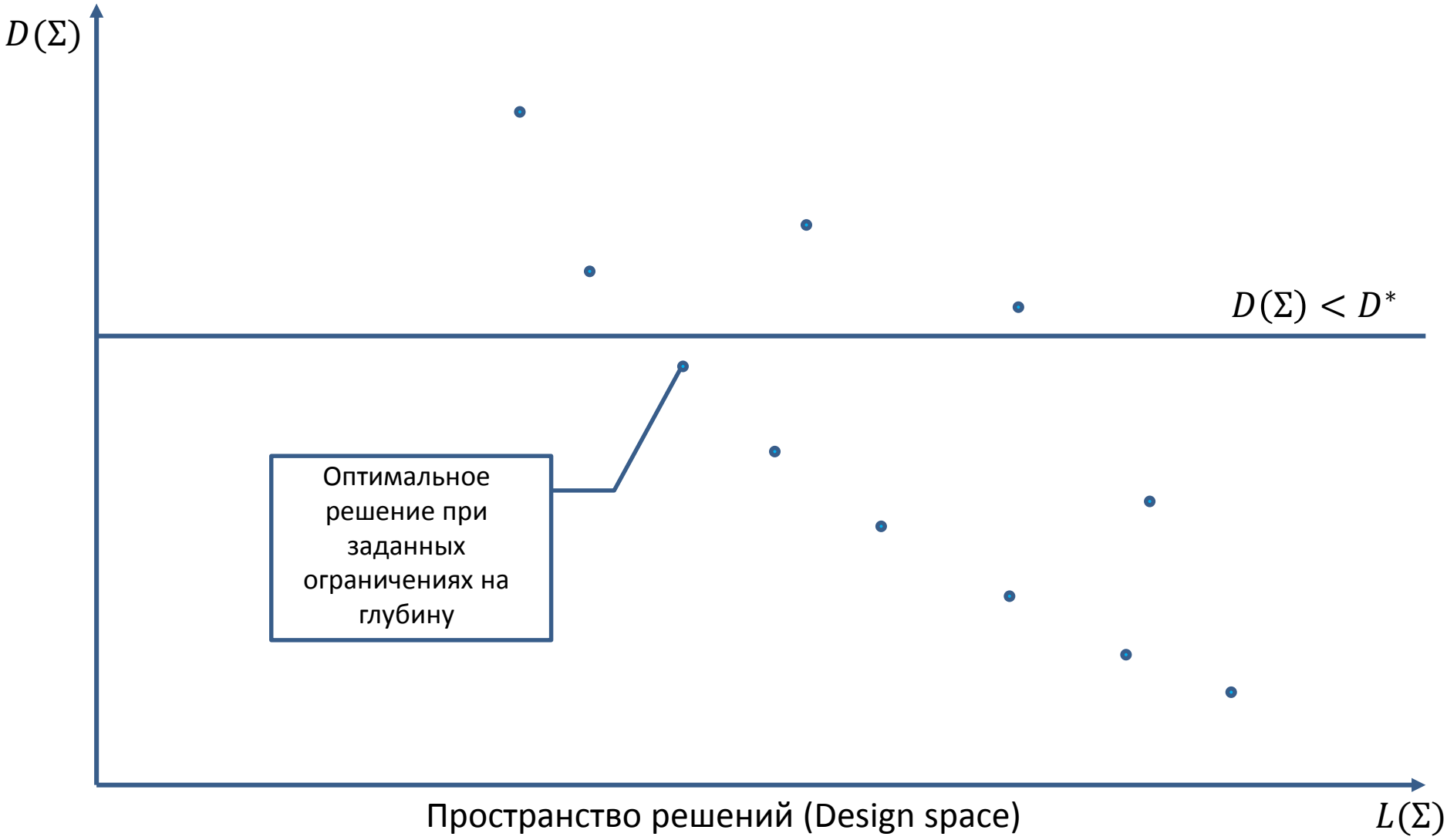
Пространство решений задачи оптимизации комбинационных логических сетей



Парето оптимальная кривая решений многокритериальной задачи оптимизации



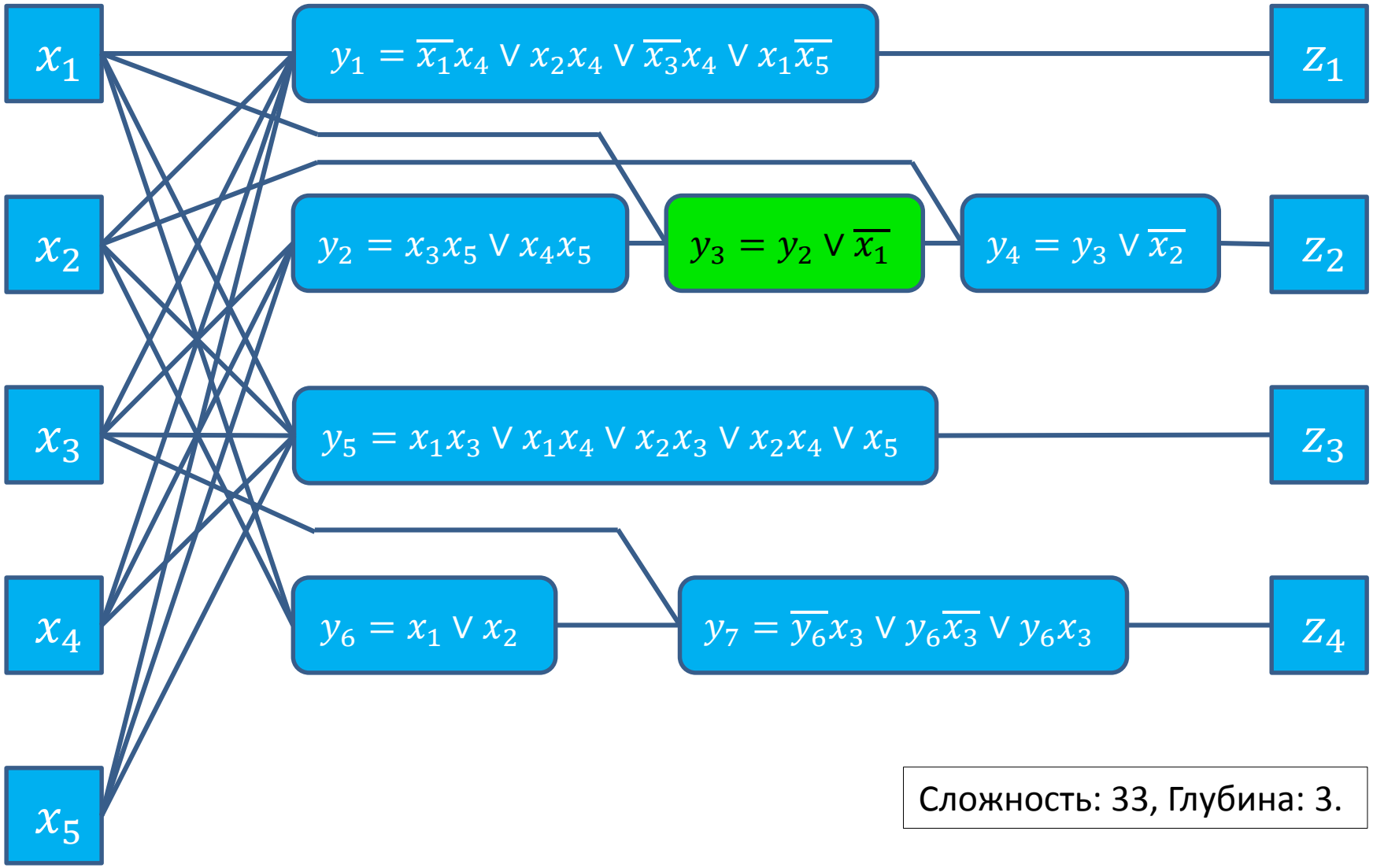
Однокритериальная задача оптимизации с дополнительными ограничениями



Преобразования комбинационных логических сетей

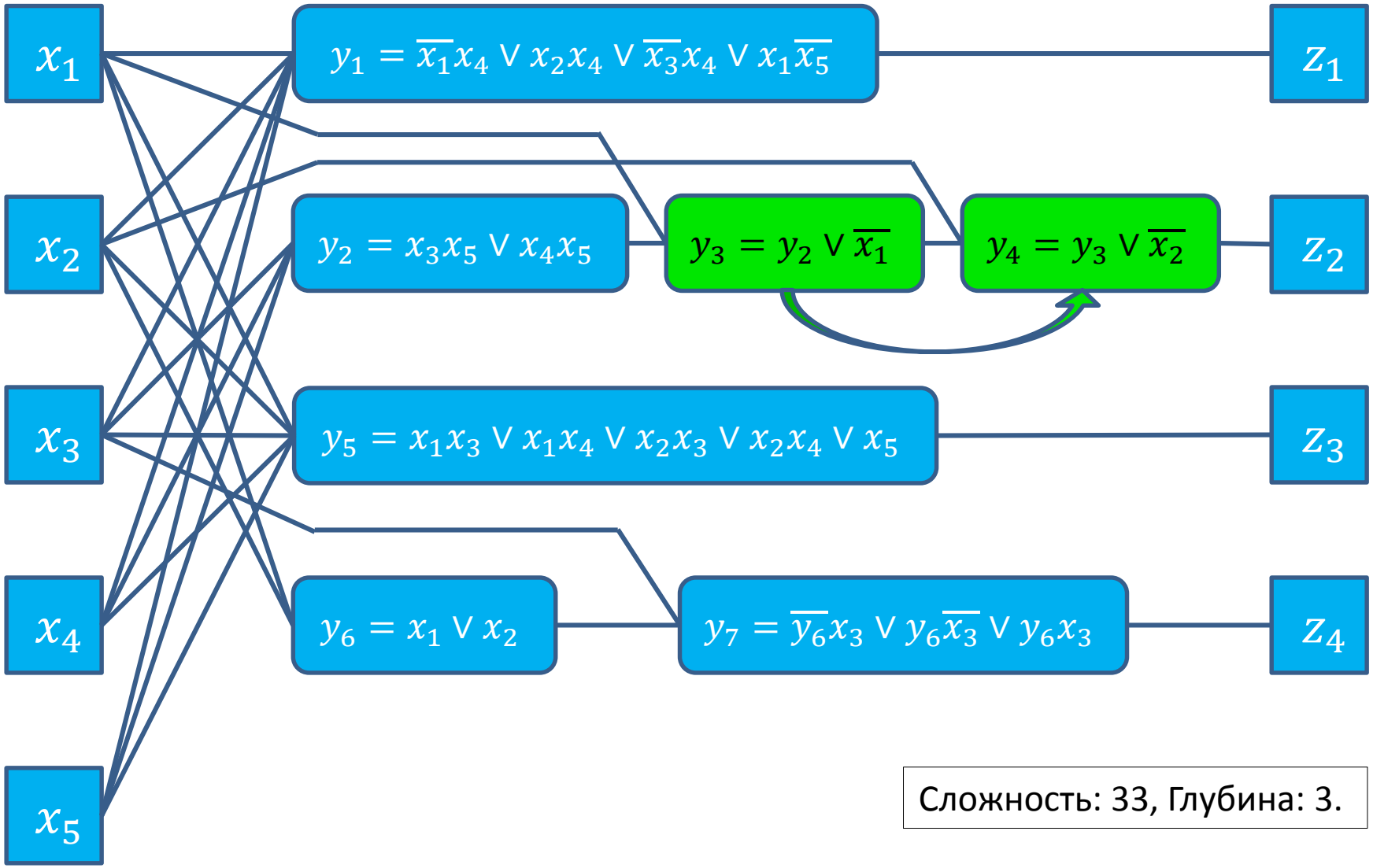
- Исключение (ELIMINATION) – удаление внутренней вершины КЛС и соответствующая подстановка ФАЛ, реализуемой этой вершиной в другие вершины
- Разложение (DECOMPOSITION) – замена внутренней вершины несколькими вершинами, которые формируют подсеть, реализующую такую же ФАЛ, что и исходная вершина
- Извлечение (EXTRACTION) – добавление новой внутренней вершины, которая реализует ФАЛ, являющейся подфункцией для нескольких других внутренних вершин, и соответствующая подстановка символа новой вершины в указанные вершины
- Упрощение (SIMPLIFICATION) – оптимизация представления ФАЛ, реализуемой во внутренней вершине
- Подстановка (SUBSTITUTION) – упрощение представления ФАЛ, реализуемой в вершине за счет увеличения числа переменных, от которых зависит указанная ФАЛ

Пример исключения (ELIMINATION)



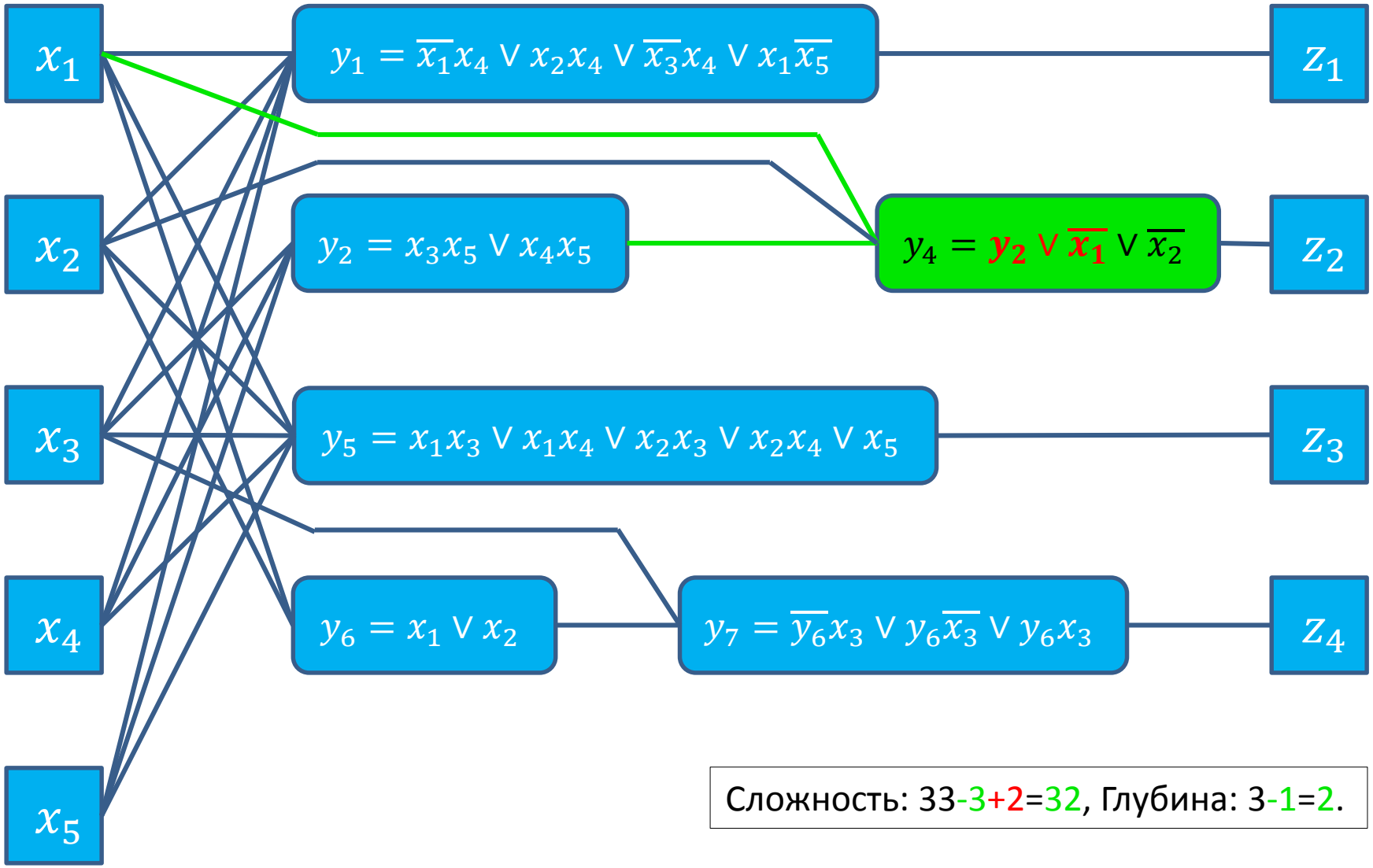
Сложность: 33, Глубина: 3.

Пример исключения (ELIMINATION)



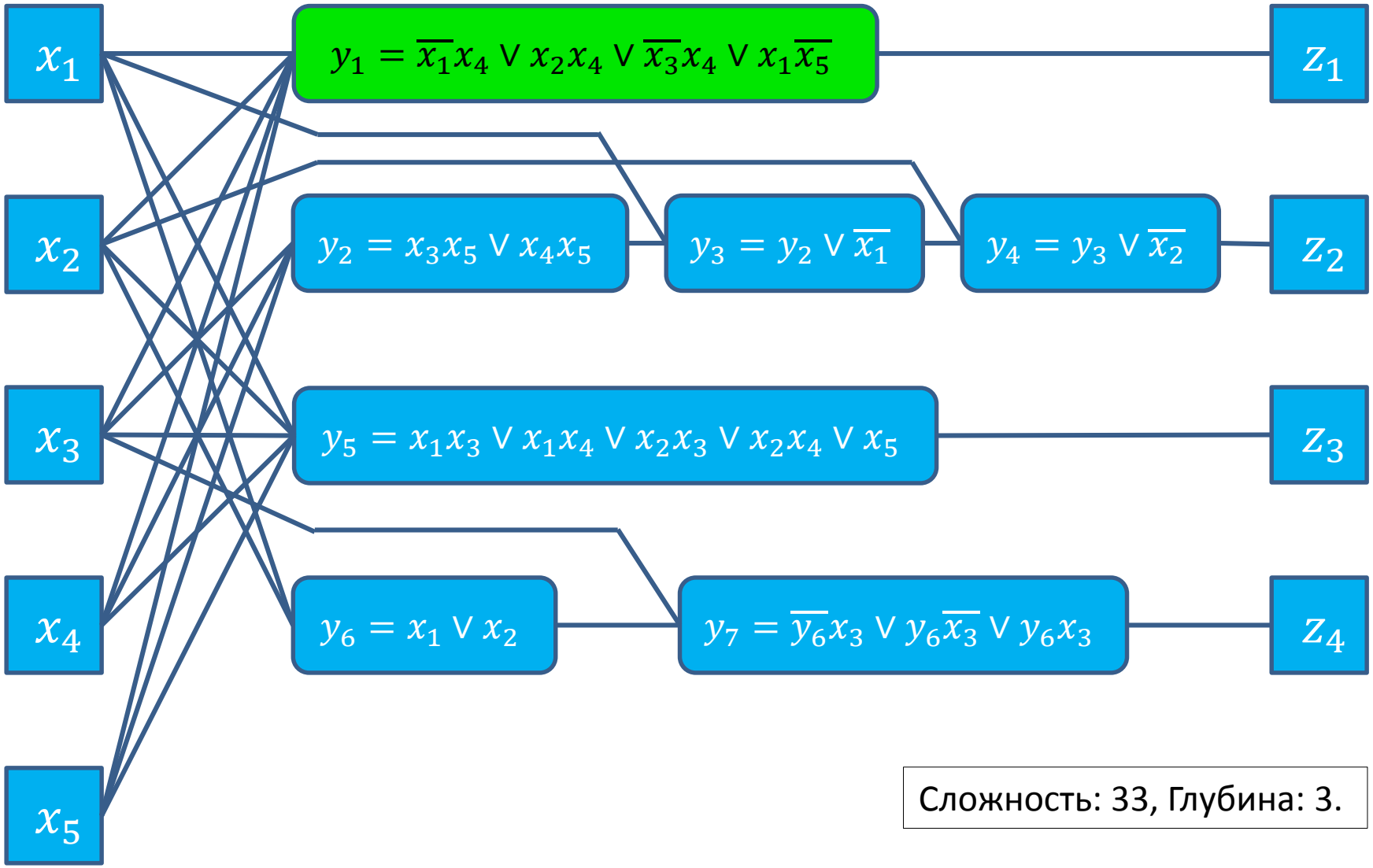
Сложность: 33, Глубина: 3.

Пример исключения (ELIMINATION)



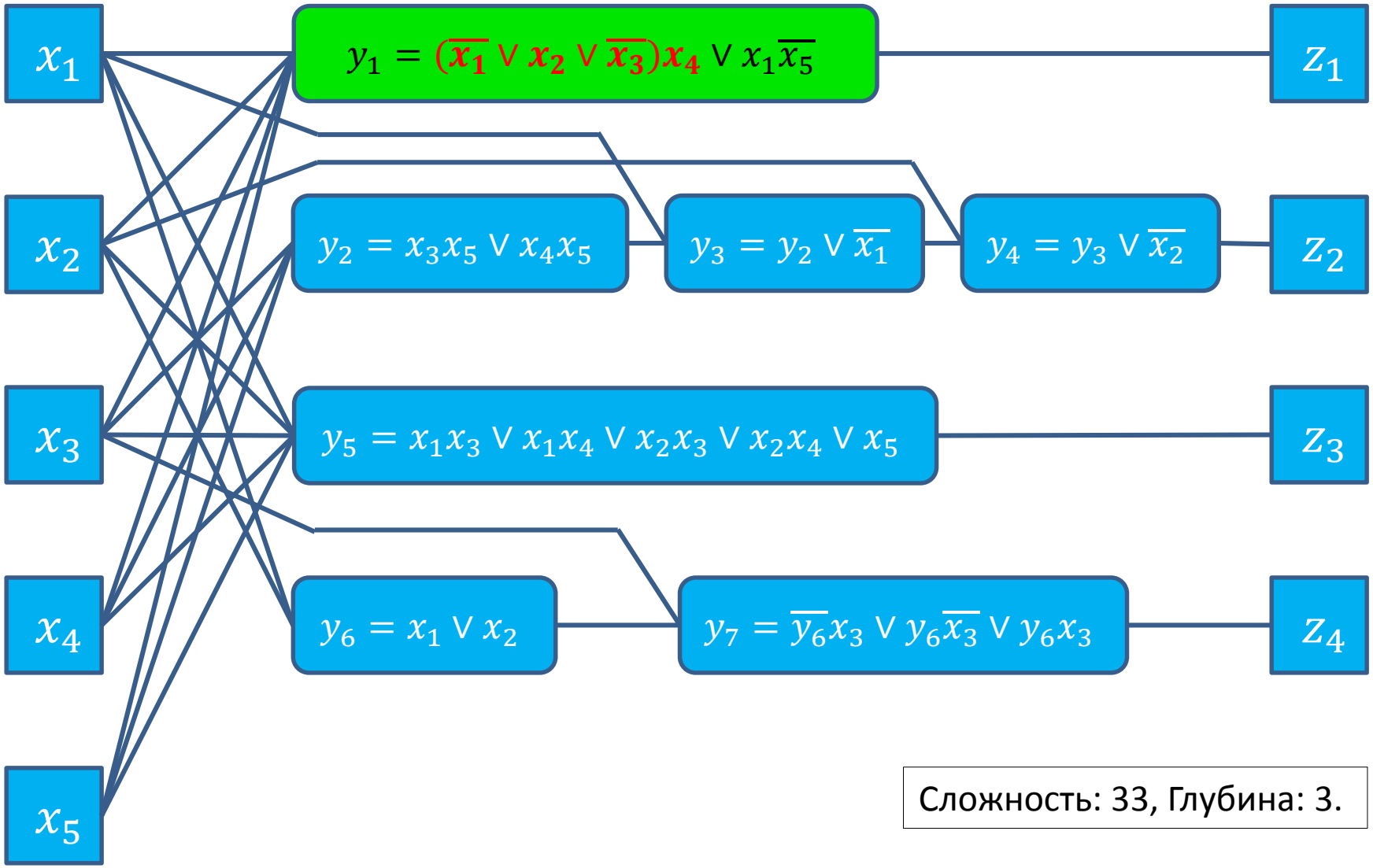
Сложность: 33-3+2=32, Глубина: 3-1=2.

Пример разложения (DECOMPOSITION)



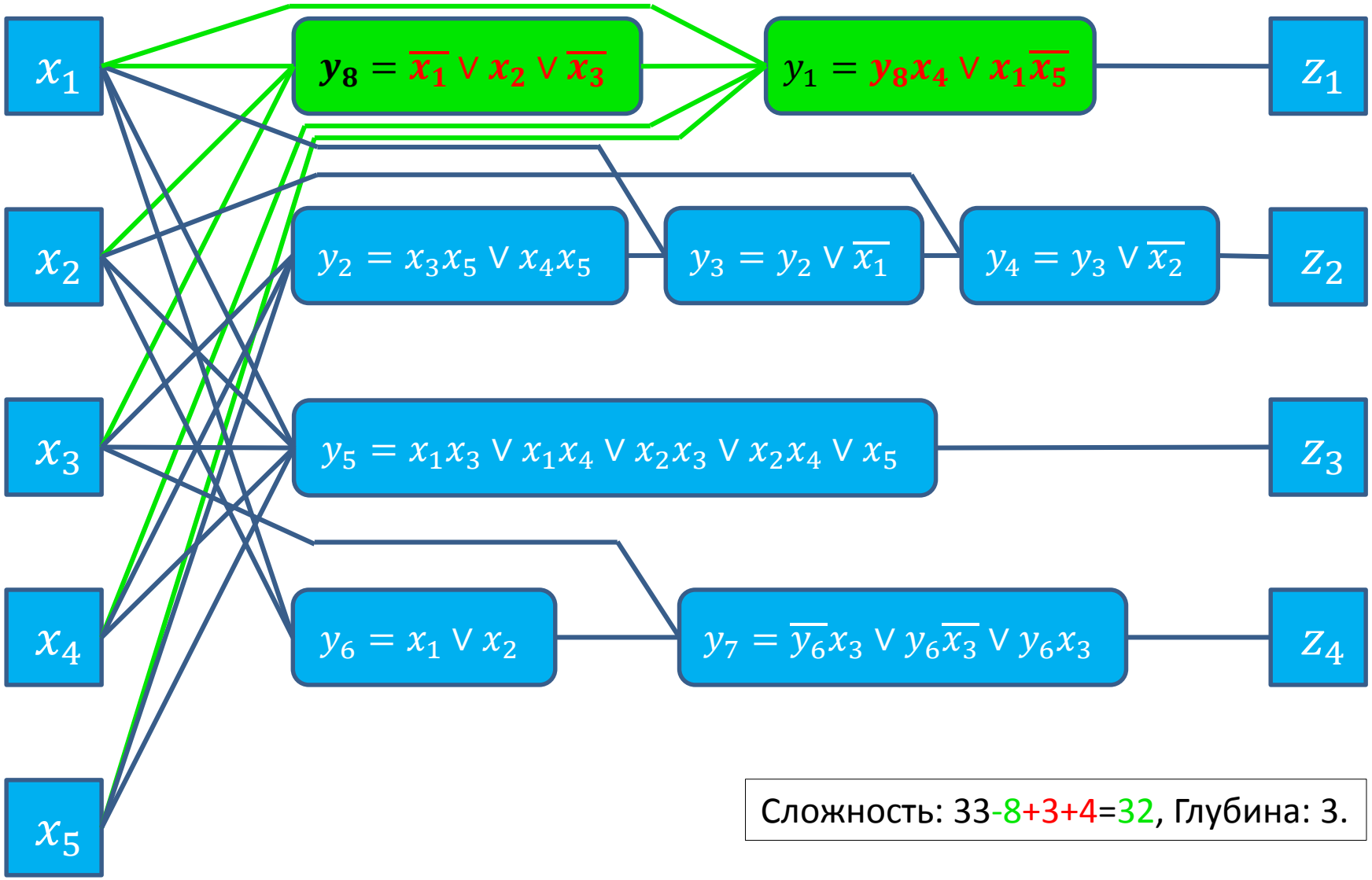
Сложность: 33, Глубина: 3.

Пример разложения (DECOMPOSITION)



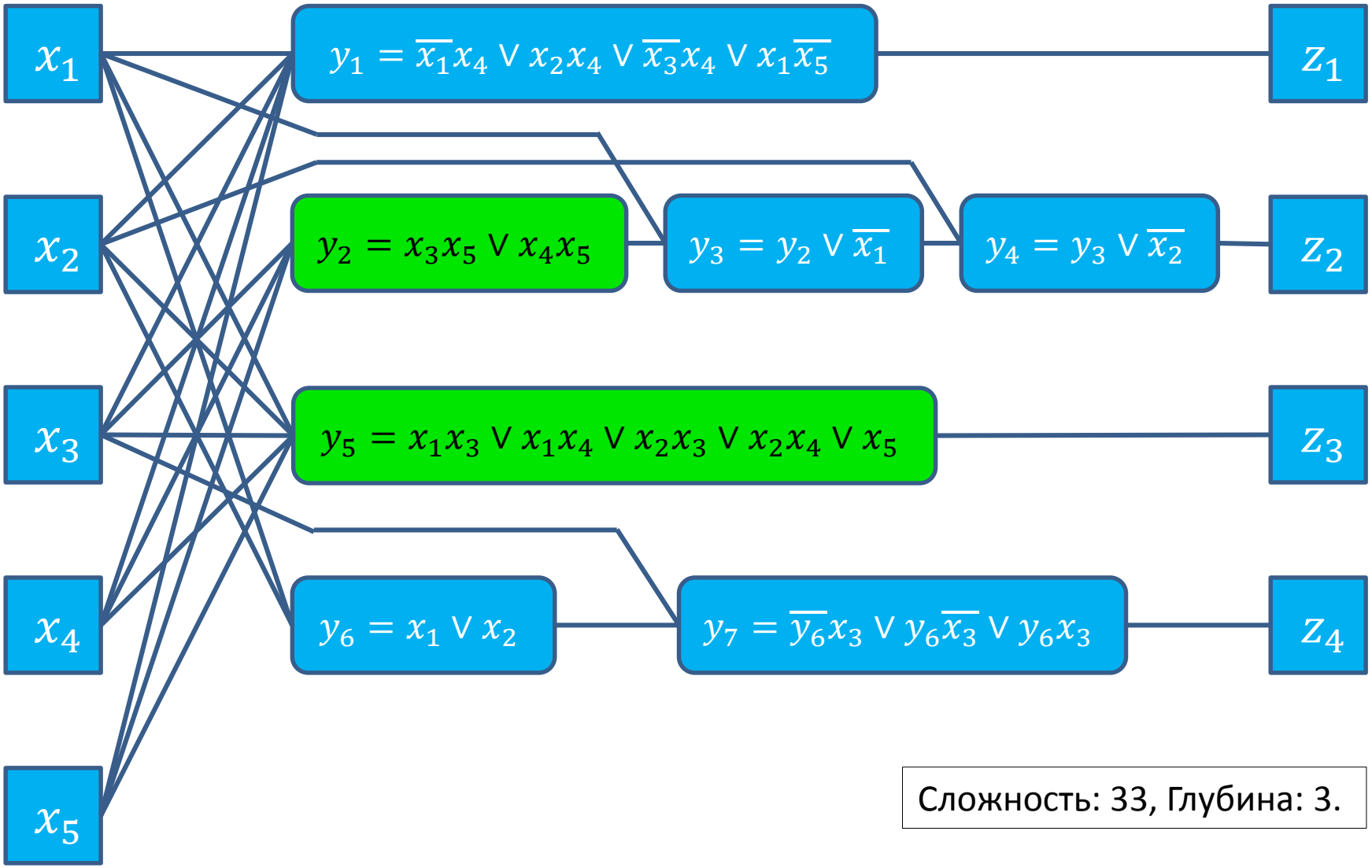
Сложность: 33, Глубина: 3.

Пример разложения (DECOMPOSITION)



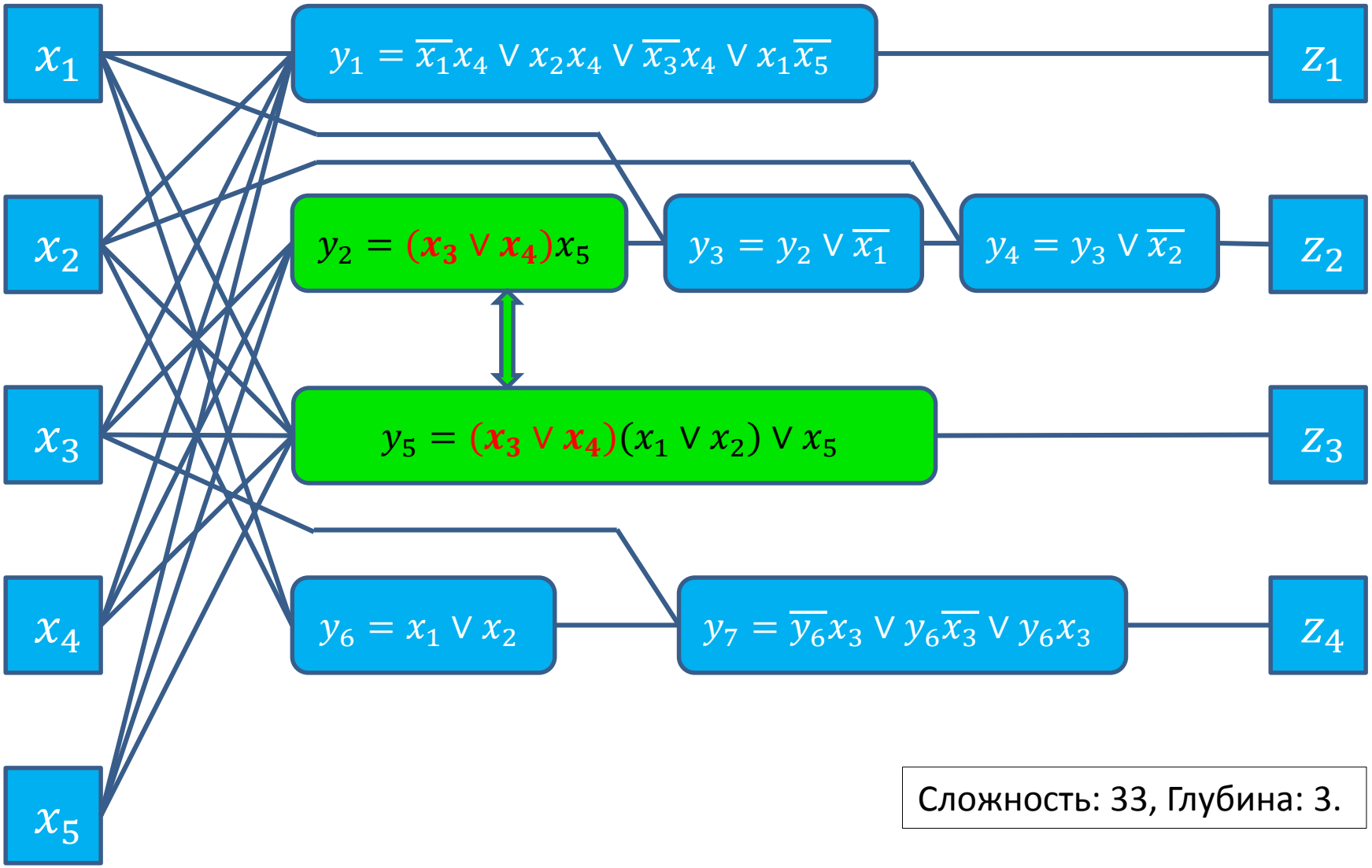
Сложность: 33-8+3+4=32, Глубина: 3.

Пример извлечения (EXTRACTION)



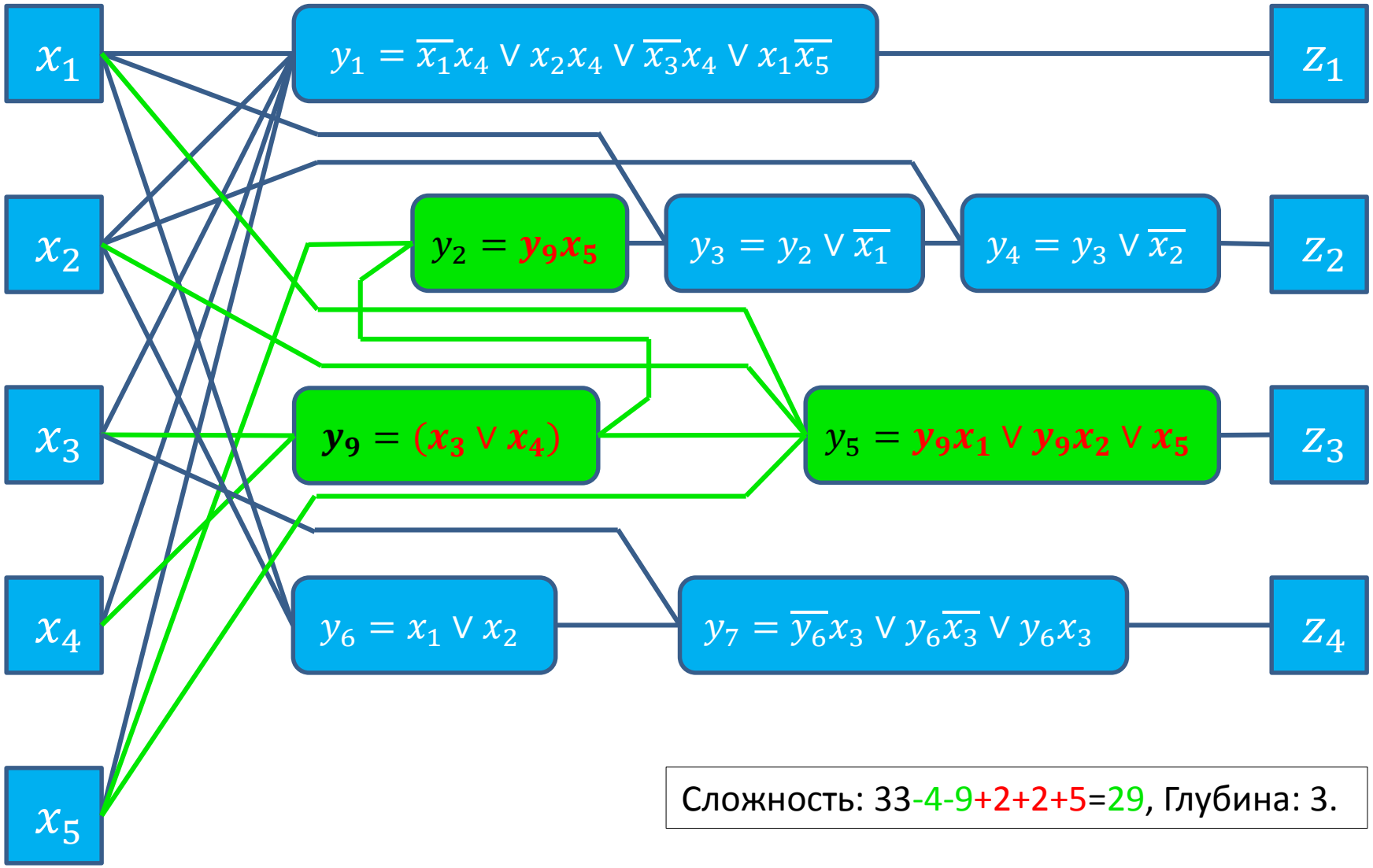
Сложность: 33, Глубина: 3.

Пример извлечения (EXTRACTION)



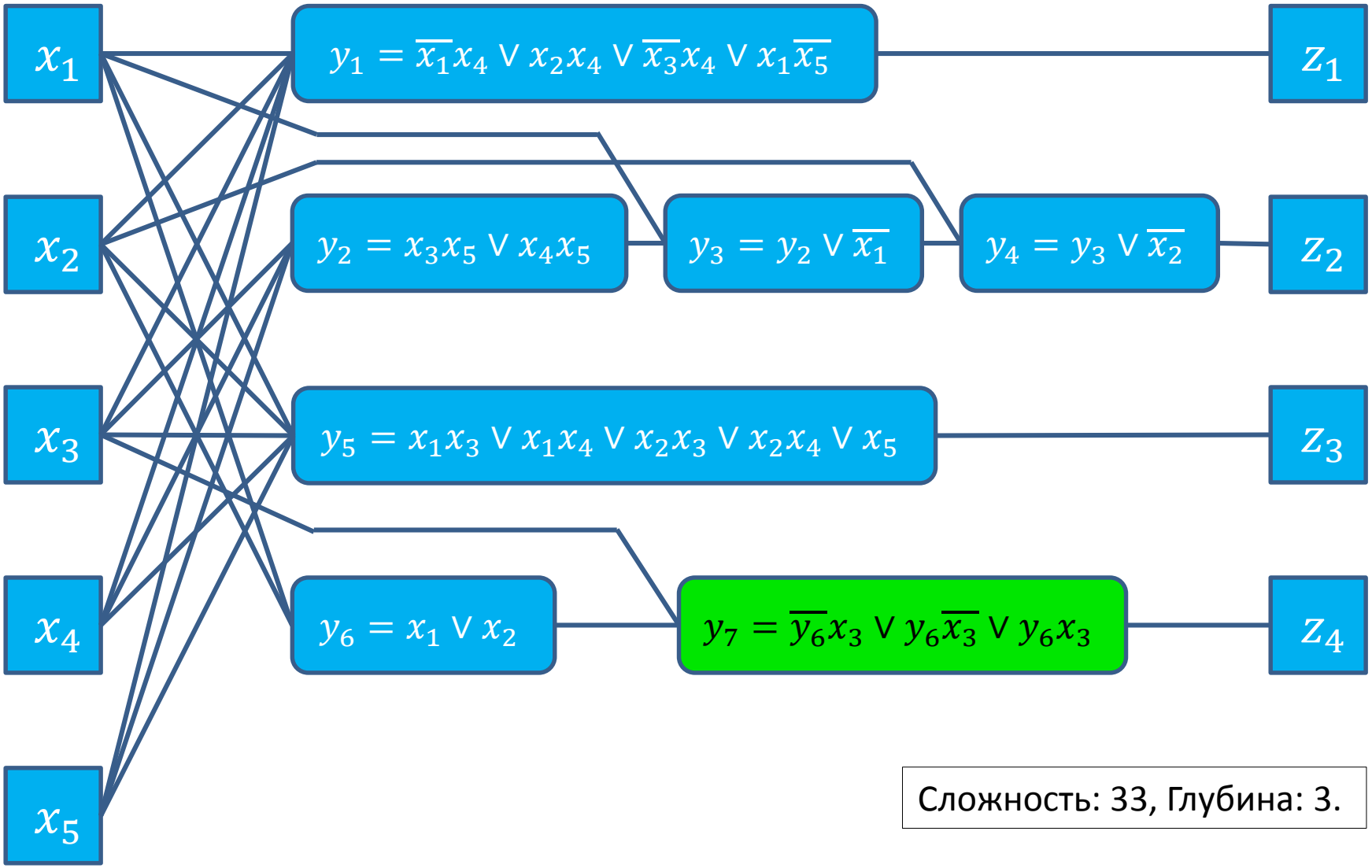
Сложность: 33, Глубина: 3.

Пример извлечения (EXTRACTION)



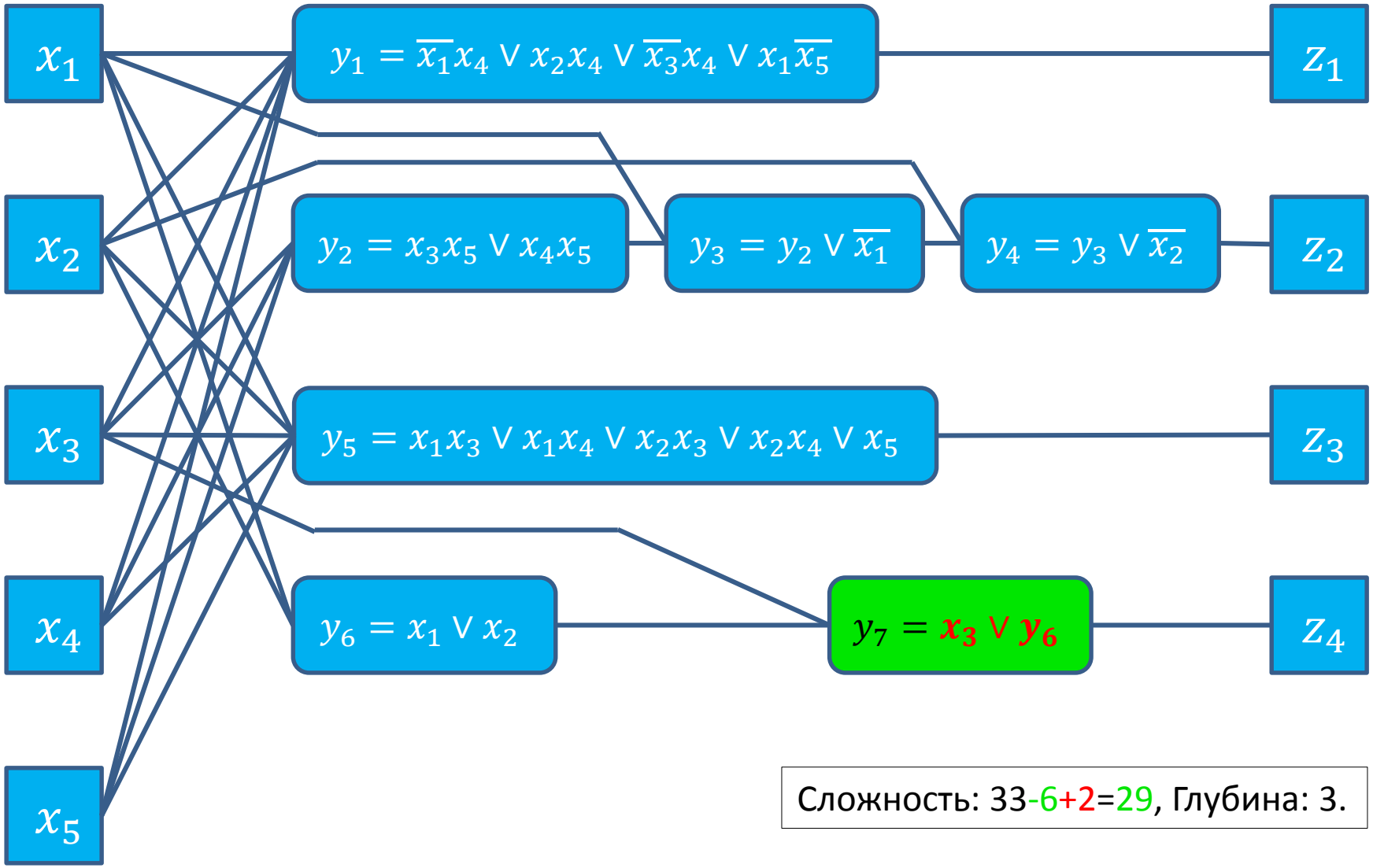
Сложность: 33-4-9+2+2+5=29, Глубина: 3.

Пример упрощения (SIMPLIFICATION)



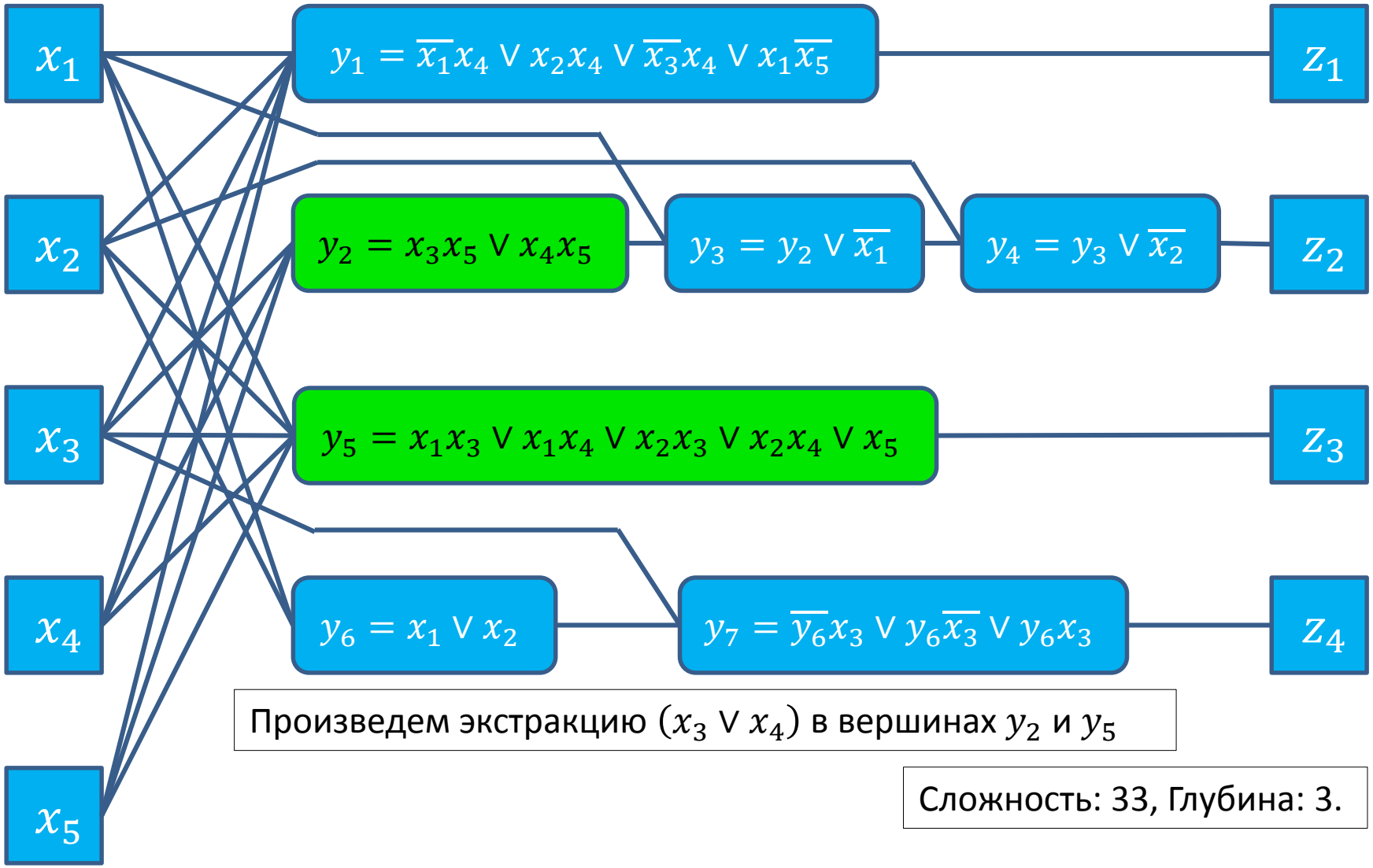
Сложность: 33, Глубина: 3.

Пример упрощения (SIMPLIFICATION)



Сложность: 33-6+2=29, Глубина: 3.

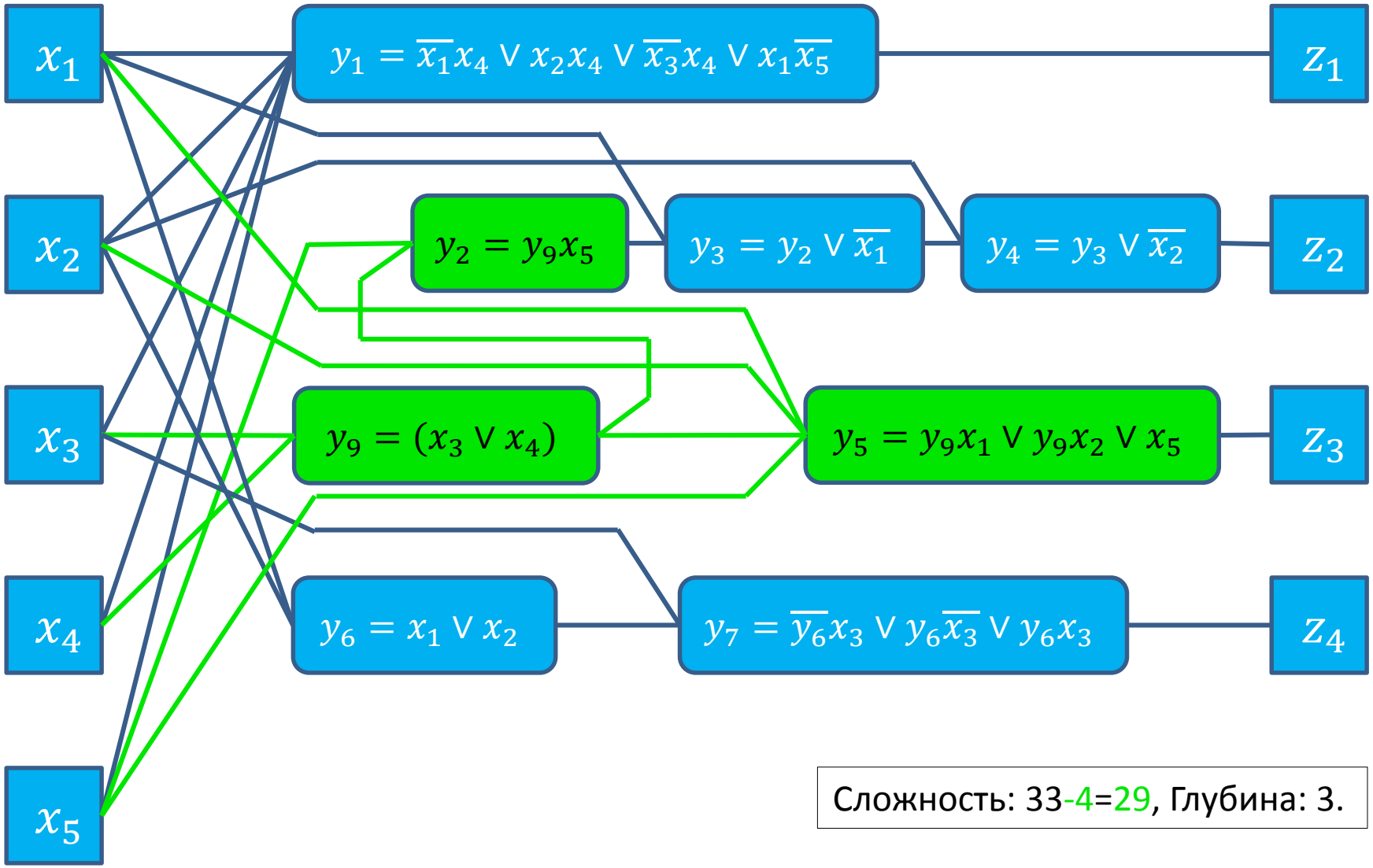
Пример подстановки (SUBSTITUTION)



Произведем экстракцию $(x_3 \vee x_4)$ в вершинах y_2 и y_5

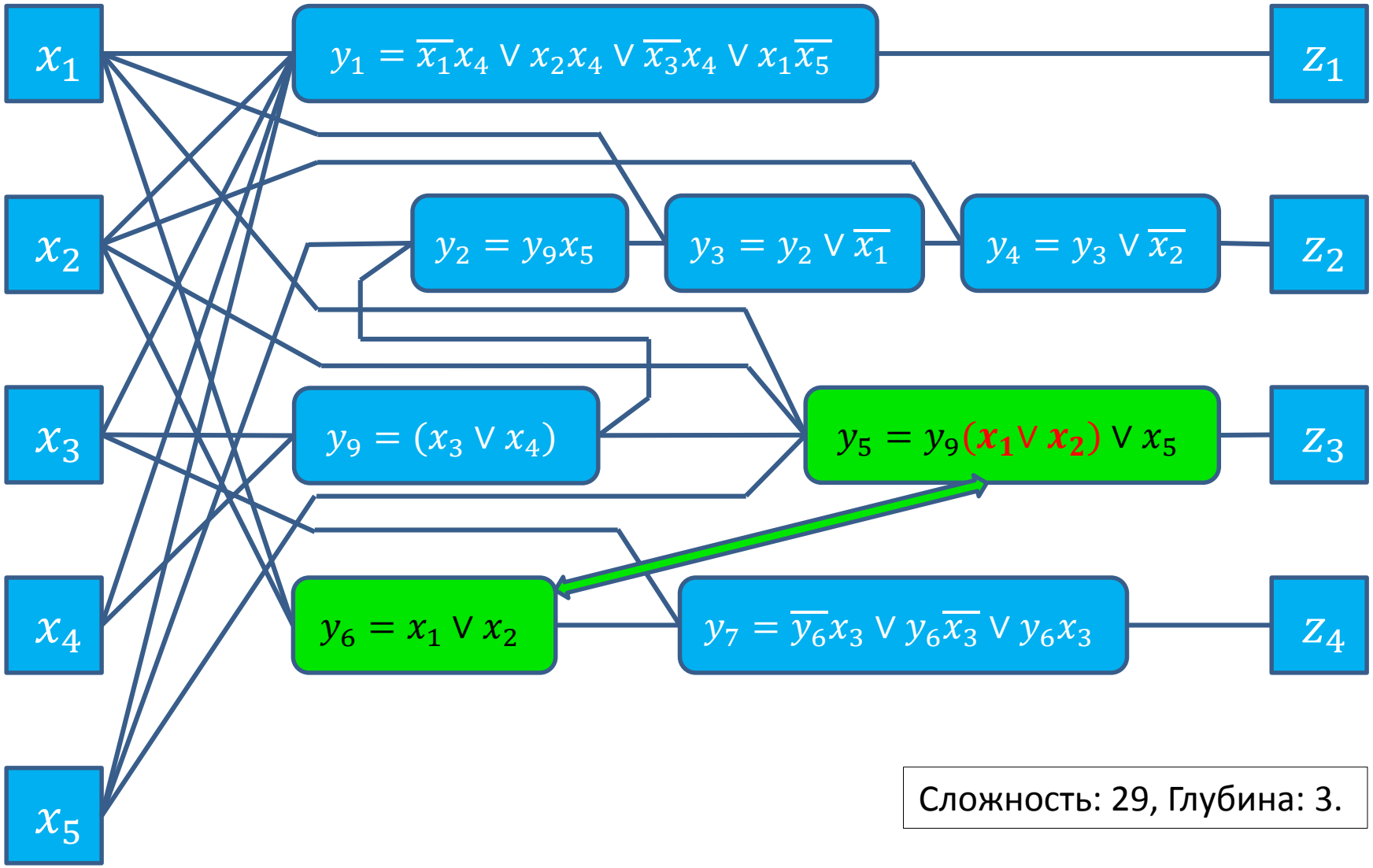
Сложность: 33, Глубина: 3.

Пример подстановки (SUBSTITUTION)



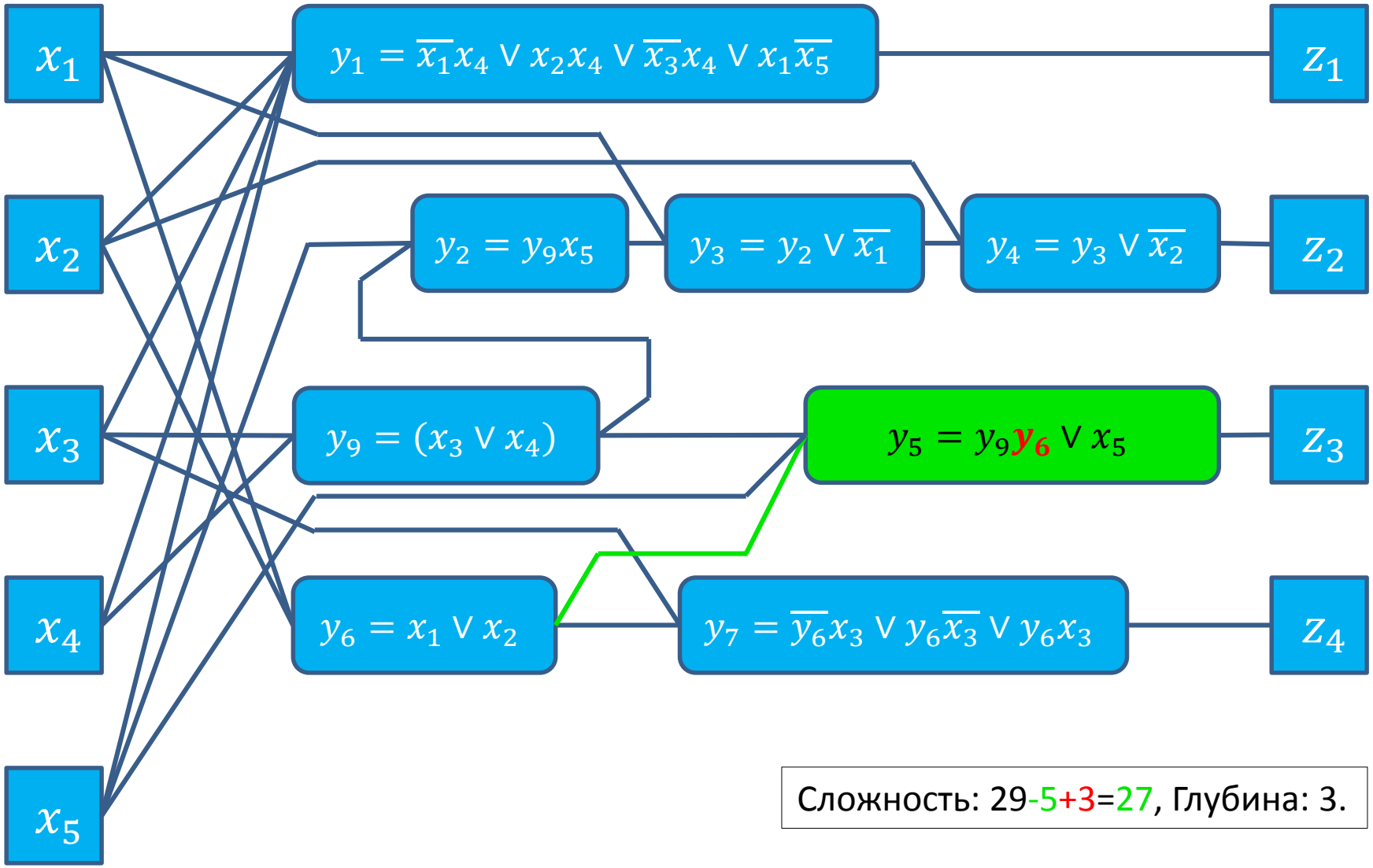
Сложность: 33-4=29, Глубина: 3.

Пример подстановки (SUBSTITUTION)



Сложность: 29, Глубина: 3.

Пример подстановки (SUBSTITUTION)



Сложность: 29-5+3=27, Глубина: 3.

Разложение на основе деления (Factorization)

- Для заданной ДНФ требуется найти хорошее разложение
  - Нужно найти хороший делитель
  - Нужен алгоритм, выполняющий деление
    - Возвращает делимое и остаток

- Почему деление?
  - Используется во многих операциях преобразования КЛС:
    - факторизация
    - разложение
    - подстановка
    - извлечение

Операция деления

Определение:

Операция OP называется делением, если для двух входных ДНФ F и G она порождает ДНФ H и R (<H,R> = OP(F,G)), такие что

$$F = GH + R.$$

G - делитель

H - частное

R - остаток

Определение:

ДНФ F называется алгебраической, если ни одна ее ЭК не вложена в другую

Определение:

Произведение ДНФ GH называется алгебраическим, если G и H алгебраические ДНФ и не имеют общих переменных. Если OP порождает алгебраическое произведение, то такая операция называется алгебраическим делением, которое будем обозначать F // G).

В противном случае деление будем называть булевым и обозначать $F \div G$.

Деление примеры (f = gh+r)

Пример:

$f = ad + ae + bcd + j$
$g_1 = a + bc$
$g_2 = a + b$

- Алгебраическое деление:
  - $f // a = d + e, r = bcd + j$
  - $f // (bc) = d, r = ad + ae + j$
  - (Кроме того, $f // a = d$ или $f // a = e$, деление не уникально)
  - $h_1 = f // g_1 = d, r_1 = ae + j$

- Булево деление:
  - $h_2 = f \div g_2 = (a + c)d, r_2 = ae + j.$
    То есть: $f = (a+b)(a+c)d + ae + j$

Деление

Определение:

G является алгебраическим множителем F, если существует алгебраическая ДНФ H, такая что

F = GH (произведение алгебраическое).

Определение:

G является булевым множителем F, если существует ДНФ H, такая что

F = GH (булево произведение).

Пример:

f = ac + ad + bc + bd

(a+b) алгебраический множитель f, так как f = (a+b)(c+d)

f = ~ab + ac + bc

(a+b) булевый множитель f, так как f = (a+b)(~a+c)

Зачем нужна алгебраическая модель?

- Требуется широкий спектр возможны преобразований КЛС
  - существуют быстрые алгоритмы для алгебраических операций
- Можно рассматривать функции алгебры логики как обычные полиномы
  - Эффективность хранения
  - существуют быстрые методы для работы с полиномами
- Возможна потеря оптимальности, но часто результат довольно хороший
- Можно использовать многократно и чередовать с булевыми методами
- В некоторых специальных случаях можно обобщать алгебраические методы при помощи булевых

«Слабое» деление

«Слабое» деление – специальный вид алгебраического деления.

Определение:

Пусть даны две алгебраические ДНФ F и G. Тогда деление называется «слабым», если

- оно алгебраическое
- остаток R содержит минимально возможное количество ЭК.

Частное H, получающееся в результате «слабого» деления обозначим через F/G.

Утверждение: Для заданных ДНФ F и G, ДНФ H и R, порождаемые в результате «слабого» деления, являются единственными.

Алгоритм «слабого» деления

```
ALGORITHM WEAK_DIV(F,G) {   // G={g1,g2,...}, f=(f1,f2,...}
  foreach gi {
    Vgi=∅
    foreach fj {
      if(fj contains all literals of gi) {
        vij=fj - literals of gi
        Vgi=Vgi ∪vij
      }
    }
  }

  H = ⋂ Vgi
  R = F - GH
  return (H,R);
}
```

Пример WEAK_DIV

Пример:

F = ace + ade + bc + bd + be +a’b + ab

G = ae + b

V^{ae}= c + d

V^{b} = c + d + e + a’ + a

H = c + d = F/G

R = be + a’b + ab

$H = \cap V^{g_i}$

R = F \ GH

F = (ae + b)(c + d) + be + a’b + ab

Деление – как найти делитель?

- Weak_Div позволяет разделить ДНФ на заданный делитель
- Но как найти «хороший» делитель?
  - Среди множества алгебраических делителей
  - Как обобщить на случай булевых делителей
- Задача:
  - Для заданного набора булевых функций $\{F_i\}$ найти общие алгебраические делители, которые можно использовать в алгоритме «слабого» деления.

Основные делители

Определение:

ДНФ является приведенной, если никакая ее ЭК не является ее множителем (то есть у ЭК нет общих литералов).

ab + c приведенная

ab + ac и abc не является приведенной

Примечание: приведенная ДНФ должна содержат как минимум две ЭК.

Определение:

Основные делители ДНФ F задаются следующим множеством:

$D(F) = \{F/c \mid c - ЭК\}$.

Ядровые делители и их образы

Определение:

Ядро ДНФ F задается следующим множеством ДНФ:

$$K(F) = \{G \mid G \in D(F) \text{ и } G \text{ приведенная}\}.$$

Другими словами, ядро это множество приведенных основных делителей ДНФ F. Отдельные элементы ядра будем называть ядровыми.

Определение:

ЭК c, которая порождает делитель K = F/c, называется образом делителя K.

C(F) будем использовать для обозначения множества образов ДНФ F.

Пример

Пример:

x = adf + aef + bdf + bef + cdf + cef + g

= (a + b + c)(d + e)f + g

Ядровые делители	образы
a+b+c	df, ef
d+e	af, bf, cf
(a+b+c)(d+e)f+g	1

Основная теорема

Теорема:

Если ДНФ F и G обладают следующим свойством:

$$\forall k_F \in K(F), \forall k_G \in K(G) \rightarrow | k_G \cap k_F | \leq 1$$

(k_G и k_F имеют не более одного общего ядрового делителя),

Тогда F и G не имеют общих нетривиальных алгебраических делителей (i.e. with more than one cube).

Замечание:

Если ядра всех рассматриваемых функций не имеют нетривиальных пересечений, то алгебраическими делителями являются только ЭК.

Ранг ядровых делителей

Определение:

Ядровой делитель имеет ранг 0 (K^0), если он не содержит других ядровых делителей, кроме себя.

Ядровой делитель имеет ранг n (K^n), если он содержит хотя бы один делитель ранга (n-1) и не содержит других делителей ранга n (кроме себя).

- $K^n(F)$ – множество всех делителей ранга k или меньше.
- $K^0(F) \subset K^1(F) \subset K^2(F) \subset \ldots \subset K^n(F) \subset K(F)$.
- Делители ранга n: $K_n(F)=K^n(F) \setminus K^{n-1}(F)$

Пример:

$F = (a + b(c + d))(e + g)$

$k_1 = a + b(c + d) \in K^1$, $\notin K^0$ ==> ранг = 1

$k_2 = c + d \in K^0$

$k_3 = e + g \in K^0$

Алгоритм построения ядра

```
Algorithm KERNEL(j, G) {
  R = ∅
  if(CUBE_FREE(G)) R = {G}
  for(i=j+1,...,n) {
    if(l_i appears only in one term)        continue
    if(∃k ≤ i, l_k ∈ all cubes of G/l_i)    continue
    R = R ∪ KERNEL(i,MAKE_CUBE_FREE(G/l_i)
  }
  return R
}
```

MAKE_CUBE_FREE(F) removes algebraic cube factor from F

Алгоритм построения ядра

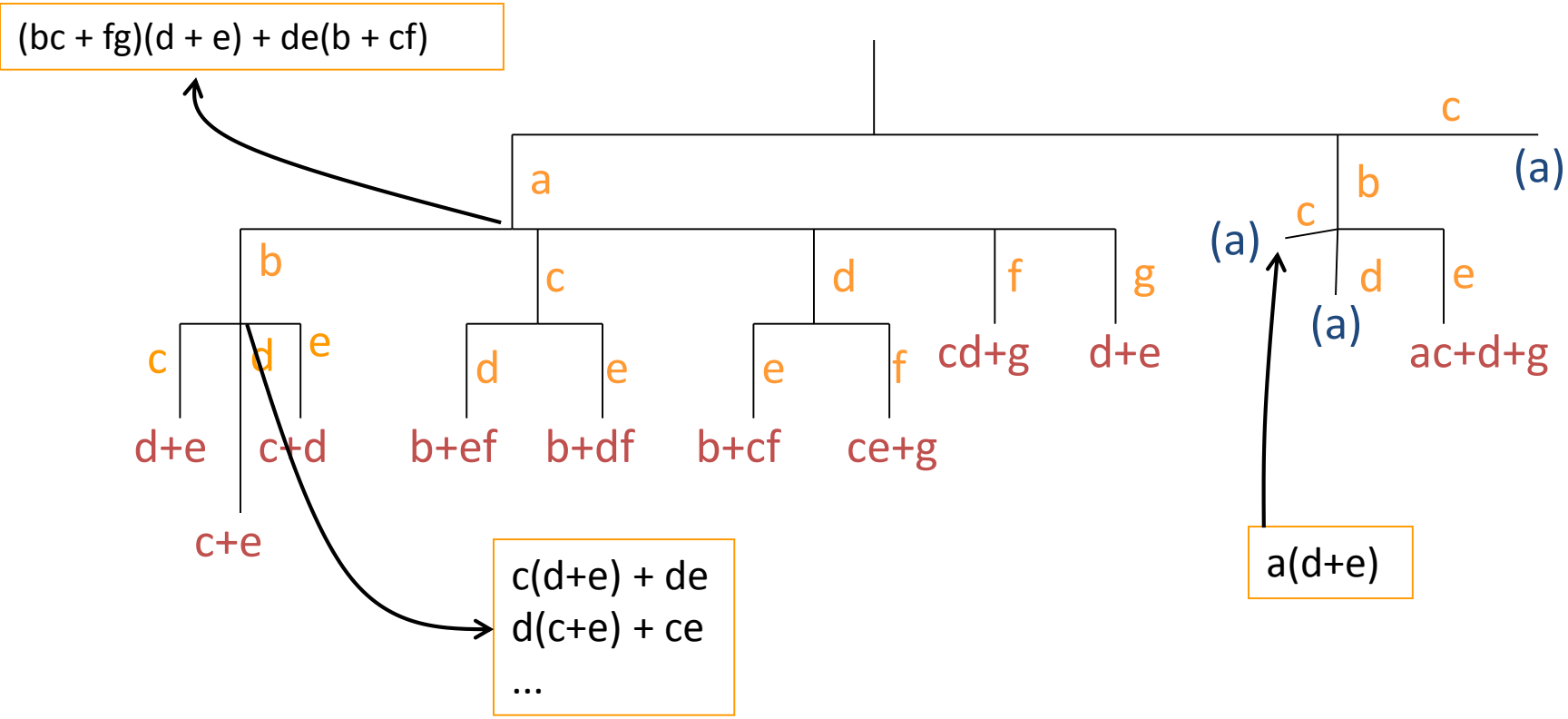
KERNEL(0, F) возвращает все ядровые делители F.

Замечания:

- Проверка ($\exists k \leq i, l_k \in$ all cubes of G/l_i) увеличивает сложность сложность алгоритма, но гарантирует, что каждый делитель просматривается только один раз.
- Алгоритм также позволяет найти образы делителей.

Пример построения ядра

abcd + abce + adfg + aefg + adbe + acdef + beg



Пример построения ядра

Образы	Ядровые делители
1	a((bc + fg)(d + e) + de(b + cf))) + beg
a	(bc + fg)(d + e) + de(b + cf)
ab	c(d+e) + de
abc	d + e
abd	c + e
abe	c + d
ac	b(d + e) + def
acd	b + ef

Примечание: f/bc = ad + ae = a(d + e)

Факторизация

```
Algorithm FACTOR(F) {
  if(F has no factor) return F
     // e.g. if |F|=1, or F is an OR of single literals
     // or of no literal appears more than once
  D      = CHOOSE_DIVISOR(F)
  (Q,R)  = DIVIDE(F,D)
  return FACTOR(Q)·FACTOR(D) + FACTOR(R)   // recur
}
```

- Можно использовать разные эвристики для CHOOSE_DIVISOR
- Различные алгоритмы DIVIDE можно использовать

Пример и проблемы факторизации

Example:

F = abc + abd + ae + af + g
D = c + d
Q = ab
P = ab(c + d) + ae + af + g
O = ab(c + d) + a(e + f) + g

«O» не является оптимальными, так как не максимально факторизовано.

Альтернативная факторизация:

a(b(c + d) + e + f) + g

Обозначения:

F = исходная функция,
D = делитель,
Q = частное,
P = частичная факторизованная форма,
O = финальная факторизованная форма, полученная алгоритмомFACTOR.
Рассматриваются только алгебраические операции.

Данная проблема возникает, когда

- частное Q является ЭК и
- Некоторые литералы Q попадают в остаток R.

Решение проблем факторизации

Решение проблемы:

- Проверить частное Q: если не ЭК, то остановиться, иначе
- Выбираем литерал l_1 в Q, который встречается в максимальном количестве ЭК F.
- Делим F на l_1 для получения нового делителя D_1.
  Теперь, F имеет новую частично факторизованную форму
  $$(l_1)(D_1) + (R_1)$$
  и литерал l_1 не встречается в R_1.

Примечание:

Новый делитель D_1 содержит D как делитель самого себя, так как l_1 является литералом Q. Когда происходит рекурсивная факторизация D_1, делитель D может быть вновь обнаружен.

Вторая проблема факторизации

Пример:

$F = ace + ade + bce + bde + cf + df$

$D = a + b$

$Q = ce + de$

$P = (ce + de)(a + b) + (c + d)\, f$

$O = e(c + d)(a + b) + (c + d)f$

Обозначения:

F = исходная функция,
D = делитель,
Q = частное,
P = частичная факторизованная форма,
O = финальная факторизованная форма

«O» не является максимально факторизованной, так как (c + d) является общим слагаемым для произведения e(c + d)(a + b) и для остатка (c + d)f.

Финальная форма должна иметь следующий вид:

$(c+d)(e(a + b) + f)$

Решение второй проблемы

Решение проблемы:

- Поменять местами D и Q!
- Сделать Q приведенной, при этом получаем Q_1
- Получить новый делитель D_1, разделив F на Q_1
- Если D_1 приведенная, то частичная факторизованная форма имеет вид $F = (Q_1)(D_1) + R_1$ и может быть рекурсивно факторизована по Q_1, D_1, и R_1
- Иначе, пусть $D_1 = cD_2$ и $D_3 = Q_1D_2$. Имеем частичную факторизацию $F = cD_3 + R_1$. Можно рекурсивно факторизовать по D_3 и R_1.

Улучшенный алгоритм факторизации

```
Algorithm GFACTOR(F, DIVISOR, DIVIDE)
  D = DIVISOR(F)
  if(D = 0) return F
  Q = DIVIDE(F,D)
  if (|Q| = 1) return LF(F, Q, DIVISOR, DIVIDE)
  Q = MAKE_CUBE_FREE(Q)
  (D, R) = DIVIDE(F,Q)
  if (CUBE_FREE(D)) {
    Q = GFACTOR(Q, DIVISOR, DIVIDE)
    D = GFACTOR(D, DIVISOR, DIVIDE)
    R = GFACTOR(R, DIVISOR, DIVIDE)
    return Q · D + R
  }
   else {
     C = COMMON_CUBE(D)
     return LF(F, C, DIVISOR, DIVIDE)
  }
```

Улучшенный алгоритм факторизации

```
Algorithm LF(F, C, DIVISOR, DIVIDE) {
  L = BEST_LITERAL(F, C)  // most frequent
  (Q, R) = DIVIDE(F, L)
  C = COMMON_CUBE(Q)      // largest one
  Q = CUBE_FREE(Q)
  Q = GFACTOR(Q, DIVISOR, DIVIDE)
  R = GFACTOR(R, DIVISOR, DIVIDE)
  return    L · C · Q + R
}
```

Улучшенный выбор делителя

- Различные варианты факторизации могут быть получены за счет выбора алгоритма генерации делителя DIVISOR и алгоритма деления DIVIDE.

- CHOOSE_DIVISOR:
  - LITERAL – литерал с наибольшей частотой
  - QUICK_DIVISOR – первый ядровой делитель нулевого ранга
  - BEST_DIVISOR – лучший ядровой делитель

- DIVIDE:
  - Algebraic Division
  - Boolean Division

Примеры факторизации

$x = ac + ad + ae + ag + bc + bd + be + bf + ce + cf + df + dg$

LITERAL_FACTOR:

$x = a(c + d + e + g) + b(c + d + e + f) + c(e + f) + d(f + g)$

QUICK_FACTOR:

$x = g(a + d) + (a + b)(c + d + e) + c(e + f) + f(b + d)$

GOOD_FACTOR:

$(c + d + e)(a + b) + f(b + c + d) + g(a + d) + ce$

Пример: QUICK_FACTOR

QUICK_FACTOR использует

- GFACTOR,
- QUICK_DIVISOR
- WEAK_DIV.

x = ae + afg + afh + bce + bcfg + bcfh + bde + bdfg + bcfh

D = c + d -> первый ядровой делитель ранга 0
Q = x/D = b(e + f(g + h)) -> слабое деление
Q = e + f(g + h) -> приведение
(D, R) = WEAK_DIV(x, Q) -> второе деление
D = a + b(c + d)
x = QD + R R = 0

x = (e + f(g + h)) (a + b(c + d))

Разложение

- Разложение похоже на факторизацию, но со следующими особенностями:
  - Делители добавляются в КЛС в качестве новых вершин.
  - Новые вершины могут быть использованы в качестве подфункций другими функциями

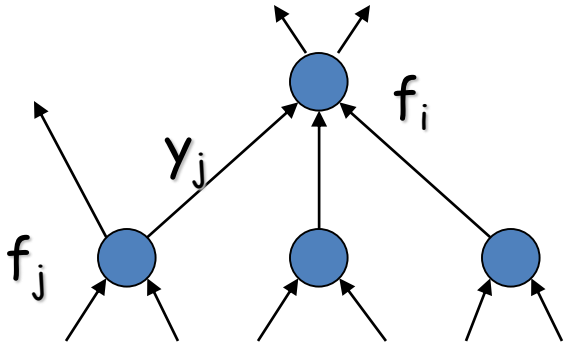
```
Algorithm DECOMP(f_i) {
  k = CHOOSE_KERNEL(f_i)
  if (k == 0) return
  f_{m+j} = k                              // create new node m + j
  f_i   = (f_i/k)y_{m+j}+(f_i/k')y'_{m+j}+r // change node i using new
                                             // node for kernel
  DECOMP(f_i)
  DECOMP(f_{m+j})
}
```

По аналогии с факторизацией можно определить:

QUICK_DECOMP: выбирает ядровой делитель ранга ноль.

GOOD_DECOMP: выбирает лучший ядровой делитель.

Подстановка (Re-substitution)



- Идея: Существующая вершина КЛС может быть «полезным» делителем для другой вершины. Если это действительно так, то можно изменить структуру схемы. При этом сложность не возрастает, но может возрасти задержка.

- Алгебраическая подстановка представляет процесс деления функции f_i в вершине i при помощи функции f_j (или ее отрицания f'_j) в вершине j. При этом, если f_j алгебраический делитель f_j, то f_i изменяется на

$$f_i = qy_j + r \quad (\text{или } f_i = q_1 y_j + q_0 y'_j + r)$$

- На практике это преобразование применяется к каждой паре вершин в схеме. Для КЛС из n вершин $\Rightarrow O(n^2)$ делений.

Извлечение

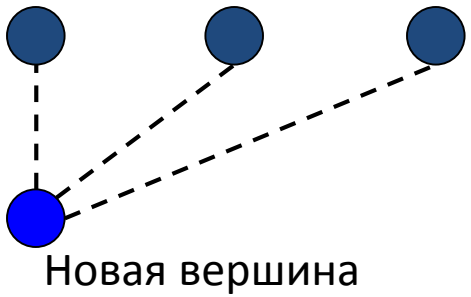
- Напоминание: Извлечение это процесс обнаружения общих подфункций в КЛС и оптимизация ее структуры с их помощью.
- При этом используется разложение и подстановка.

```
Algorithm EXTRACT
  foreach node n {
     DECOMP(n)          // decompose all network nodes
   }
  foreach node n {
    RESUB(n)           // resubstitute using existing nodes
  }
  ELIMINATE_NODES_WITH_SMALL_VALUE
}
```

Извлечение

Извлечение на базе ядра:

1. Найти все ядра всех функций
2. Найти пересечение ядер с наилучшим «весом»
3. Создать новую вершину для результирующей функции
4. Произвести подстановку новой функции
5. Повторить 1,2,3,4 пока сложность схемы улучшается



Извлечение - пример

$f_1 = ab(c(d + e) + f + g) + h$
$f_2 = ai(c(d + e) + f + j) + k$
(используются только ядровые делители нулевого ранга)

1. Извлечение: $K^0(f_1) = K^0(f_2) = \{d + e\}$
$K^0(f_1) \cap K^0(f_2) = \{d + e\}$

$l = d + e$ $f_1 = ab(cl + f + g) + h$
$f_2 = ai(cl + f + j) + k$

2. Извлечение: $K^0(f_1) = \{cl + f + g\}$; $K^0(f_2) = \{cl + f + j\}$
$K^0(f_1) \cap K^0(f_2) = cl + f$

$m = cl + f$ $f_1 = ab(m + g) + h$
$f_2 = ai(m + j) + k$

Больше нет нетривиальных пересечений ядровых делителей!

3. Извлечение ЭК:

$n = am$ $f_1 = b(n + ag) + h$
$f_2 = i(n + aj) + k$

Извлечение – задача о покрытии

- Строится специальная матрица M = R × C
  - Строки матрицы соответствуют паре функция и образ делителя
  - Столбцы соответствуют ЭК, встречающимся в делителях
  - m_{ij} = ЭК функции, который содержит ЭК делителя
  - m_{ij} = 0, если нет вложения

- Задача о прямоугольном покрытии:
  - Найти прямоугольную подматрицу M’ = R’ × C’, где R’ $\subseteq$ R, C’ $\subseteq$ C, и $m'_{ij} \neq 0$
  - Новый делитель d строится по подматрице M’
  - Производится экстракция по d

Пример

F = af + bf + ag + cg + ade + bde + cde

G = af + bf + ace + bce

H = ade + cde

Ядровой делитель/образ:

F: (de+f+g)/a

(de + f)/b

(a+b+c/de

(a + b)/f

(de+g)/c

(a+c)/g

G: (ce+f)/{a,b}

(a+b)/{f,ce}

H: (a+c)/de

$M =$

		a	b	c	ce	de	f	g
F	a					ade	af	ag
F	b					bde	bf	
F	de	ade	bde	cde				
F	f	af	bf					
F	c					cde		cg
F	g	ag		cg				
G	a				ace		af	
G	b				bce		bf	
G	ce	ace	bce					
G	f	af	bf					
H	de	ade		cde				

Пример

F = af + bf + ag + cg + ade + bde + cde

G = af + bf + ace + bce

H = ade + cde

- Выбор подматрицы M’
- Извлечение X

  X = a + b

  F = fx + ag + cg + dex + cde

  G = fx + cex

  H =ade + cde
- Обновление M

$$M = \begin{array}{cc|ccccccc} & & a & b & c & ce & de & f & g \\ \hline F & a & & & & & ade & af & ag \\ F & b & & & & & bde & bf & \\ F & de & ade & bde & cde & & & & \\ F & f & af & bf & & & & & \\ F & c & & & & & cde & & cg \\ F & g & ag & & cg & & & & \\ G & a & & & & ace & & af & \\ G & b & & & & bce & & bf & \\ G & ce & ace & bce & & & & & \\ G & f & af & bf & & & & & \\ H & de & ade & & cde & & & & \end{array}$$

Вес подматрицы

- Число литералов до извлечение – число литералов после извлечения:

$$V(R', C') = \sum_{i \in R, j \in C} v_{ij} - \sum_{i \in R'} w_i^r - \sum_{j \in C} w_j^c$$

v_{ij}: число литералов в ЭК m_{ij}

w_i^r: (число литералов в образе делителя в строке i) + 1

w_j^c: число литералов в ЭК столбца j

- Пример:

V = 20 - 10 - 2 = 8

			a	b	c	ce	de	f	g
	F	a					ade	af	ag
	F	b					bde	bf	
	F	de	ade	bde	cde				
	F	f	af	bf					
M =	F	c					cde		cg
	F	g	ag		cg				
	G	a				ace		af	
	G	b				bce		bf	
	G	ce	ace	bce					
	G	f	af	bf					
	H	de	ade		cde				