

Верификация параллельных программных и аппаратных систем



Курс лекций

Шошмина Ирина Владимировна

Карпов Юрий Глебович



План курса

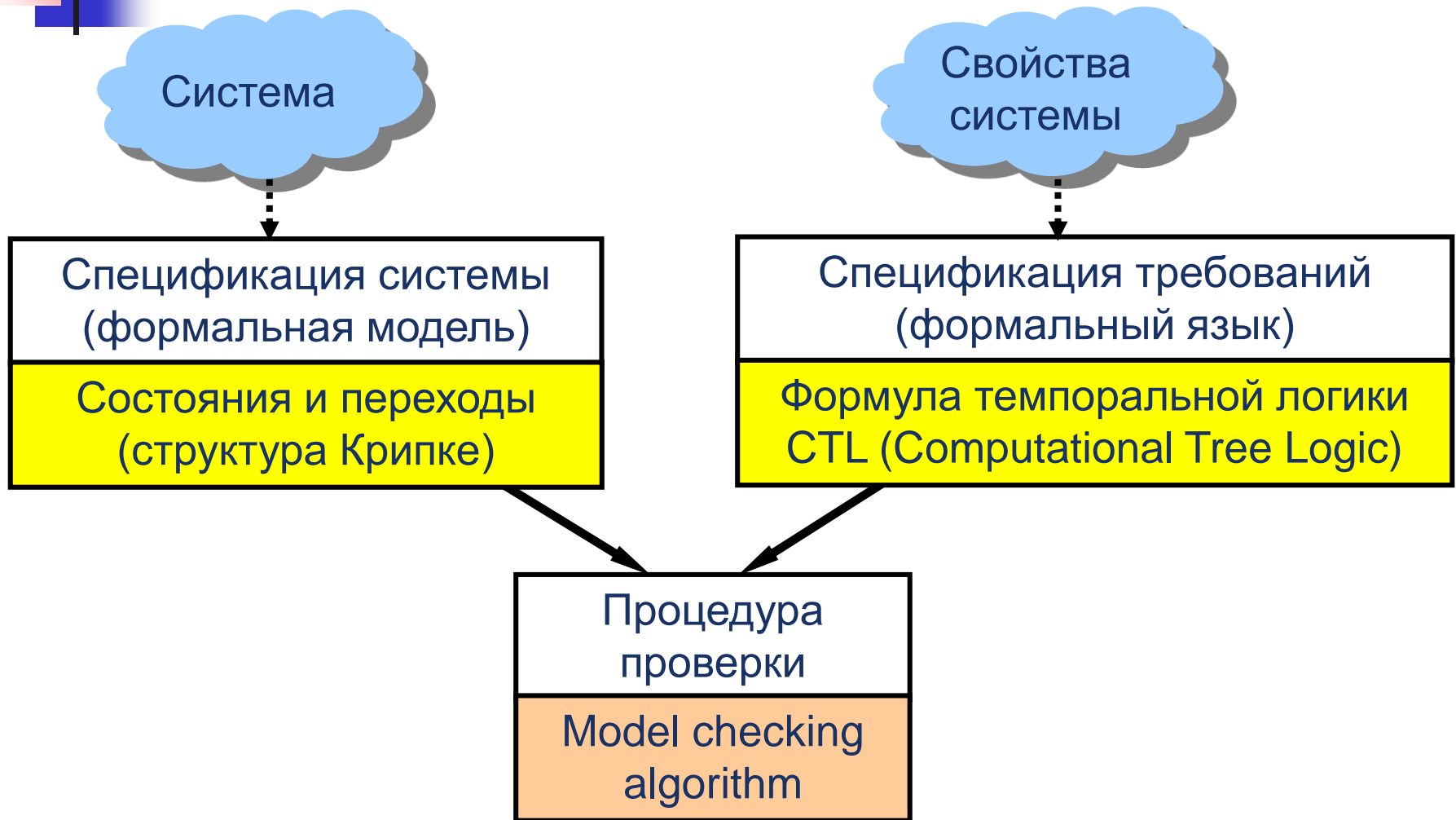
1. Введение
2. Метод Флойда-Хоара доказательства корректности программ
3. Исчисление взаимодействующих систем (CCS) Р.Милнера
4. Темпоральные логики
5. Алгоритм model checking для проверки формул CTL
6. Автоматный подход к проверке выполнения формул LTL
7. Система верификации Spin и язык Promela. Примеры верификации
8. Структура Крипке как модель реагирующих систем
9. Темпоральные свойства систем
10. Применения метода верификации model checking
11. Символьная верификация: BDD
12. Символьный алгоритм верификации model checking
13. Количественный анализ дискретных систем при их верификации
14. Верификация систем реального времени
15. Консультации по курсовой работе



Лекция 11

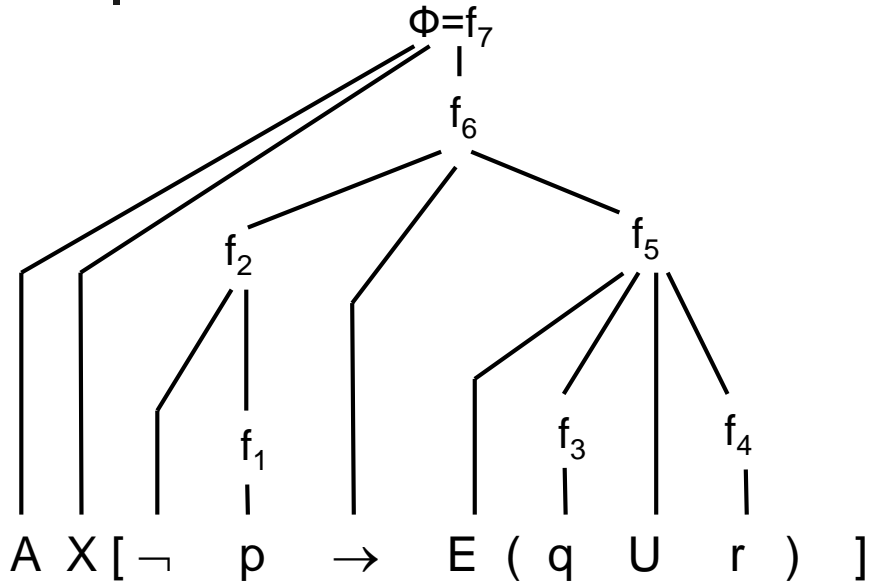
Символьная верификация: BDD

Верификация методом Model checking (CTL)

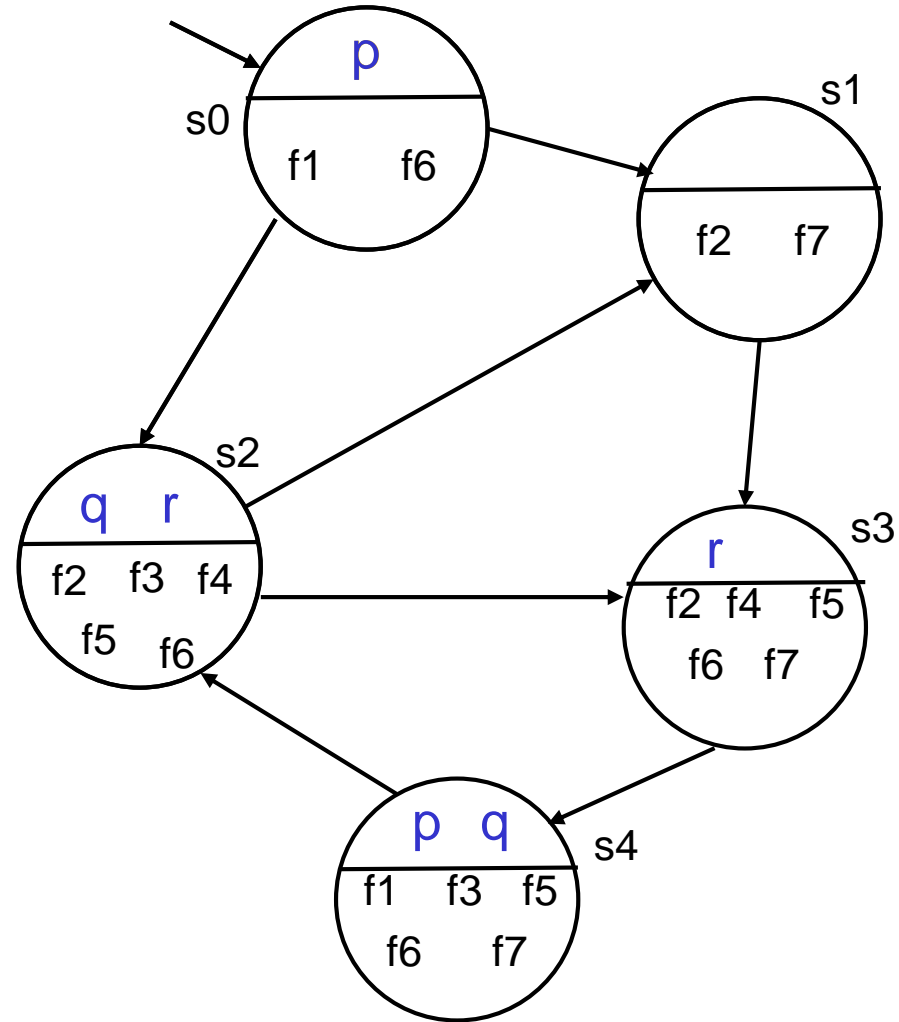


Алгоритм маркировки для CTL формул. Конечное число состояний

$A X [\neg p \rightarrow E (q U r)]$



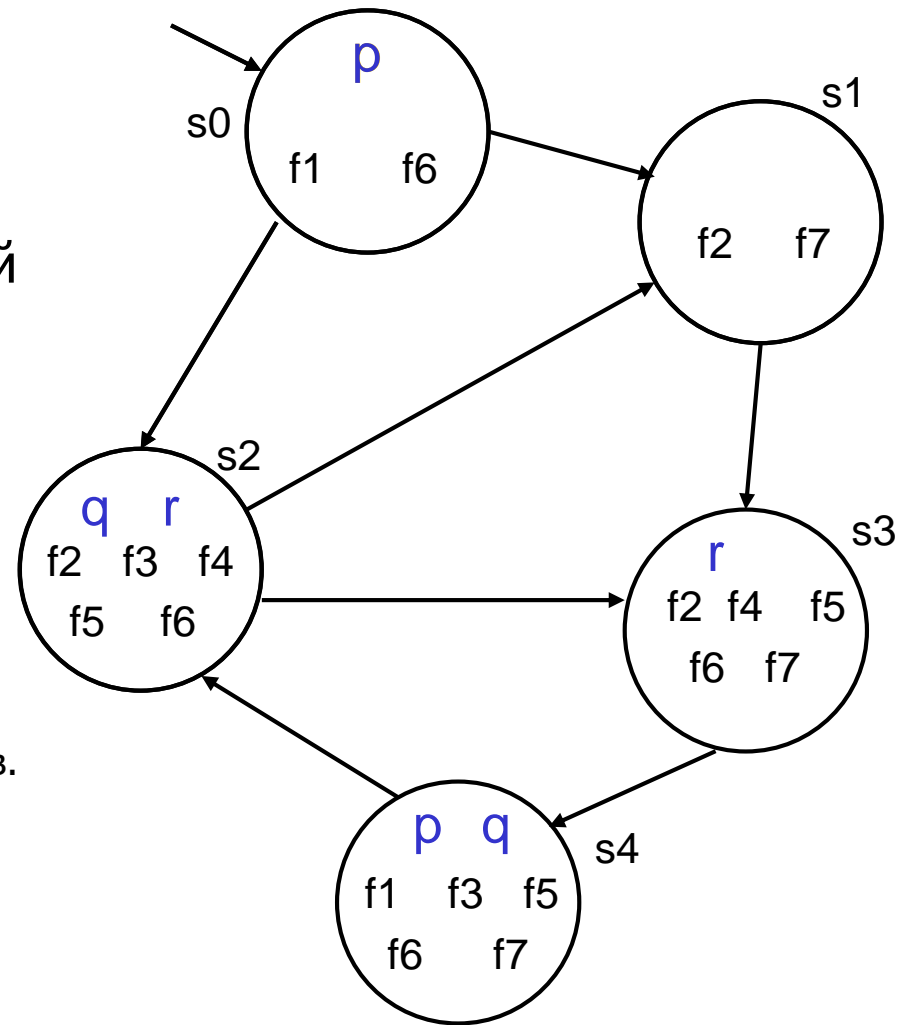
$f_1 = p$	$f_1 : \{ s_0, s_4 \}$
$f_2 = \neg f_1$	$f_2 : \{ s_1, s_2, s_3 \}$
$f_3 = q$	$f_3 : \{ s_2, s_4 \}$
$f_4 = r$	$f_4 : \{ s_2, s_3 \}$
$f_5 = E(f_3 U f_4)$	$f_5 : \{ s_2, s_3, s_4 \}$
$f_6 = f_2 \rightarrow f_5$	$f_6 : \{ s_0, s_2, s_3, s_4 \}$
$f_7 = \Phi = A X f_6$	$f_7 : \{ s_1, s_3, s_4 \}$



На M не выполняется Φ

Какого объема системы можно верифицировать?

- Алгоритм Model checking работает с состояниями и подмножествами состояний || системы процессов
- Системы с каким числом состояний можно верифицировать?
- Явное представление состояний - списком:
 - Каждое состояние ~ 100 байт –
 - какие подформулы истинны в нем;
 - с какими состояниями связано;
 - из каких состояний переходы;
 - нужно хранить множество подмножеств.
- Пусть в компьютере – 1 терабайт (10^{12} байт).
- Можно работать \sim с 10^7 (миллионами) состояний.



10^7 состояний

Верификация реактивных систем:

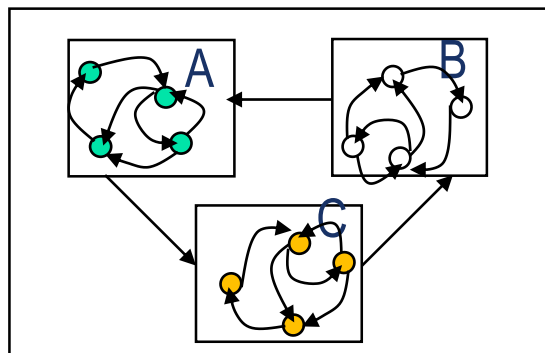
State explosion problem

$$2^{10} \sim 10^3$$

Классические алгоритмы верификации реактивных систем могут работать только с игрушечными системами – число состояний реальных систем очень велико

Обычная техника Model checking работает с системами $\sim 10^6$ - 10^7 состояний – представление каждого состояния требует ~ 50 - 100 байт

Пример:
три процесса,
три канала связи



Простая система, а
число состояний -
миллионы миллиардов!!

Каждая подсистема – 4 состояния + 1 целая переменная (short, 1 байт) + каналы для передачи целых (байт) значений.

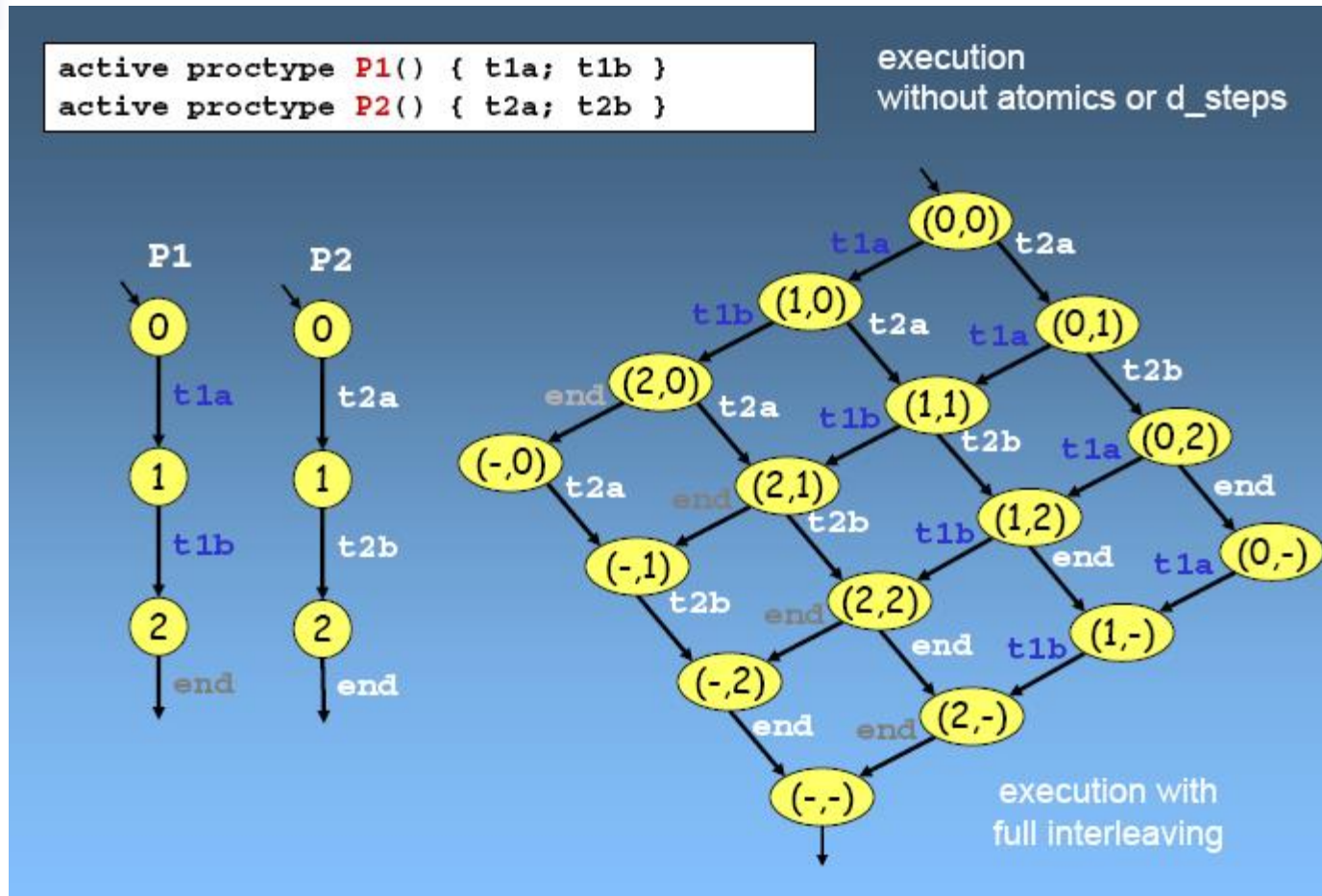
Пусть для простоты очередь каждом из 3-х каналов не больше 1 сообщения

Всего $(4 \times 2^8)^3 \times (2^8)^3 = 2^{54} \approx 10^{16}$ глобальных состояний системы - **State**

Explosion Problem. Система в своей жизни проходит ничтожную долю своих возможных состояний, но мы не знаем, какие именно!! Число частиц во

Вселенной $\sim 10^{78}$
Ю.Г.Карпов

State explosion problem: интерливинг



- Два параллельных процесса с тремя состояниями каждый: экспоненциальный рост числа глобальных состояний для анализа.



Example - ADGS-2100 Adaptive Display & Guidance System

ADVANCED COMPUTING SYSTEMS



883 Subsystems

9,772 Simulink Blocks

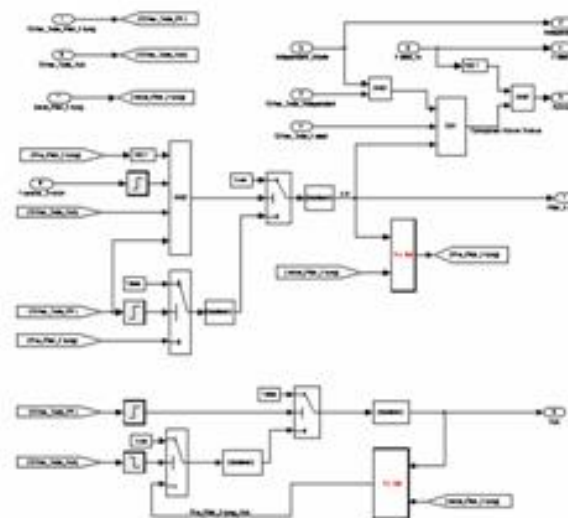
2.9×10^{52} Reachable States

Requirement

Drive the Maximum Number of Display Units
Given the Available Graphics Processors

Counterexample Found in 5 Seconds!

Checking 373 Properties
Found Over 60 Errors



**Rockwell
Collins**

- 
- Число состояний в реальных системах $\sim 10^{50}$ и более.
 - При использовании явного представления состояний нужна память в 10^{100} байт только для одного множества состояний.
 - Современные компьютеры – Терабайты (верхний предел).
 - Килобайт – 10^3 байт
 - Мегабайт – 10^6 байт
 - Гигабайт – 10^9 байт
 - Терабайт – 10^{12} байт
 - Петабайт – 10^{15} байт
 - Экзабайт – 10^{18} байт



Что делать?

Использовать специальные
эффективные методы представления
подмножеств конечных множеств и
операций над ними

Этот подход называется
“СИМВОЛЬНОЙ ВЕРИФИКАЦИЕЙ”



What Are Model Checkers?

ADVANCED COMPUTING SYSTEMS

- **Breakthrough Technology of the 1990's**
- **Widely Used in Hardware Verification (Intel, Motorola, IBM, ...)**
- **Several Different Types of Model Checkers**
 - Explicit, Symbolic, Bounded, Infinite Bounded, ...
- **Exhaustive Search of the Global State Space**
 - Consider All Combinations of Inputs and States
 - Equivalent to Exhaustive Testing of the Model
 - Produces a Counter Example if a Property is Not True
- **Easy to Use**
 - “Push Button” Formal Methods
 - Very Little Human Effort Unless You're at the Tool's Limits
- **Limitations**
 - State Space Explosion ($10^{100} - 10^{300}$ States)



Верификация методом Model checking

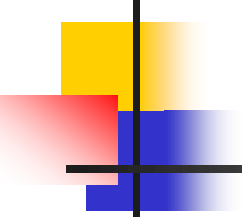
- Алгоритмы Model checking весьма эффективны: сложность проверки выполнимости формулы CTL Φ на структуре Крипке K пропорциональна числу подформул Φ и сложности K
- Какой сложности структуру Крипке можно разместить в памяти современного компьютера при явном представлении состояний?
- ~ 50 байт на состояние
- Каждое отношение – множество пар, ~ 100 байт на пару
- Число пар в отношении – порядка квадрата числа состояний
- Вся память пусть 1 терабайт = 10^{12} байт
- Вывод: явное представление состояний структуры Крипке позволяет работать с такими структурами, содержащими $\sim 10^6$ состояний

Структуры Крипке реальных систем содержат $10^{100} - 10^{200}$ состояний

Что делать?

Методы борьбы со “state explosion problem”:

1. Символьная верификация
2. Редукция частичных порядков
3. Композициональная верификация

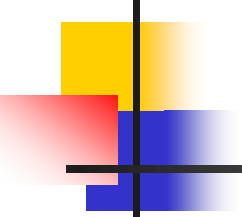


Использование бинарных решающих диаграмм (BDD) для представления множеств состояний позволило повысить число состояний систем, к которым можно применить алгоритмы верификации

от 10^7 до 10^{300}

**в триллионы триллионов триллионов триллионов
триллионов триллионов триллионов ... раз**

Возможность борьбы с **state explosion problem** при верификации реальных программных систем стало одним из самых удивительных применений BDD.

- 
- BDD совершили революцию в представлении конечных структур данных и алгоритмах работы с ними.
 - *“Бинарные решающие диаграммы (BDD) – удивительны, чем больше я играю с ними, тем больше я их люблю. Около 15 месяцев я был как ребенок с новой игрушкой, получив возможность решать такие проблемы, о которых я никогда не думал, что они могут быть решены”*

– Дональд Кнут



Binary Decision Diagrams

что это такое??

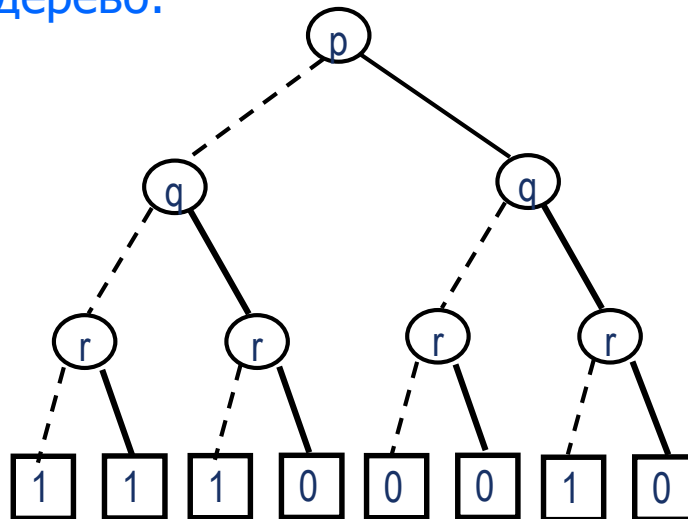
Представления булевых функций

Работать с БФ легко и удобно – если число переменных 3-4-5

1. Таблица истинности:

pqr	f
000	1
001	1
010	1
011	0
100	0
101	0
110	1
111	0

2. Семантическое дерево:



Каноническое представление

- Произвольная формула, ДНФ и КНФ не являются каноническими представлениями
- ТИ, семантическое дерево, СДНФ и СКНФ, полином Жегалкина являются каноническими представлениями

3. Логическая формула $f(p,q,r) = \neg p \vee q \oplus r q (p \vee r)$

4. СДНФ: $f(p,q,r) = \neg p \neg q r \vee \neg p \neg q \neg r \vee q p \neg r \vee q \neg p \neg r$

5. ДНФ: $f(p,q,r) = \neg p \neg q \vee q \neg r$

6. КНФ: $f(p,q,r) = (\neg p \vee q) (\neg q \vee \neg r)$

7. Полином Жегалкина: $f(p,q,r) = 1 \oplus p \oplus p q \oplus q r$

Но все они громоздки!!



Немасштабируемость решения задач с БФ

БФ, представленные стандартными способами, не позволяют эффективно решать проблемы вследствие экспоненциального роста представления БФ

Это - отражение главного положения теории сложности вычислений:

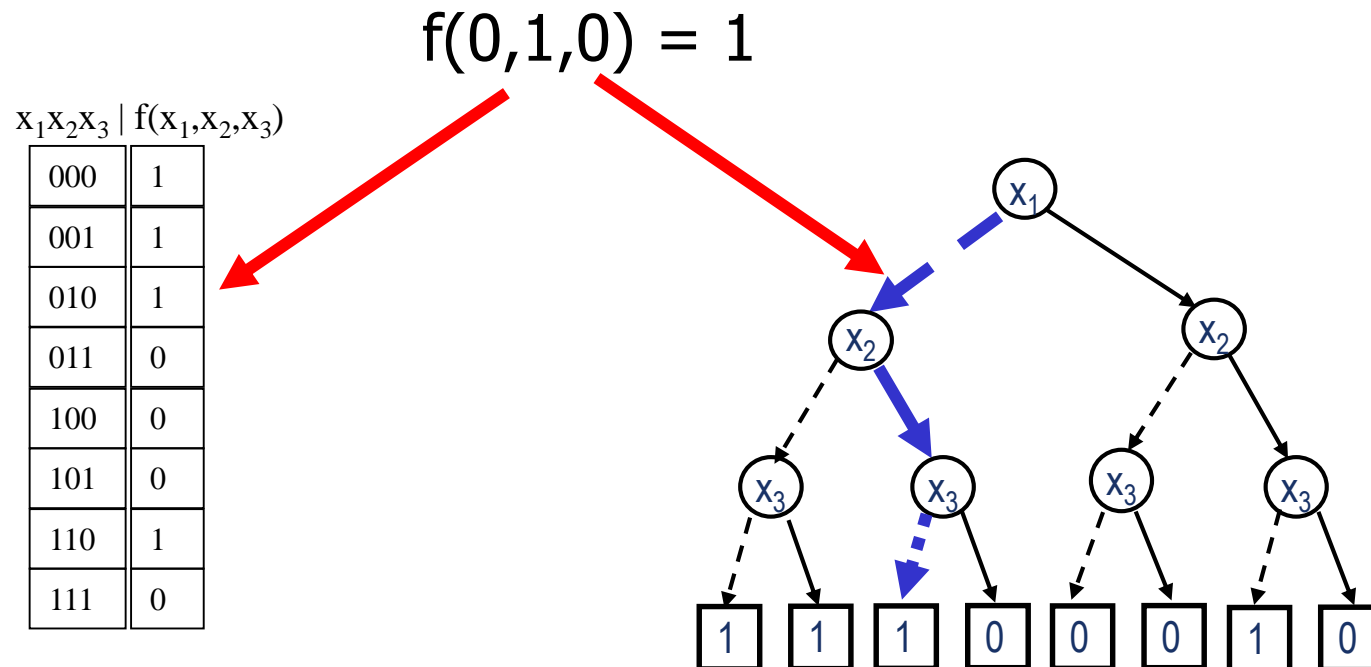
задача является практически неразрешимой, если время, требуемое для решения конкретных экземпляров этой задачи, растет экспоненциально с увеличением размеров этих экземпляров

Пример.

ТИ функции от 20 переменных имеет $2^{20} = 10^6$ строк, нужно оперировать мегабайтами информации

ТИ функции от 50 переменных имеет $2^{50} = 10^{15}$ строк, тысячи терабайт

Как вычислять значение двоичной функции по бинарному решающему дереву



Значение функции вычисляется простым движением по ветвям от корня к листу



BDD – новая форма представления булевых функций

В 1986 г. Randell Briant предложил новую очень эффективную форму представления БФ, которая называется Binary Decision Diagrams (BDD)

BDD – это представление функции в виде направленного графа - решающей диаграммы, в которой нет избыточностей

BDD имеют множество хороших свойств, с их помощью в настоящее время решаются многие задачи

Представление в BDD используемых на практике БФ растет полиномиально, или даже линейно

Наиболее экономно использование BDD там, где есть симметрия (например, аппаратные системы)

▪ R.E.Bryant. Graph-based algorithms for boolean functions manipulation. *IEEE Transactions on Computers*, 8 (C35), 1986

BDD – Бинарные решающие диаграммы (Binary Decision Diagrams)

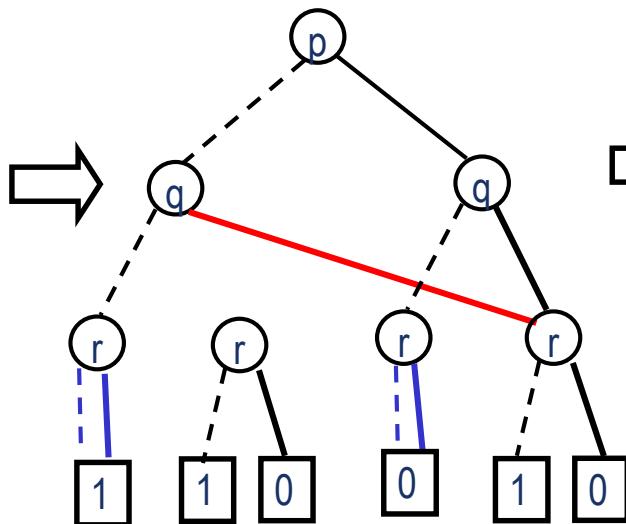
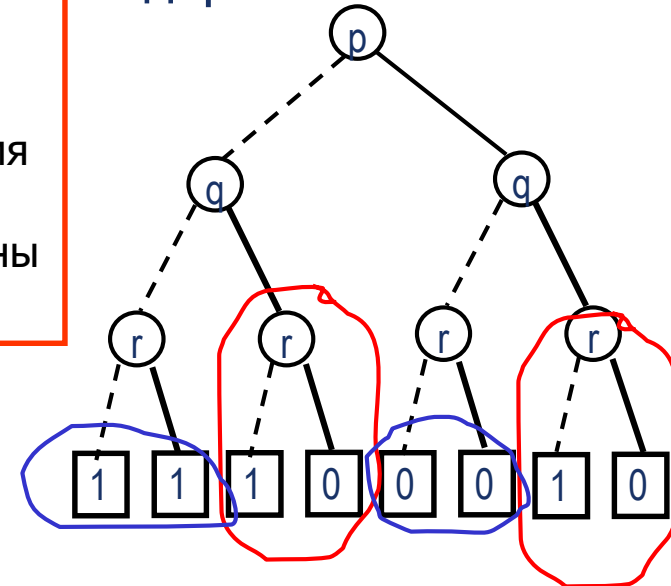
BDD-это семантическое дерево без избыточностей

BDD – ациклический орграф, в котором отсутствуют повторения в структуре, с одной корневой вершиной, двумя листьями, помеченными 0 и 1, и промежуточными вершинами. Корневые и промежуточные вершины помечены переменными, из них выходят ровно два ребра

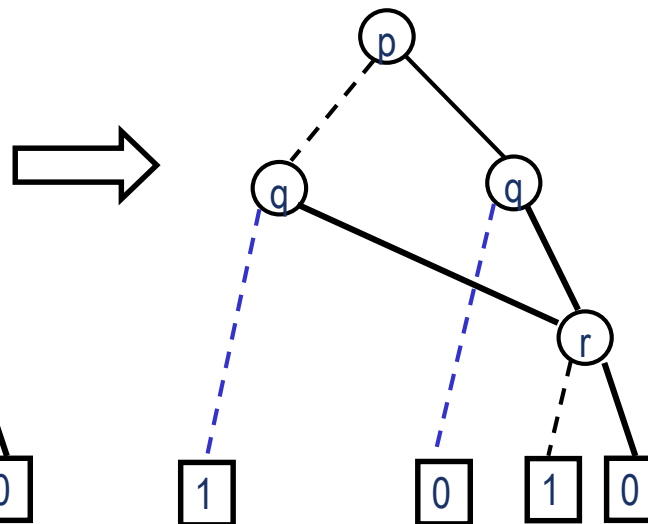
Binary – потому что двоичные переменные

Decision – потому что решения принимаются при направленном движении по графу (диаграмме)

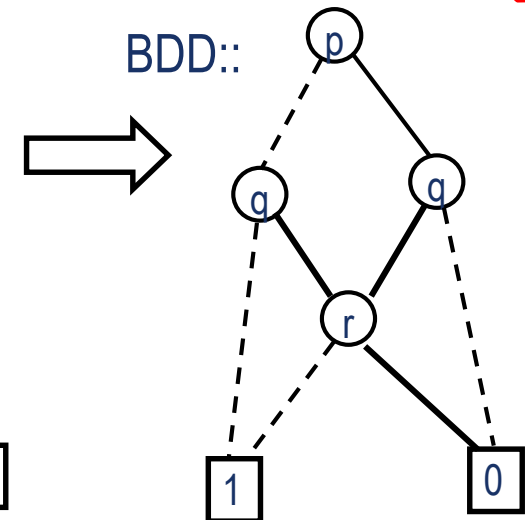
Семантическое
дерево:



Ю.Г.Карпов



Верификация. Model checking



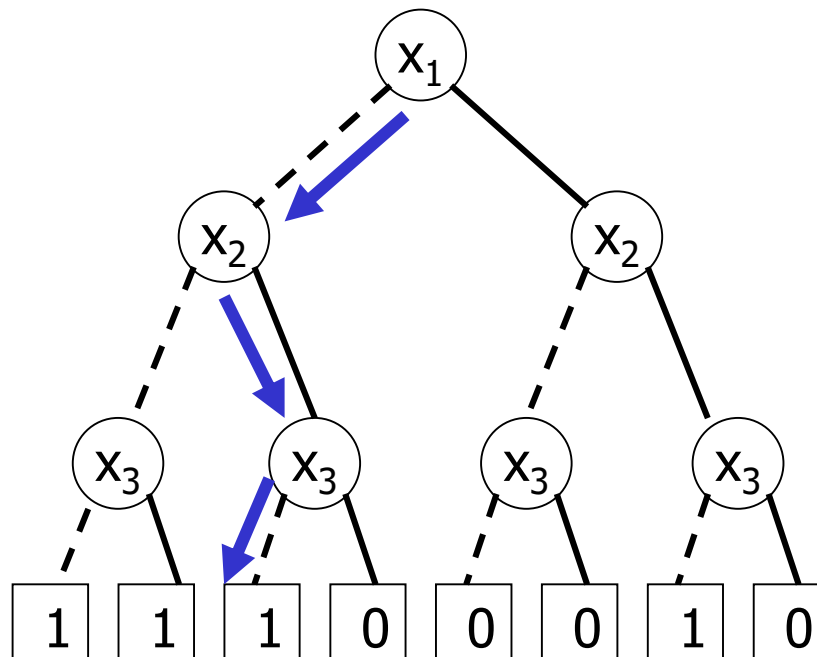
21

Как вычислять значение двоичной функции по бинарному решающему дереву

Вычисление f на наборе $\langle 0,1,0 \rangle$:
 $f(0,1,0) = 1$

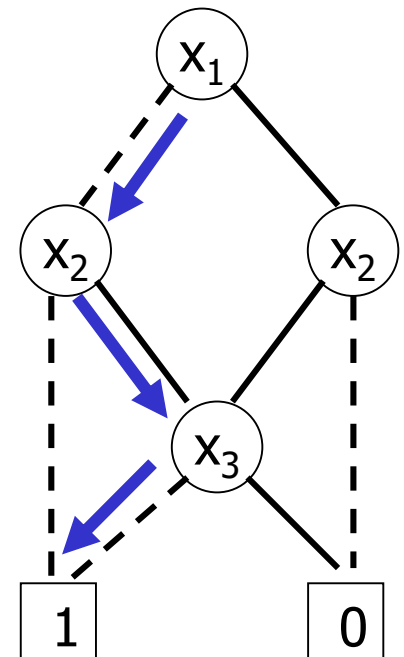
$x_1 x_2 x_3 \mid f(x_1, x_2, x_3)$

000	1
001	1
010	1
011	0
100	0
101	0
110	1
111	0



ТИ

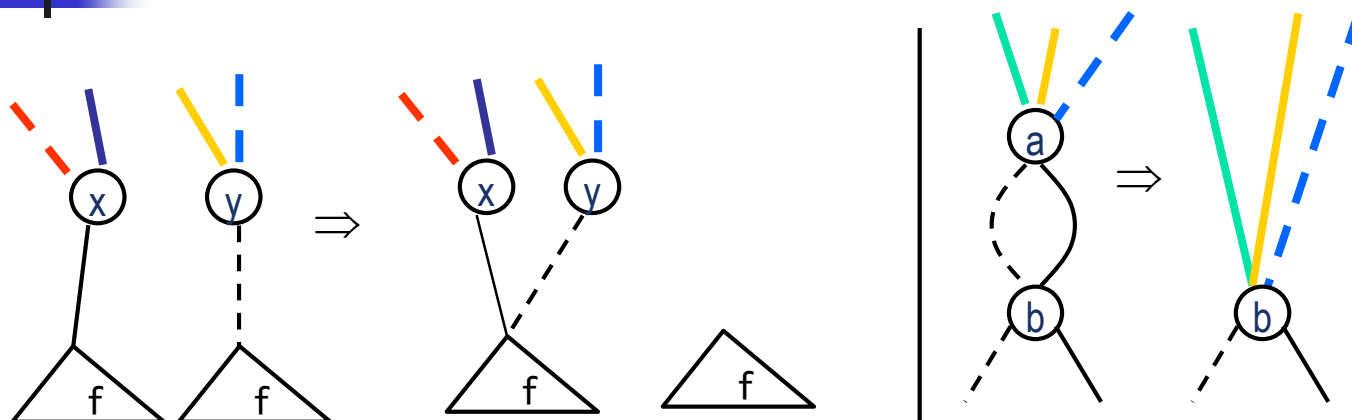
Binary Decision Tree



Binary Decision Diagram

Значение функции вычисляется простым движением по ветвям от корня к листу

Алгоритм Reduce: преобразование семантического дерева в BDD



C1: Если в графе имеются одинаковые подструктуры, то остается только одна из них

C2: Если обе выходные дуги вершины v ведут в одну вершину, то вершина v выбрасывается

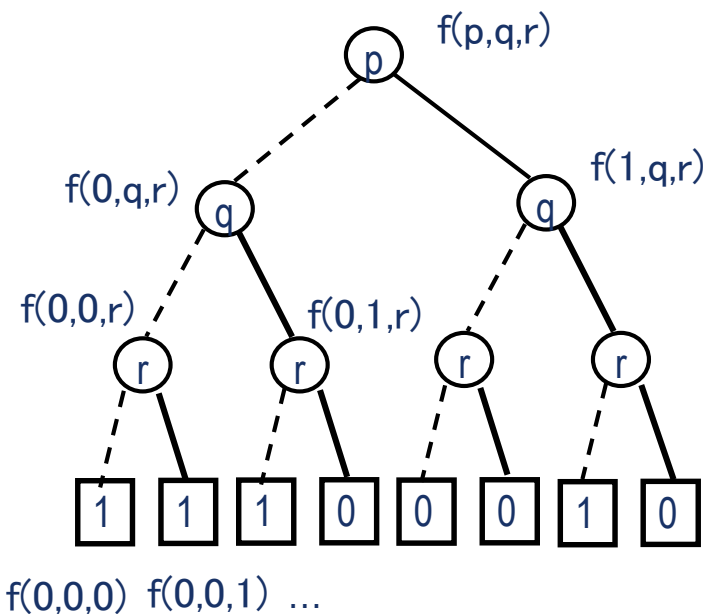
Повторное применение C1, C2 в любом порядке к семантическому дереву функции f или к любому полученному из него графу, приводит к минимальному представлению f , которое называется BDD

Что такое BDD (формально)

Разложение Шеннона:

$$f = \neg x f_{x=0} \vee x f_{x=1}$$

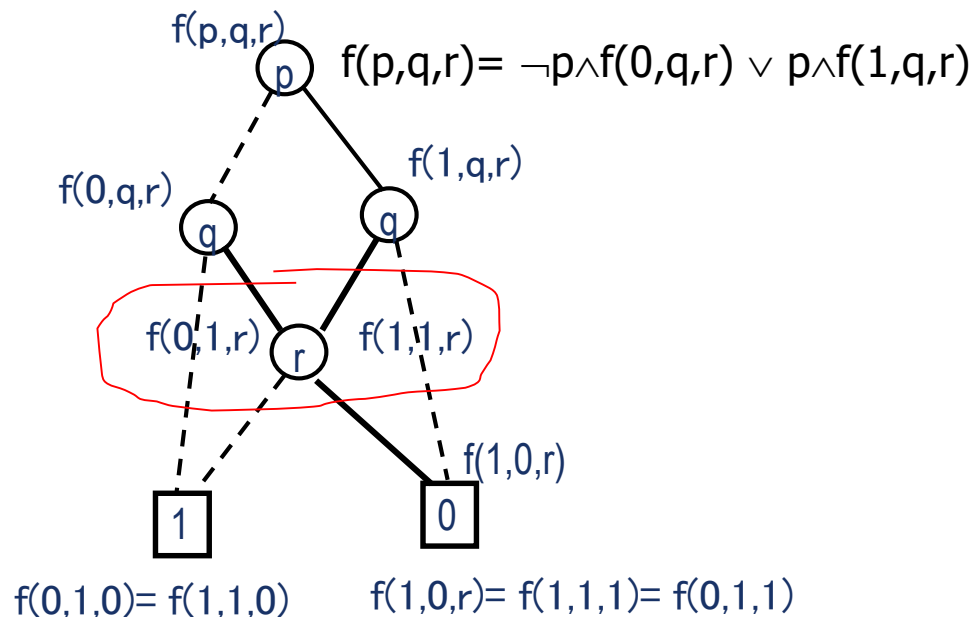
Семантическое дерево::



Булева функция $\text{ite}(p,q,r)$:

$$f(p,q,r) = \text{If } p \text{ then } f(1,q,r) \text{ else } f(0,q,r)$$

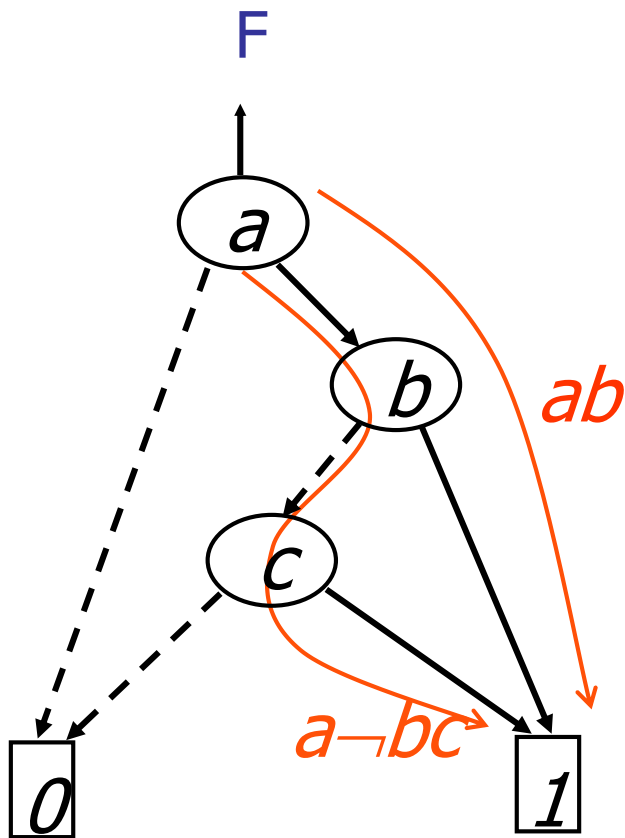
BDD::



BDD - это минимальное представление f в базисе $\{ \text{ite}(p,q,r), 0, 1 \}$

Вычисление значений функции по BDD

$$F(a,b,c) = ab \vee a\neg bc$$

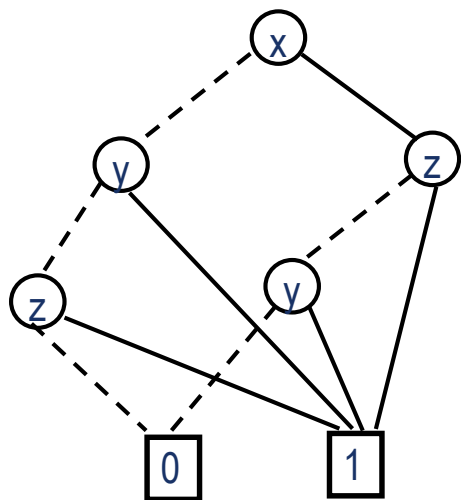


Вид функции в ДНФ определяют те пути, которые из корневой вершины идут в 1



BDD, OBDD и ROBDD

Не любая BDD является каноническим представлением, **нужно еще ограничение на порядок переменных**



BDD, но не OBDD

Ordered BDD - это BDD, в которой переменные не повторяются и встречаются в фиксированном порядке

Reduced Ordered BDD - это редуцированная OBDD. Именно ROBDD обладает всеми хорошими свойствами

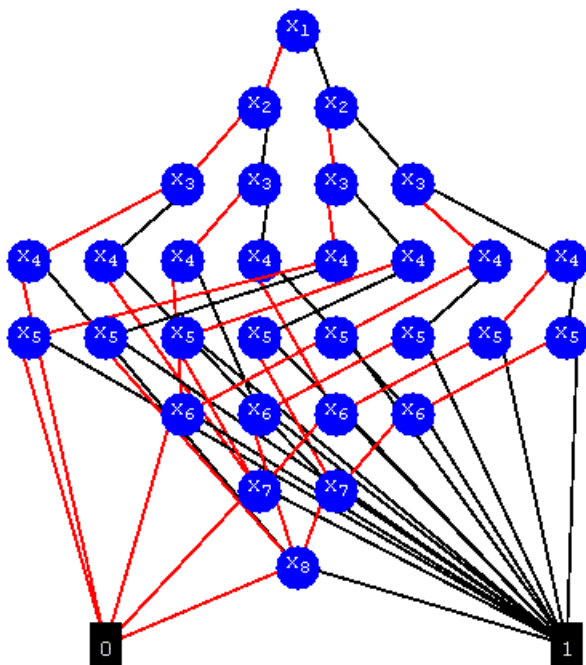
Чаще всего ROBDD называют просто BDD

R.E.Bryant. Symbolic function manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3), 1992

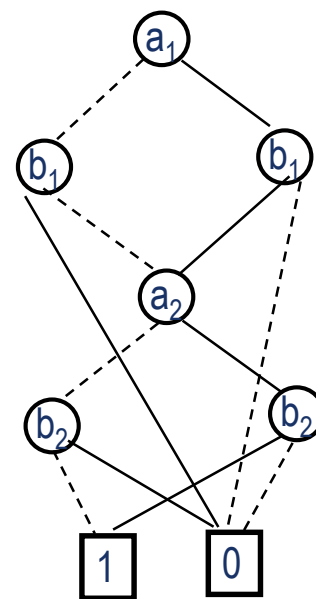


Примеры ROBDD

$$f = x_1 x_3 x_5 \neg x_7 \vee x_2 \neg x_4 x_5 \vee x_2 \neg x_3 \neg x_6 x_7 \vee x_5 x_6 x_7 \neg x_8$$



$$f = (a_1 \equiv b_1)(a_2 \equiv b_2)$$

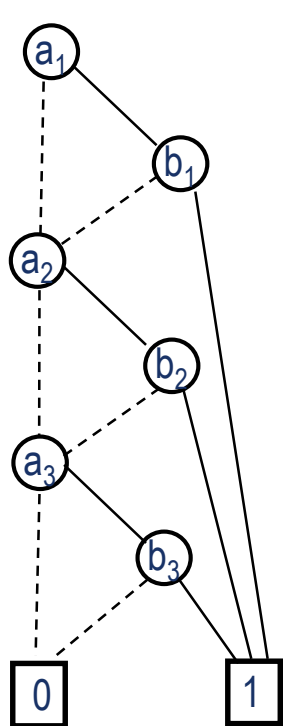


$$a1 < b1 < a2 < b2$$



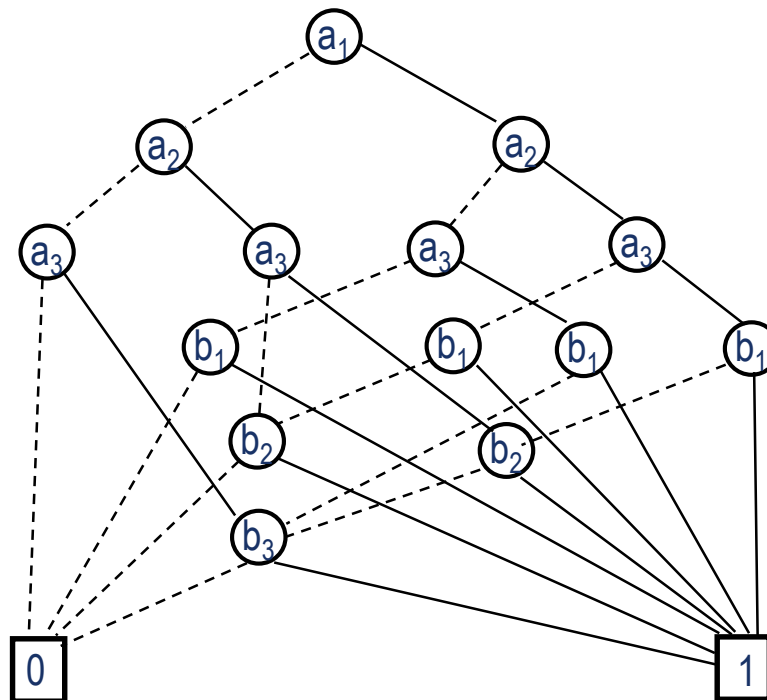
Сложность BDD зависит от порядка переменных

$$f = a_1 b_1 \vee a_2 b_2 \vee a_3 b_3$$



$a_1 < b_1 < a_2 < b_2 < a_3 < b_3$

Рост линейен



$a_1 < a_2 < a_3 < b_1 < b_2 < b_3$

Рост экспоненциален

Задача проверки оптимальности порядка является NP полной

Существуют эвристики для нахождения субоптимального порядка

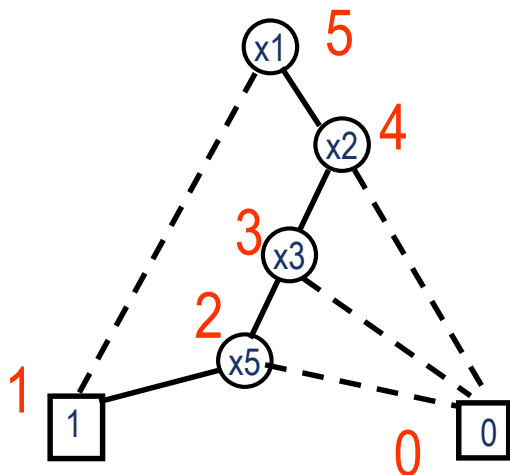
Какова сложность BDD при оптимальной упорядоченности?

Параллельный сумматор: при одном порядке переменных – линейная сложность, при другом – экспоненциальная

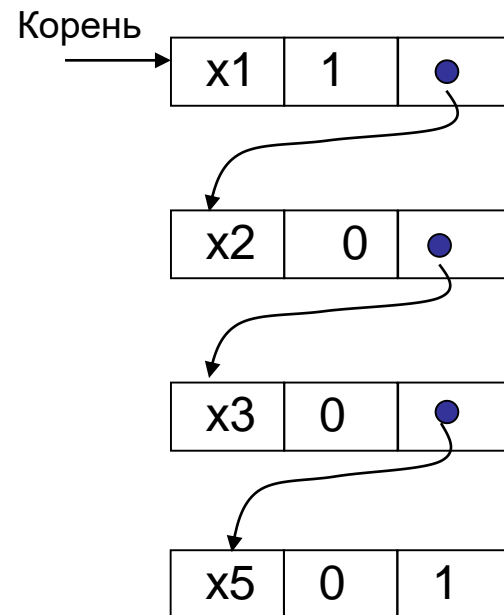
Для подавляющего числа встречающихся на практике функций сложность BDD линейна

Представление BDD

$$f = \neg x_1 \vee x_2 x_3 x_5$$



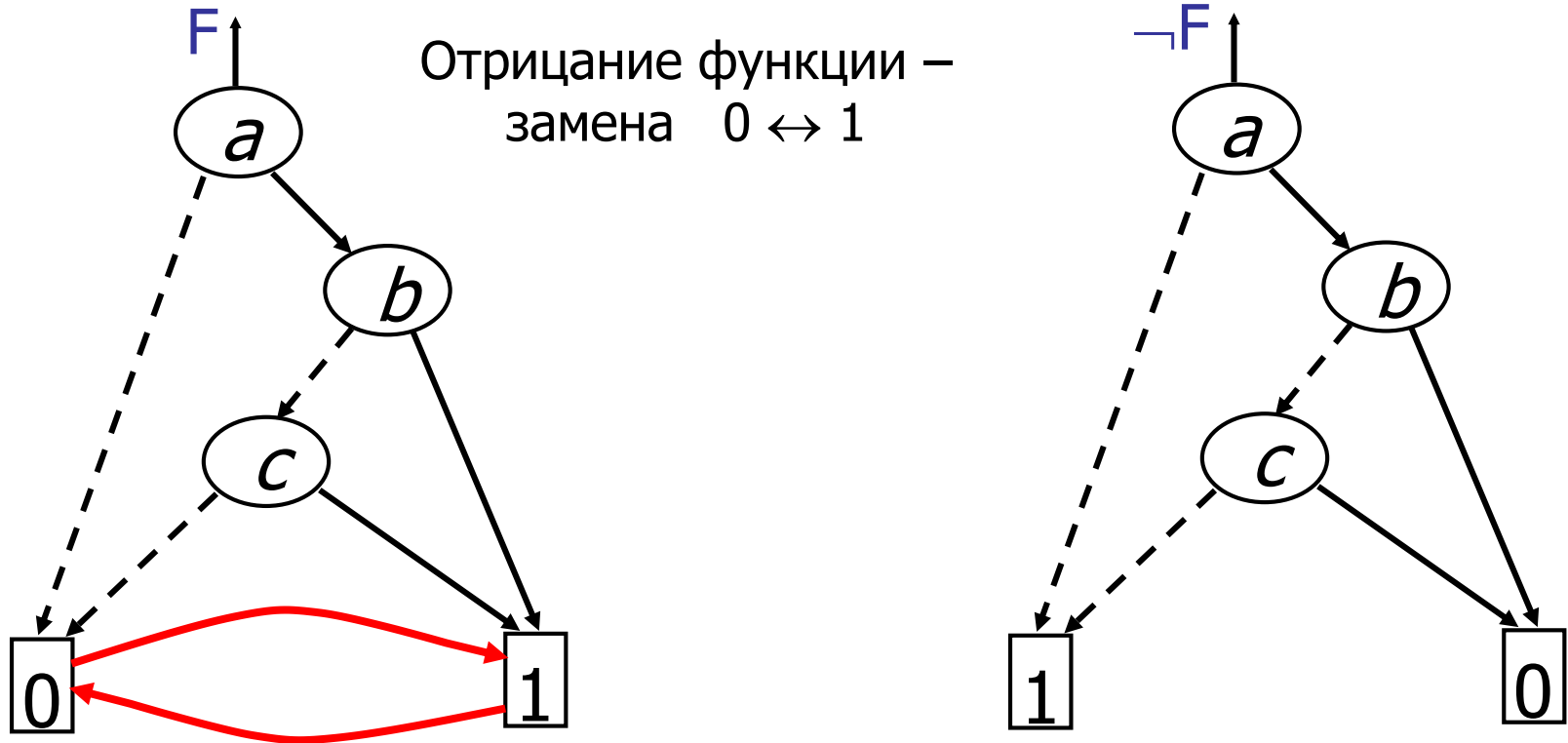
n	var	low	high
0			
1			
2	5	0	1
3	3	0	2
4	2	0	3
5	1	1	4



- BDD представляется таблицей, в которой указываются номер вершины, номер переменной, и куда направлены два выхода, 0 и 1
- Вместо таблицы можно использовать связный список
- Сложность представления БФ в BDD пропорциональна числу вершин



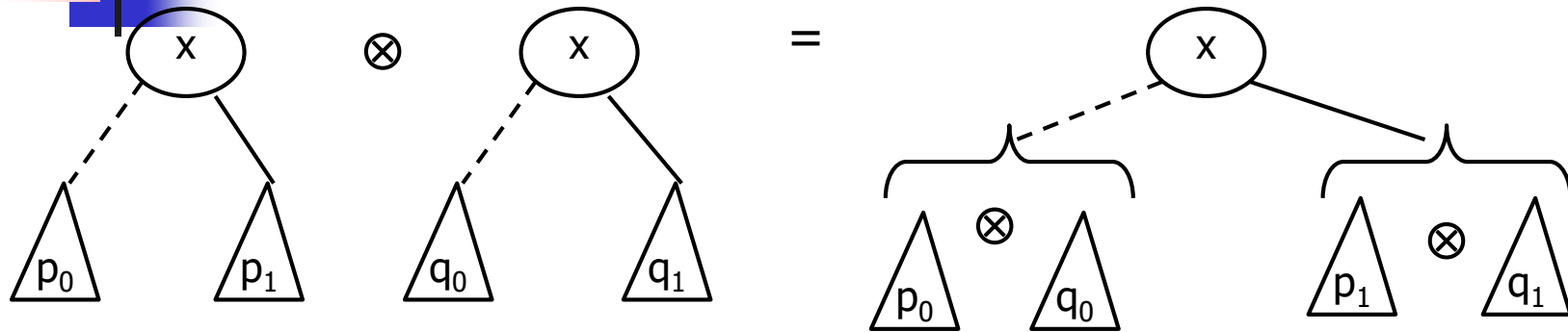
Булевы операции над BDD – операция Apply



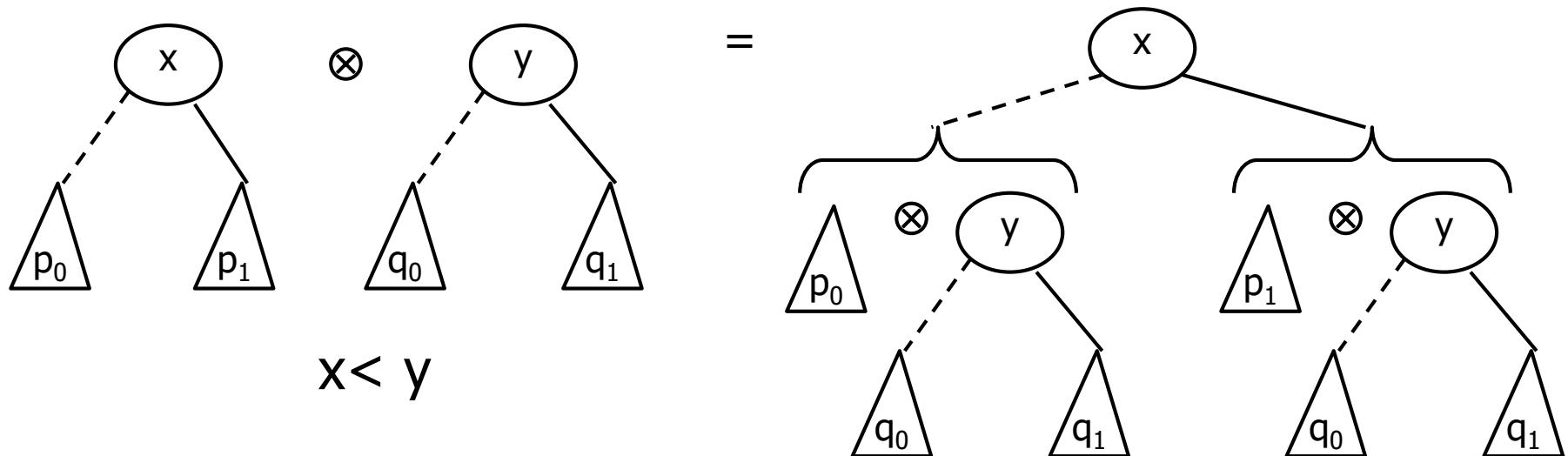
Бинарные операции – на основе разложения Шеннона:

$$F(x_1, \dots, x_n) \otimes \Phi(x_1, \dots, x_n) = x_i (F(x_1, \dots, 1, \dots, x_n) \otimes \Phi(x_1, \dots, 1, \dots, x_n)) \vee \neg x_i (F(x_1, \dots, 0, \dots, x_n) \otimes \Phi(x_1, \dots, 0, \dots, x_n))$$

Булевы операции над BDD – операция Apply



$$[x \rightarrow (p_0, p_1)] \otimes [x \rightarrow (q_0, q_1)] = [x \rightarrow (p_0 \otimes q_0, p_1 \otimes q_1)]$$



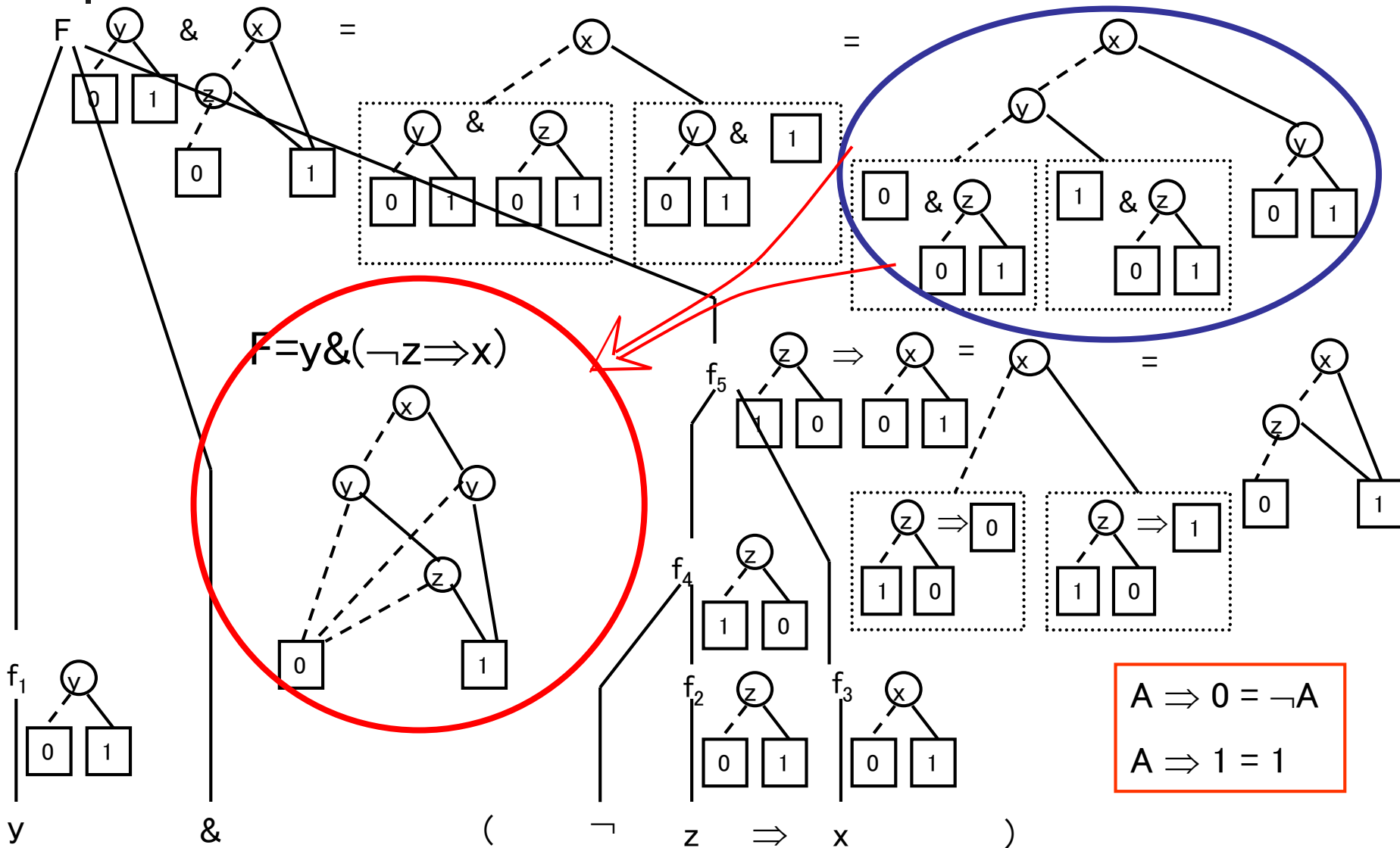
$$[x \rightarrow (p_0, p_1)] \otimes [y \rightarrow (q_0, q_1)] = [x \rightarrow (p_0 \otimes [y \rightarrow (q_0, q_1)], p_1 \otimes [y \rightarrow (q_0, q_1)])]$$

Реализация булевых операций над BDD имеет линейную сложность

Как построить BDD по булевой функции

$$F = y \& (\neg z \Rightarrow x)$$

$$x < y < z$$





Свойства BDD

1. BDD – **каноническое** представление – **любая** БФ имеет *единственное BDD-представление*

2. Сложность зависит от порядка переменных.

Пример: функция $(a_1 \oplus b_1) \& \dots \& (a_n \oplus b_n)$

при порядке $a_1 \dots a_n b_1 \dots b_n$ имеет сложность $3 \times 2^{n-1}$

при порядке $a_1 b_1 a_2 b_2 \dots a_n b_n$ имеет сложность $3 \times n + 2$

3. Проблема нахождения оптимального порядка NP – трудна. Но есть эвристики

4. BDD для большинства функций имеет линейную сложность.

Некоторые классы функций имеют экспоненциальную сложность BDD *при любом* порядке переменных (пример: функция, выдающая средний бит результата произведения $A \times B$ n -разрядных переменных A и B)

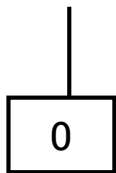
5. Булевы операции над БФ, представленными в BDD, просты.

Теорема. Сложность выполнения булевых операций над двумя функциями f и g , представленными в BDD, полиномиальна: $O(|f| \times |g|)$

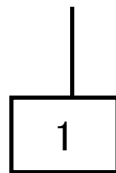
Свойства BDD (2)

- Выполнимость (Теорема Кука: “Проблема выполнимости булевой формулы *NP-полна*”)
 - Для BDD проверка выполнимости, невыполнимости и общезначимости **тривиальны** – не является ли BDD вырожденной (значения 0 или 1)
 - Но построение представления функции в BDD в худшем случае занимает экспоненциальное время

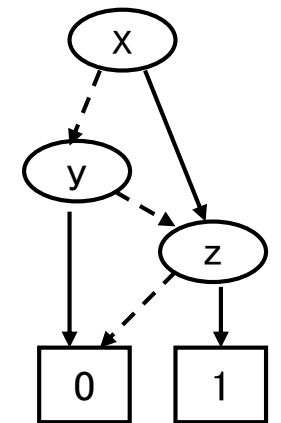
В ы р о ж д е н н ы е BDD:



Н е в ы п о л
н и м а



О б щ е з н а
ч и м а



В ы п о л н
и м а

- Совершенно неожиданно:

Для того, чтобы эффективно
работать с двоичными
функциями, нужно оперировать
не формулами, а графами.

ABCD: The ABCD package by Armin Biere

<http://fmv.jku.at/abcd/>.

BuDDy: A BDD Package by Jørn Lind-Nielsen.

<http://sourceforge.net/projects/buddy/>

CMU BDD, BDD package, Carnegie Mellon University, Pittsburgh

<http://www.cs.cmu.edu/~modelcheck/bdd.html>

CUDD: BDD package, University of Colorado, Boulder

<http://vlsi.colorado.edu/~fabio/CUDD/>

JavaBDD, a Java port of BuDDy that also interfaces to CUDD, CAL, and JDD

<http://javabdd.sourceforge.net/>

The Berkeley CAL package which does breadth-first manipulation

http://embedded.eecs.berkeley.edu/Research/cal_bdd/

TUD BDD: A BDD Package and a World-Level package by Stefan Höreth

<http://www.rs.e-technik.tu-darmstadt.de/~sth/>

Vahidi's JDD , a java library that supports common BDD and ZBDD operations

<http://javaddlib.sourceforge.net/jdd/>



Применения BDD

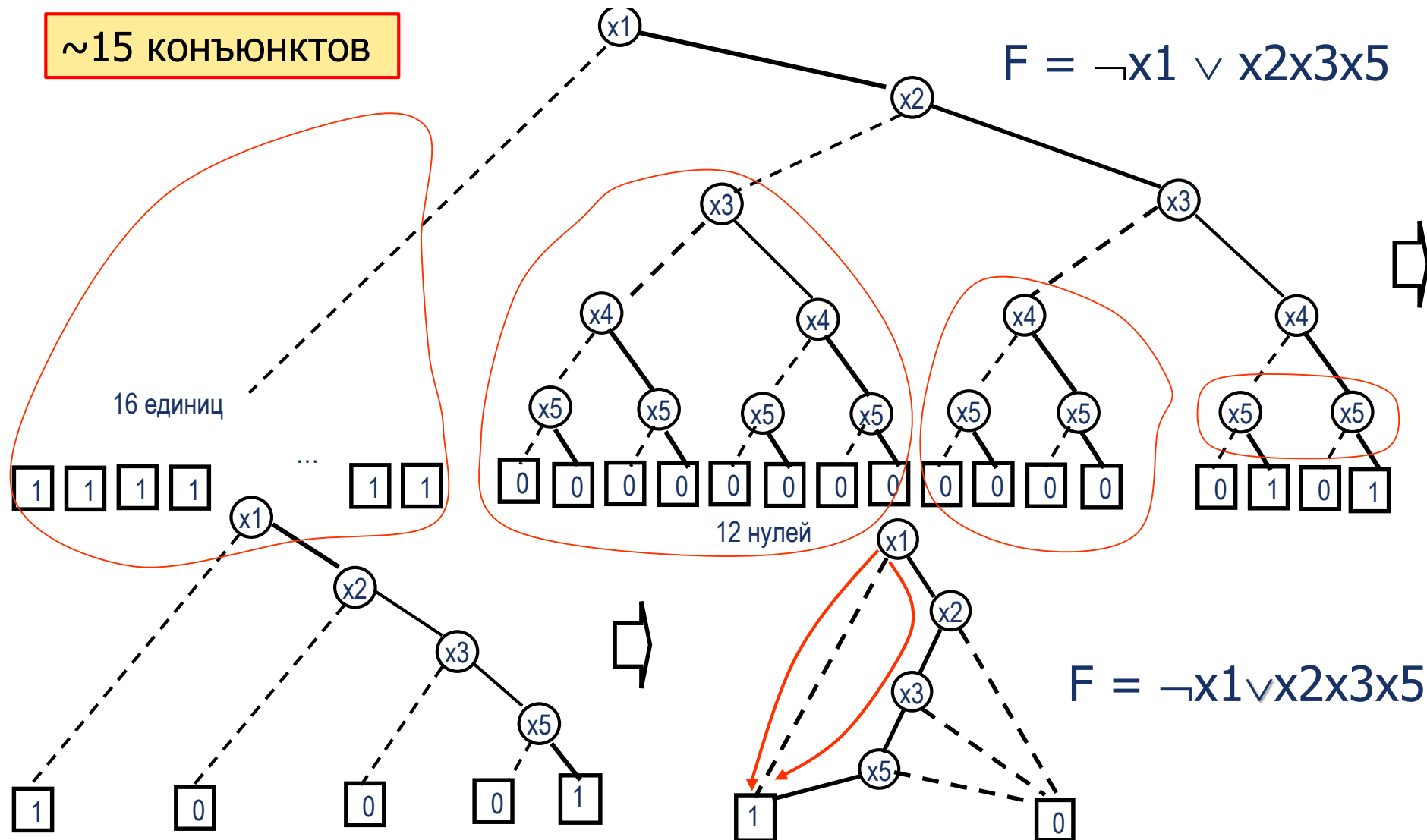
- Применений оказалось огромное количество
- “Прямые” применения
 - Минимизация логических функций
 - Операции над БФ от большого числа переменных
 - Синтез логических схем по их представлению в форме BDD
 - Верификация и проверка эквивалентности логических схем
- Борьба с “проклятием размерности”
 - Упрощение любых алгоритмов, манипулирующих конечными структурами данных большого объема
 - Компрессия изображений, Поиск в БД, Алгоритмы на графах
 - ...
 - Верификация реактивных (reactive) систем
 - С использованием BDD стало возможным увеличить сложность верифицируемых систем на многие порядки!
- Расширения BDD

Как сократить представление логической функции

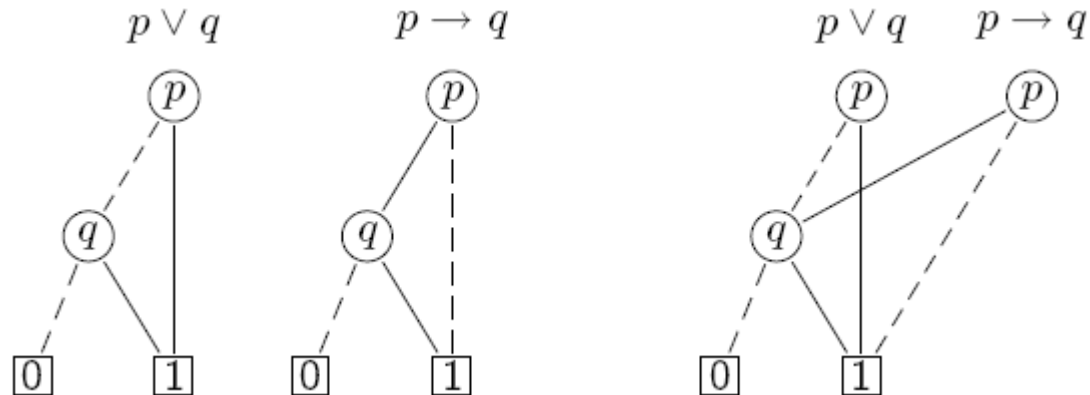
$$F = \neg x_1 \neg x_2 \neg x_3 \neg x_4 \neg x_5 \vee \neg x_1 \neg x_2 x_3 \neg x_4 \neg x_5 \vee \dots \vee \neg x_1 \neg x_2 x_1 x_2 x_3 \neg x_4 x_5 \vee x_1 x_2 x_3 x_4 x_5$$

~15 конъюнктов

$$F = \neg x_1 \vee x_2 x_3 x_5$$



Многокорневые BDD – совместное представление БФ



- Представление двух функций с помощью BDD с двумя корневыми вершинами – фактически, совместная минимизация нескольких функций

Библиотеки обычно содержат операции над многокорневыми BDD

В 2010 в С.Петербургском университете разработана еще одна подобная библиотека

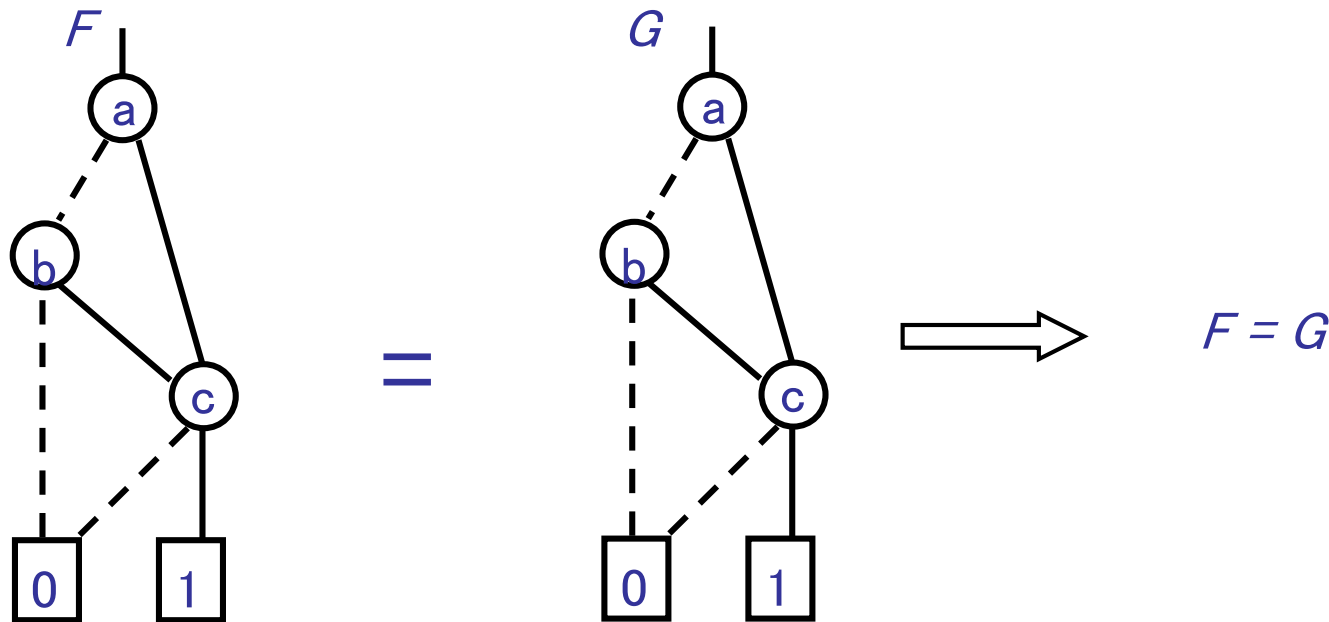
Д.Ю.Бугайченко, И.П.Соловьев. Библиотека многокорневых бинарных решающих диаграмм BddFunctions и её применение // DmitryBugaychenko@math.spbu.ru

Проверка эквивалентности булевых функций

$$F = a\bar{b}c + abc + ab\bar{c}$$

$$G = ac + bc$$

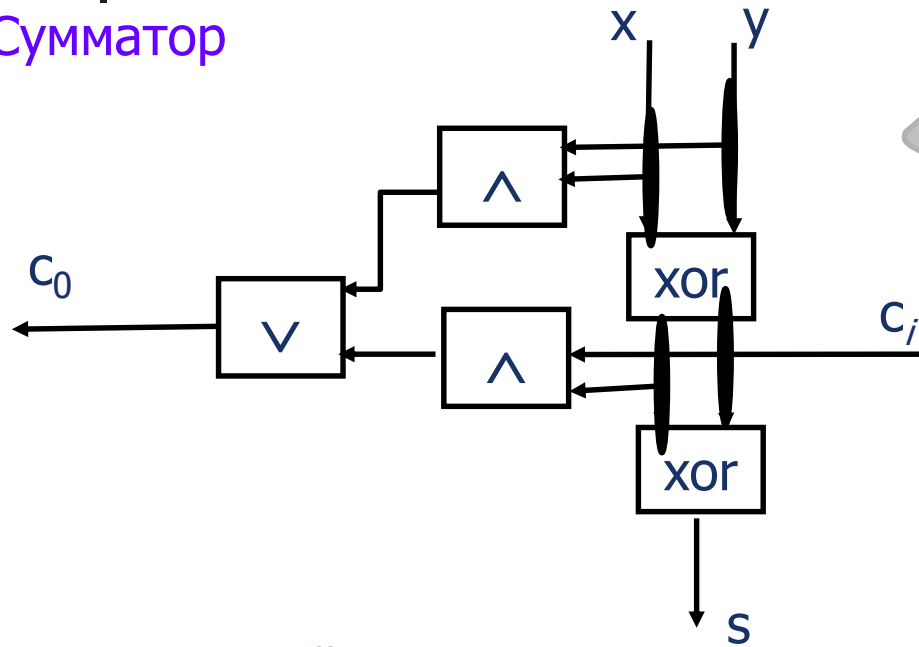
$$F = G (?)$$



Поскольку BDD – каноническое представление, проверка эквивалентности двух функций, представленных в BDD, сводится к проверке совпадения направленных графов

Применение BDD: удовлетворяет ли логическая схема ее спецификации?

Сумматор

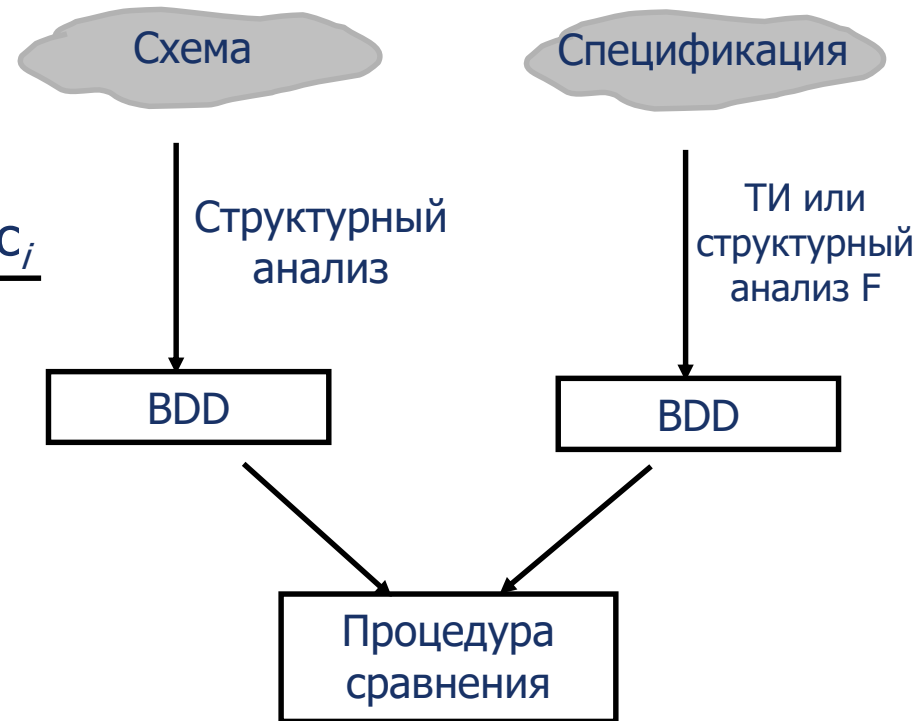


Структурный анализ схемы:

$$s = (x \oplus y) \oplus c_i$$

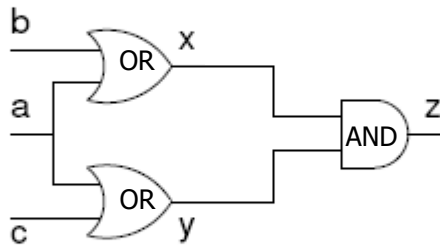
$$c_0 = x \wedge y \vee (x \oplus y) \wedge c_i$$

Спецификация: ТИ или формула



Аналогично проверяется эквивалентность двух схем

Как построить BDD по структуре схемы

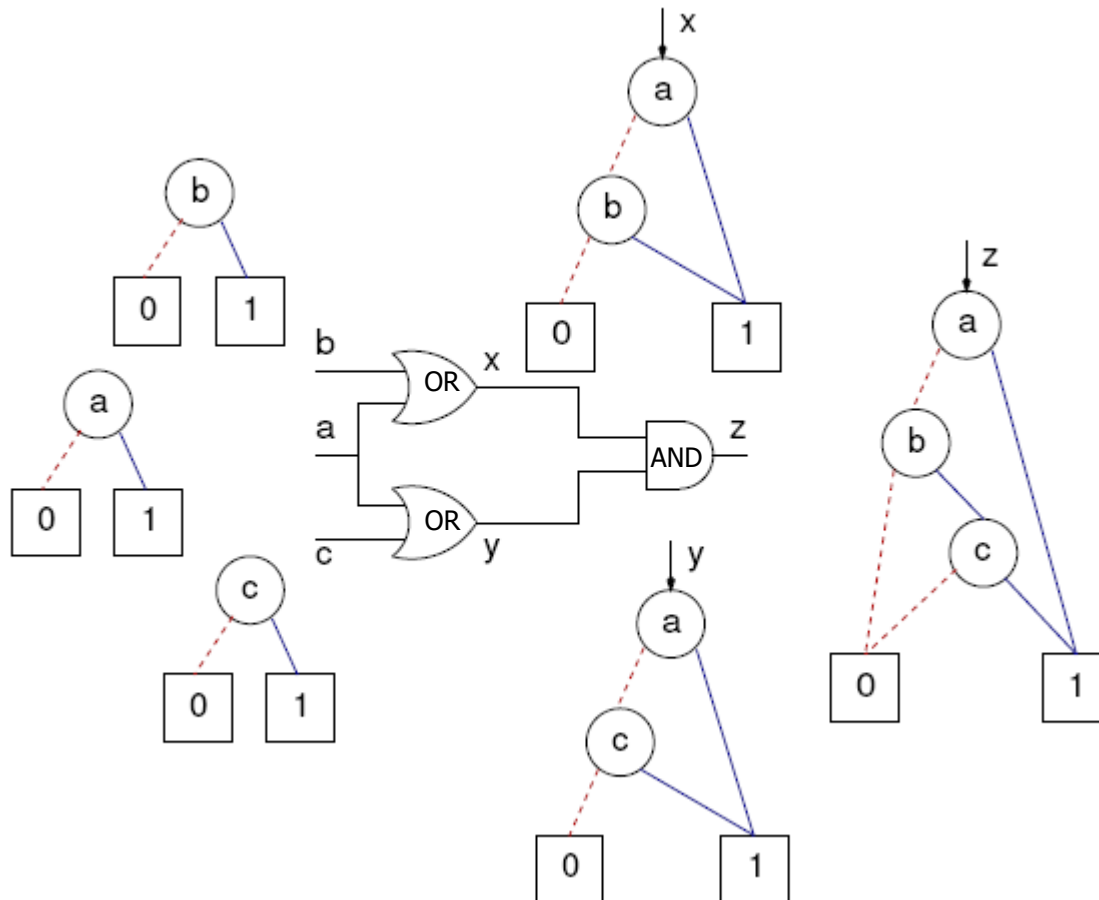


1. Сначала строятся BDD исходных переменных

2. Затем по ярусам строятся BDD всех промежуточных подфункций

3. Результирующая BDD представляет ЛФ, реализуемую схемой

Как построить BDD по структуре схемы

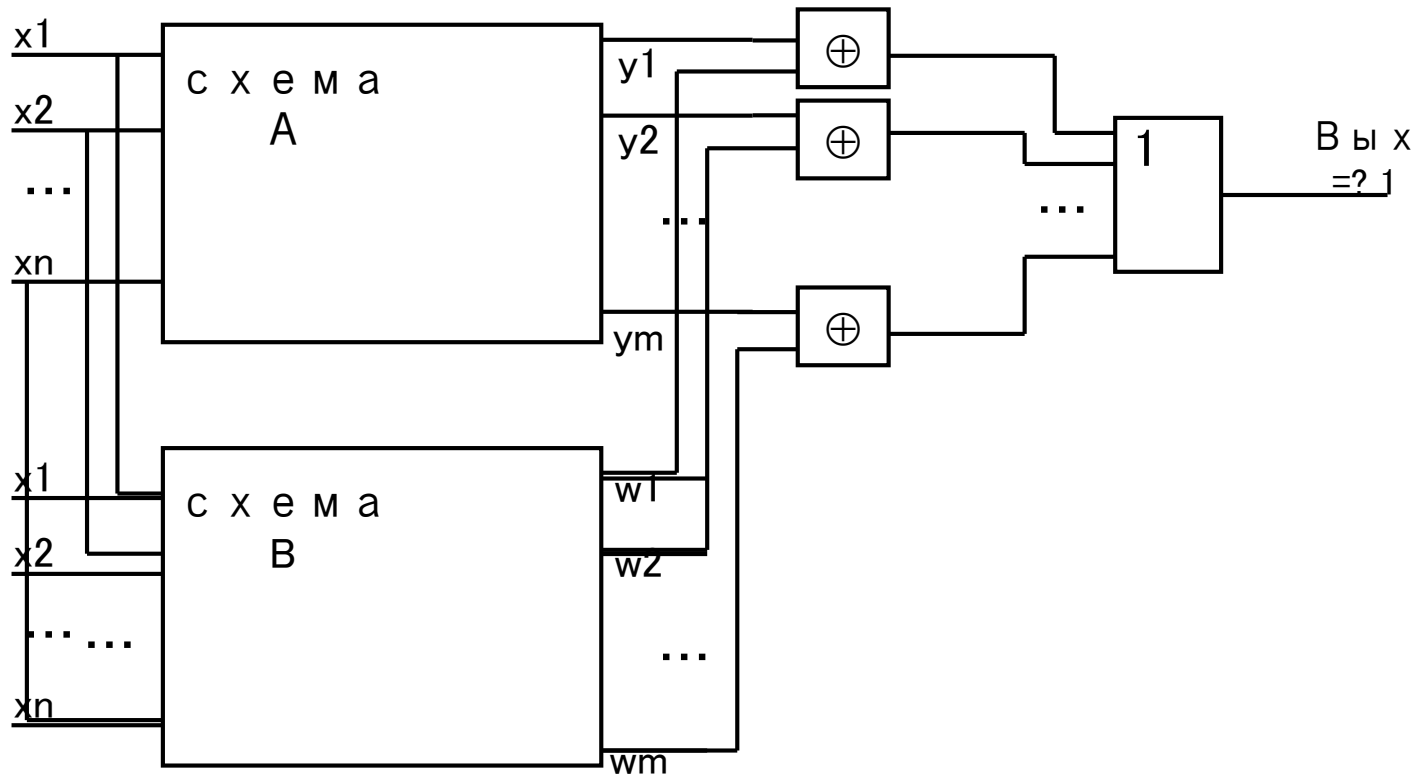


1. Сначала строятся BDD исходных переменных

2. Затем по ярусам строятся BDD всех промежуточных подфункций

3. Результирующая BDD представляет ЛФ, реализуемую схемой

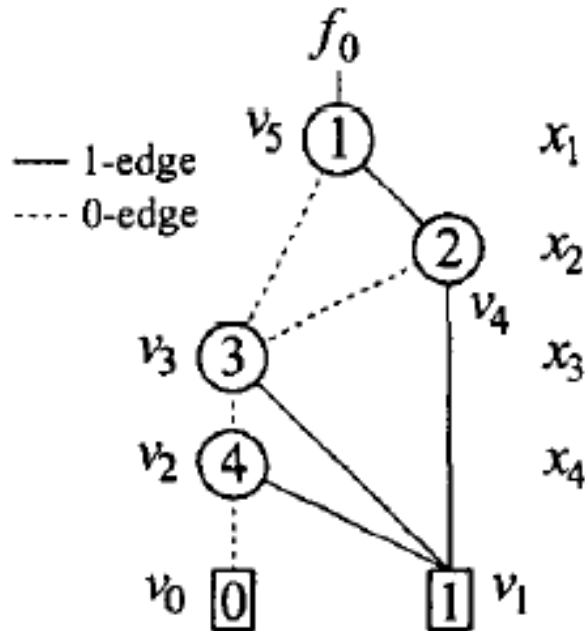
Эквивалентность двух схем – общий случай



- Строим БФ F для композиции и проверяем, существуют ли интерпретации, на которых F равна 1? Но это проблема SAT!
- С BDD эта проблема ОБЫЧНО решается просто!

Автоматическая генерация программы для вычисления значений функции по BDD

$$F = \neg x_1 \neg x_2 \neg x_3 \neg x_4 \neg x_5 \vee \neg x_1 \neg x_2 x_3 \neg x_4 \neg x_5 \vee \dots \vee \neg x_1 \neg x_2 \vee x_1 x_2 x_3 \neg x_4 x_5 \vee x_1 x_2 x_3 x_4 x_5$$



(a) BDD.

v_5 : if($x_1 == 0$) goto v_3 ;

else goto v_4 ;

v_4 : if($x_2 == 0$) goto v_3 ;

else goto v_1 ;

v_3 : if($x_3 == 0$) goto v_2 ;

else goto v_1 ;

v_2 : if($x_4 == 0$) goto v_0 ;

else goto v_1 ;

v_1 : return(1);

v_0 : return(0);

(b) Branching program
generated from BDD.

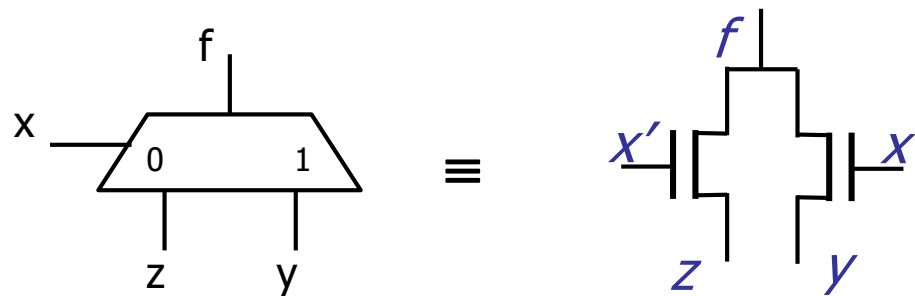
Вычисление значений БФ по BDD – просто движение по упорядоченному графу.

Ответ всегда не более, чем за N шагов

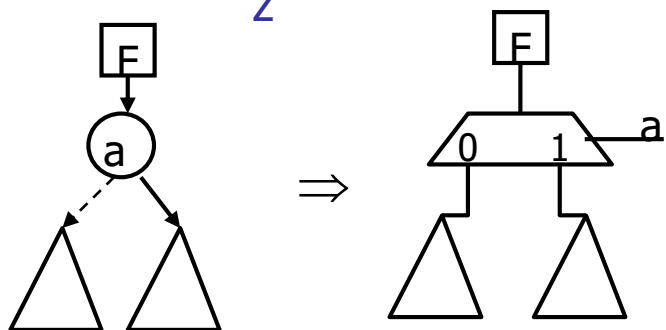
Реализация схемы по BDD

Используется мультиплексор (MUX) – схема, с тремя входами и одним выходом

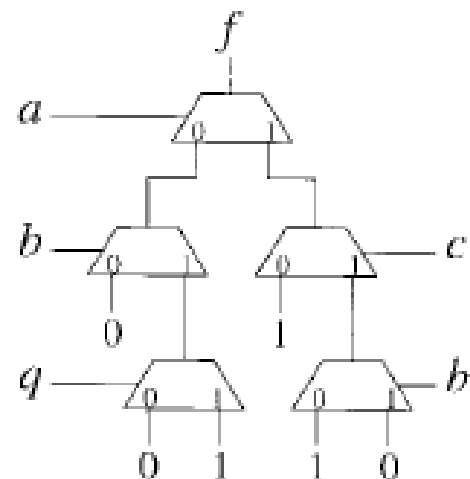
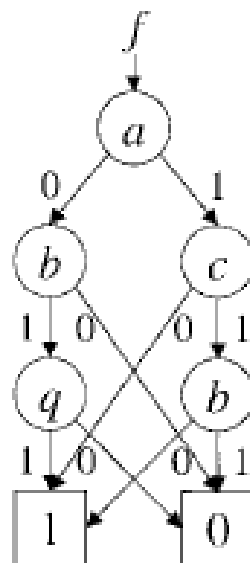
Существует технология PTL (Pass Transistor Logic) построения ЛС на основе MUX



$\text{ite}(x,y,z) = \text{if } x \text{ then } y \text{ else } z$ $f = x'z \vee xy$

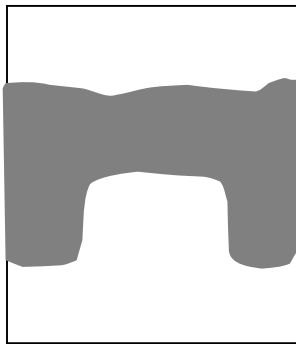


PTL синтез, основанный на "прямом" BDD mapping

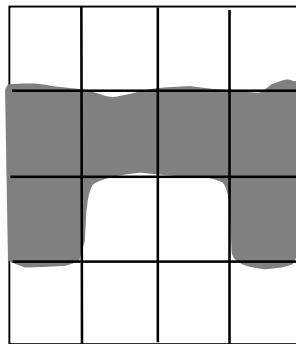


Интенсивно развивается направление BDD-Based Synthesis с PTL и комбинации PTL с CMOS логикой

Компрессия изображений



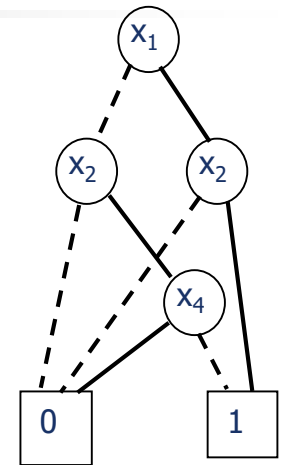
а)



б)

		X_3X_4			
		10	11	01	00
X_1X_2	10	0	0	0	0
	11	1	1	1	1
	01	1	0	0	1
	00	0	0	0	0

в)



г)

$$f = x_1x_2 \vee x_2\neg x_4$$

а) двумерное черно-белое изображение

б) разбиение изображения на множество пикселей

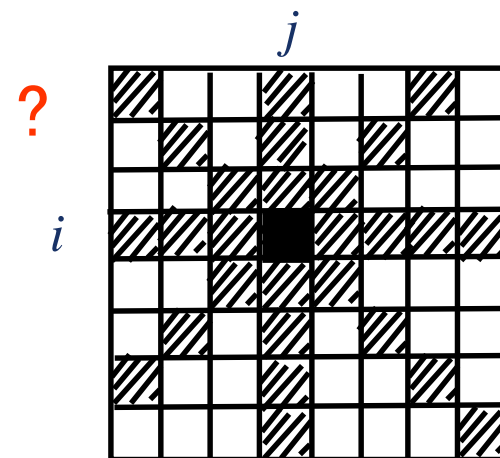
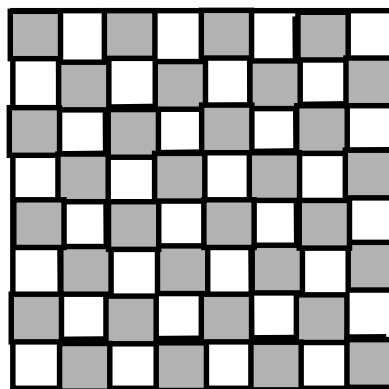
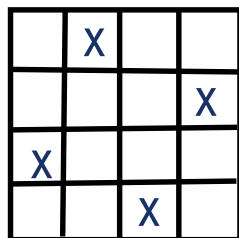
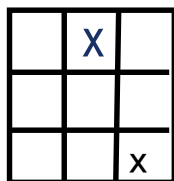
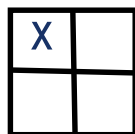
в) соответствующая двоичная функция $x_1x_2 \vee x_2\neg x_4$

г) BDD, представляющая изображение

Для представления мегабайтного изображения нужна BDD функции от 23 переменных (поскольку $\lceil \log(8 \cdot 10^6) \rceil = 23$)

Логические задачи, решаемые с помощью БФ

Задача о расстановке ферзей



Если ферзь в клетке (i, j) , ТО:
($x_{ij} \Rightarrow$):

на i -й горизонтали нет ферзей:

И на j -й вертикали нет ферзей:

И на с-з диагонали нет ферзей:

И на с-в диагонали нет ферзей:

И на каждой горизонтали есть ферзь: $\bigwedge_{1 \leq k \leq N} x_{i1} \vee x_{i2} \vee x_{i3} \dots \vee x_{iN}$

$$\bigwedge_{1 \leq k \leq N, k \neq j} \neg x_{ik}$$

$$\bigwedge_{1 \leq k \leq N, k \neq i} \neg x_{kj}$$

$$\bigwedge_{1 \leq k \leq N, 1 \leq j+k-i \leq N, k \neq i} \neg x_{k, j+k-i}$$

$$\bigwedge_{1 \leq k \leq N, 1 \leq j-i+k \leq N, k \neq i} \neg x_{k, j-i+k}$$

Конъюнкция всех этих формул даст такую БФ F , что любая интерпретация, на которой $F=1$, будет решением

Для решения задачи нужна BDD для ф-ции от 64 переменных



Программирование в ограничениях

Constrained programming (Constraint Satisfaction Problem, CSP)

Основано на описании модели задачи, а не алгоритма ее решения

Модель специфицируется в виде отношений – ограничений, которые отражают связи, существующие между параметрами задачи

Постановка задачи

На переменные Z_1, \dots, Z_n ($Z_i \in D_i$) наложены ограничения $C_k(Z_1, \dots, Z_n)$ - уравнения, неравенства, логические выражения и т.п

Найти наборы значений $\langle a_1, \dots, a_n \rangle$ ($a_i \in D_i$), удовлетворяющие ограничениям

Использование BDD:

Значения в каждой области D_i кодируются двоичными наборами

Каждое ограничение C_k преобразуется в логическую функцию f_k , которая представляется в форме BDD

Множество ограничений представляется конъюнкцией этих функций $F = \&_i f_i$

Решение определяется такими наборами кодировок, на которых $F = 1$

Пример: логическая головоломка

Задача (С.Расел, П.Норвиг: *Искусственный интеллект. Современный подход*)

В пяти домах разного цвета живут лица разных национальностей, которые пьют разные напитки, держат разных животных, курят разные сигареты

D^1 : Цвет = {красный, синий, желтый, зеленый, белый}	$= \{ 000, \dots, 100 \} (x_i^1, y_i^1, z_i^1)$
D^2 : Нац = {англ, испанец, японец, норвежец, украинец}	$= \{ 000, \dots, 100 \} (x_i^2, y_i^2, z_i^2)$
D^3 : Напиток = {сок, вино, чай, молоко, вода}	$= \{ 000, \dots, 100 \} (x_i^3, y_i^3, z_i^3)$
D^4 : Животное = {улитка, лиса, кошка, лошадь, собака}	$= \{ 000, \dots, 100 \} (x_i^4, y_i^4, z_i^4)$
D^5 : Сигареты = {"Парламент", "Кулс"..., "Честерфилд"}	$= \{ 000, \dots, 100 \} (x_i^5, y_i^5, z_i^5)$

Ограничения

C_1 : Англичанин живет в желтом доме
 C_2 : Человек, держащий лису, пьет сок
 C_3 : ...

C_1 : Англ \Rightarrow Желтый

C_2 : Лиса \Rightarrow Сок

$f_1 = \&_i (\neg x_i^2 \neg y_i^2 \neg z_i^2 \Rightarrow \neg x_i^1 \ y_i^1 \neg z_i^1)$

$f_2 = \&_i (\neg x_i^4 \neg y_i^4 \ z_i^4 \Rightarrow \neg x_i^3 \neg y_i^3 \neg z_i^3)$

Q_1, \dots, Q_5 – объекты (дома)

X_i^j – j-й параметр i-го объекта (25 переменных)

x_i^j, y_i^j, z_i^j – двоичные разряды кода X_i^j

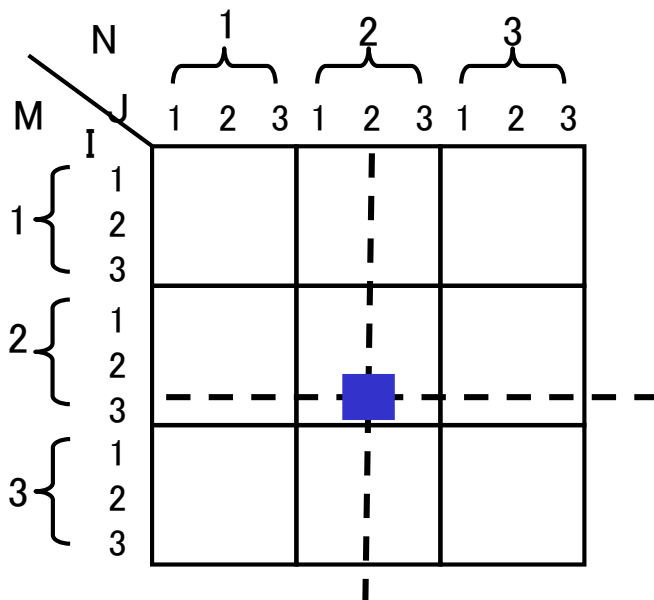
Возможные наборы параметров задаются теми наборами, на которых $F = \&_i f_i = 1$

BDD для функции F от 75 двоичных переменных решает задачу

Пример: сложные комбинаторные задачи

Судoku:

9						1	2	
			1		2			
		5		8		6		
	7		8					4
		4				9		2
	3						8	
4		2		6				9
5					9	3		
			7	4				



Цифры от 1 до 9, все разные:

на каждой горизонтали,
на каждой вертикали
и в каждом квадрате

$$A[M, N, I, J] \in \{1, 2, \dots, 9\}$$

$$M, N, I, J, m, n, i, j \in \{1, 2, 3\}$$

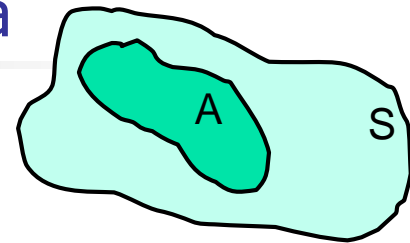
Горизонталь: $\forall M \forall N \forall I \forall J \forall n \forall j [(n \neq N) \vee (j \neq J) \Rightarrow A[M, N, I, J] \neq A[M, n, I, j]]$

Вертикаль: $\forall M \forall N \forall I \forall J \forall m \forall i [(m \neq M) \vee (i \neq I) \Rightarrow A[M, N, I, J] \neq A[m, N, i, J]]$

Квадрат: $\forall M \forall N \forall I \forall J \forall i \forall j [(i \neq I) \vee (j \neq J) \Rightarrow A[M, N, I, J] \neq A[M, N, I, j]]$

Для решения задачи нужна BDD для ф-ции от 324 переменных

Конечные структуры данных и BDD: характеристическая функция множества



Пусть S – конечное множество из m элементов,
 $k = \lceil \log_2(m) \rceil$. Закодируем произвольно все элементы S
векторами двоичных переменных x_1, x_2, \dots, x_k .

Пусть S состоит из 8 элементов: $\{a_0, a_1, \dots, a_7\}$.

Закодируем элементы S ($k=3$): $a_0 \Leftrightarrow 000, a_1 \Leftrightarrow 001, \dots, a_7 \Leftrightarrow 111$

Характеристическая булева функция множества из 2^m элементов = 1

Пусть $A \subseteq S$, $A = \{a_0, a_1, a_2, a_6\}$. Соответствующее множество
двоичных кодов $\{000, 001, 010, 110\}$.

Булева функция может задать подмножество конечного множества,
если она равна 1 на наборах, которые в это подмножество входят

$f(x_1, x_2, x_3)$ задает множество $\{000, 001, 010, 110\}$

Любое подмножество конечного множества может быть задано с помощью
булевой функции и, следовательно, с помощью ее BDD.

Эта функция называется характеристической функцией множества

Характеристические функции

Обозначим χ_A **характеристическую функцию** множества A. Она равна 1 на наборах, кодирующих элементы из A, т.е. на $\{000, 001, 010, 110\}$

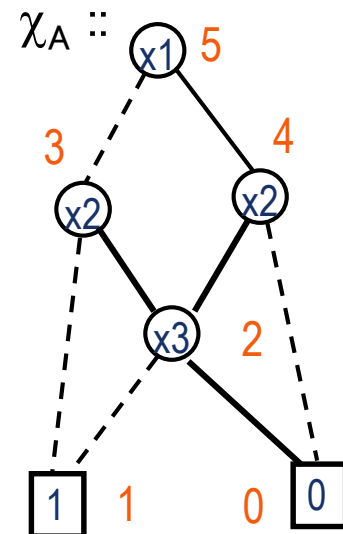
	f
000	1
001	1
010	1
011	0
100	0
101	0
110	1
111	0

Пусть x_1, x_2, x_3 – разряды кодировки.

Тогда:

$$\chi_A = \neg x_1 \neg x_2 \neg x_3 \vee \neg x_1 \neg x_2 x_3 \vee \neg x_1 x_2 \neg x_3 \vee x_1 x_2 \neg x_3$$

задает $A = \{000, 001, 010, 110\}$



Подмножества конечного множества можно задавать логической формулой, представляющей характеристическую функцию множества.

Будем писать $A = \{000, 001, 010, 110\} (x_1, x_2, x_3)$, чтобы показать переменные кодировки и их порядок



Операции над множествами

Нульарные операции (константы):

- полное множество: $\chi_S = \text{True}$
- пустое множество: $\chi_\emptyset = \text{False}$

Унарная операция:

- дополнение множества: $\chi_{S-Q} = \neg \chi_Q$

Бинарные операции:

- пересечение множеств: $\chi_{P \cap Q} = \chi_P \wedge \chi_Q$
- объединение множеств: $\chi_{P \cup Q} = \chi_P \vee \chi_Q$
- разность множеств: $\chi_{P-Q} = \chi_P \wedge \neg \chi_Q$

ВСЕ операции над множествами можно выразить через булевы операции над характеристическими булевыми функциями

Представление отношений с помощью BDD

Пусть $S = \{a_0, \dots, a_{n-1}\}$ – множество

Введем кодирование: $a_0 \Leftrightarrow 000, a_1 \Leftrightarrow 001, \dots, a_7 \Leftrightarrow 111$

Бинарное отношение на S – подмножество пар из S , $R = \{ (a_2, a_3), (a_7, a_4), (a_5, a_7) \}$

χ_R – характеристическая функция R равна 1 на кодировках $\{(a_2, a_3), (a_7, a_4), (a_5, a_7)\}$, т.е. на наборах 6 булевых переменных $\{ 010\ 011, 111\ 100, 101\ 111 \}$

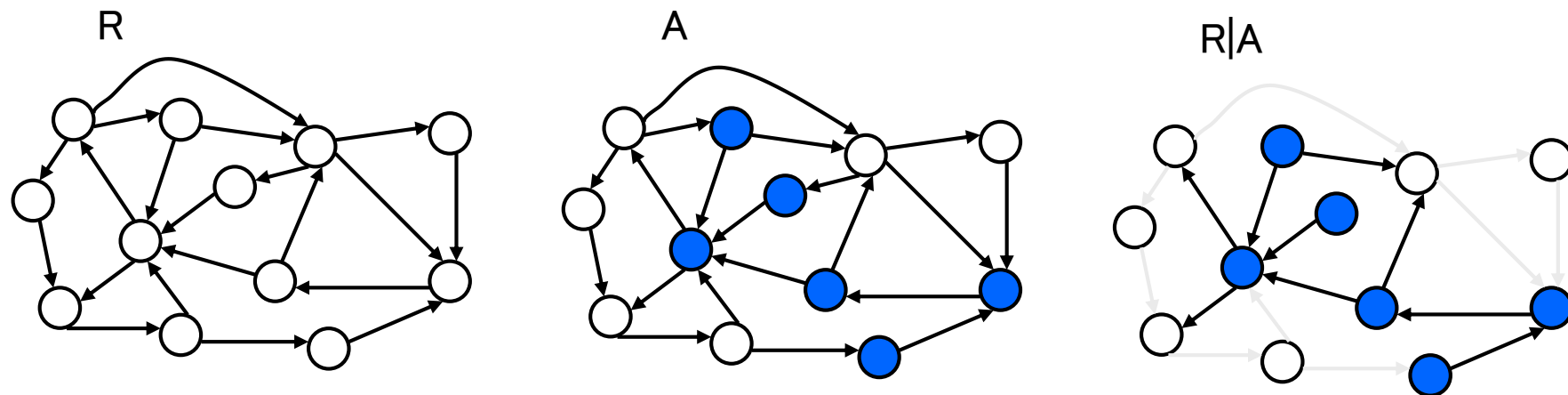
Первый элемент отношения R – это текущее состояние (pre-state), разряды его кода обозначим v (например, x_1, x_2, x_3). Второй элемент отношения – следующее состояние (post-state) и его код – v' (например, x_1', x_2', x_3')

Характеристическую функцию χ_R отношения R можно записать:

$$(\neg x_1 x_2 \neg x_3 \neg x_1' x_2' x_3') \vee (x_1 x_2 x_3 x_1' \neg x_2' \neg x_3') \vee (x_1 \neg x_2 x_3 x_1' x_2' x_3')$$

Характеристическая функция отношения может быть определена как логическая формула над штрихованными и нештрихованными переменными с одним и тем же порядком

Ограничение отношения на подмножестве



Пусть R – бинарное отношение на S , и A – подмножество S .

Обозначим χ_R - и χ_A – характеристические функции R и A

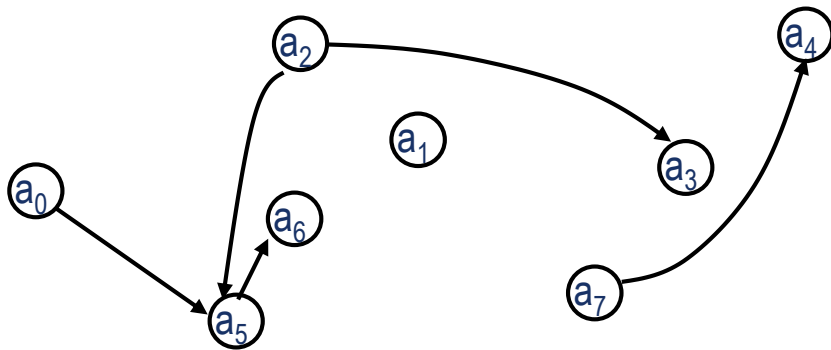
Характеристическая функция ограничения R на A строится как $\chi_A \& \chi_R$

Это все те пары отношения R , первый элемент которых принадлежит A

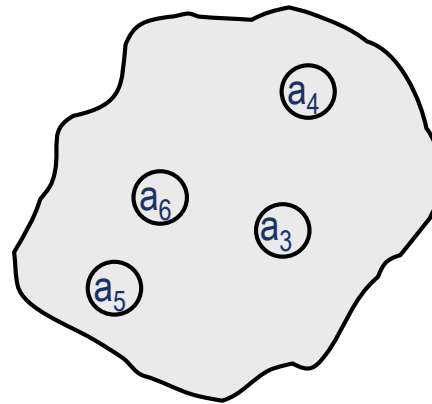
Операции над отношениями:

Прямой и обратный образы

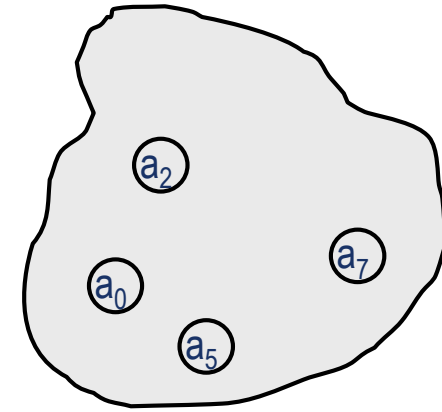
Пусть ХФ $\chi_R(v, v')$ определяет бинарное отношение (множество переходов)
 $S = \{a_0, a_1, \dots, a_7\}$



$\exists v. \chi_R(v')$



$\exists v'. \chi_R(v)$



ХФ множества тех состояний, **в которые** переходы возможны, задается операцией **Прямой образ**: $\text{Post}(R) = \exists v. \chi_R(v')$ - это ХФ множества $\{a_3, a_4, a_5, a_6\}$ - в них есть переходы из каких-то элементов множества S . (функции только от штрихованных переменных)

ХФ множества тех состояний, **из которых** переходы возможны, задается операцией **Обратный образ**: $\text{Pre}(R) = \exists v'. \chi_R(v)$ - это ХФ множества $\{a_0, a_2, a_5, a_7\}$ - из этих элементов есть переходы в какие-либо элементы множества S

Как реализовать эти функции в BDD?

Операция квантификации над функциями

Пусть множество $A = \{0010, 0101, 1011, 0110, 0011, 1100\}$ представлено характеристической функцией от булевых переменных (x_1, x_2, x_3, x_4)

Квантификация по переменной x_2 (это просто выбрасывание этой переменной) определяет множество

$$\begin{aligned} B &= \{0\mathbf{0}10, 0\mathbf{1}01, 1\mathbf{0}11, 0\mathbf{1}10, 0\mathbf{0}11, 1\mathbf{1}00\} \\ &= \{010, 001, 111, 010, 011, 100\} \\ &= \{010, 001, 111, 011, 100\} \end{aligned}$$

от переменных (x_1, x_3, x_4)

Определение квантификации характеристической функции:

$$\exists(x).f(x,y) = f(0,y) \vee f(1,y);$$

– результат не зависит от x . Это проекция на оставшиеся переменные

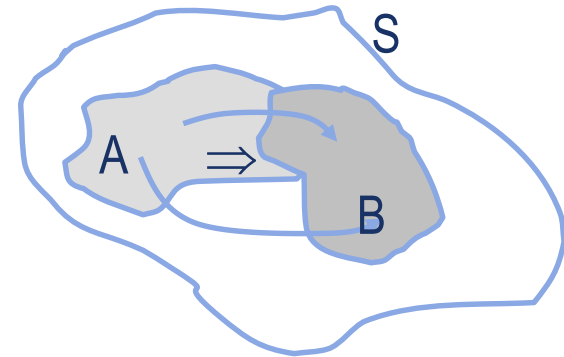
Пример: Операция квантификации: $\exists x_2. \chi_A$ определяет ХФ множества

$$\{0010, 0\mathbf{1}10, 0\mathbf{1}01, 0\mathbf{0}01, 1011, 1\mathbf{1}11, 0010, 0\mathbf{1}11, 1\mathbf{1}00, 1\mathbf{0}00\} (x_1, x_2, x_3, x_4) = \{010, 001, 111, 011, 100\} (x_1, x_3, x_4)$$

Квантификация по нескольким переменным вычисляется последовательно

Прямой Образ для бинарных отношений

Пусть $A \subseteq S$ – подмножество S ,
 R – бинарное отношение на S
В какие элементы S можно перейти из A ?



1. Определяем ограничение отношения R на тех начальных элементах R из S , которые принадлежат A :

$$\chi_{A(v)} \& \chi_{R(v, v')}$$

2. Строим $B = \text{Forward Image}(A, R) = \text{Прямой Образ } A \text{ относительно } R$:

$$\chi_{\text{Image}(A, R)(v)} = \exists v'. [\underbrace{\chi_{A(v)} \& \chi_{R(v, v')}}_{\text{Переходы из элементов } \in A}] \langle v/v' \rangle$$

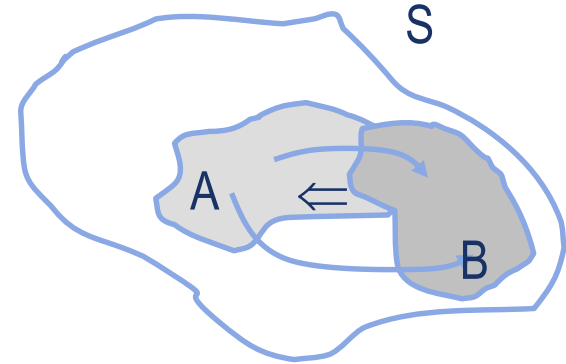
Переходы из элементов $\in A$

$\langle v/v' \rangle$ - это замена переменными v штрихованных значений v'

Итак, чтобы найти множество B всех тех элементов S , которые достижимы за один шаг отношения R из элементов множества A , нужно выполнить с булевыми характеристическими функциями $\chi_{A(v)}$ и $\chi_{R(v, v')}$ две операции: конъюнкцию и квантификацию, и выполнить переименование переменных

Обратный Образ для бинарных отношений

Пусть $B \subseteq S$ – подмножество S ,
 R – бинарное отношение на S
Из каких элементов S можно перейти в B ?



1. Строим ограничение отношения R на тех вторых элементах R из S , которые принадлежат B :

$$\chi_{B(v')} \& \chi_{R(v,v')}$$

2. Строим $A = \text{Reverse Image}(B, R) =$ обратный образ B относительно R :

$$\chi_{\text{RI}(\text{Image}(B, R))}(v) = \exists v'. (\chi_{B(v')} \& \chi_{R(v, v')}) - \text{выбрасываем все вторые элементы}$$

Чтобы найти множество A всех тех элементов S , из которых за один шаг отношения R достижимы элементы заданного множества B , нужно выполнить с булевыми характеристическими функциями $\chi_{B(v')}$ и $\chi_{R(v, v')}$ переименование и две операции: конъюнкцию и квантификацию

Решение проблем с помощью BDD

Пример: анализ достижимости на графе $\Gamma=(S,R,S_0)$

Пусть S_i – множество вершин, достижимых после i или меньшего числа переходов из S_0 .

Очевидно, $S_{i+1} = S_i \cup \text{Прямой Образ}(S_i, R)$

Алгоритм с помощью BDD

1. Представляем множества S_0 и R в форме BDD, т.е. определяем множество v переменных, кодирующих состояния из S , а также функции $\chi_{S_0}(v)$ и $\chi_R(v,v')$ в форме BDD
2. Полагаем $S_i = S_0$, т.е. $\chi_{S_i}(v) = \chi_{S_0}(v)$

Пост.проблемы

Выбор моделей

Разработка
(использование)
теории

Выбор структур
данных

Разработка
алгоритмов

Решение
проблемы

Решение проблем с помощью BDD

Пример: анализ достижимости на графе $\Gamma=(S,R,S_0)$

Пост.проблемы

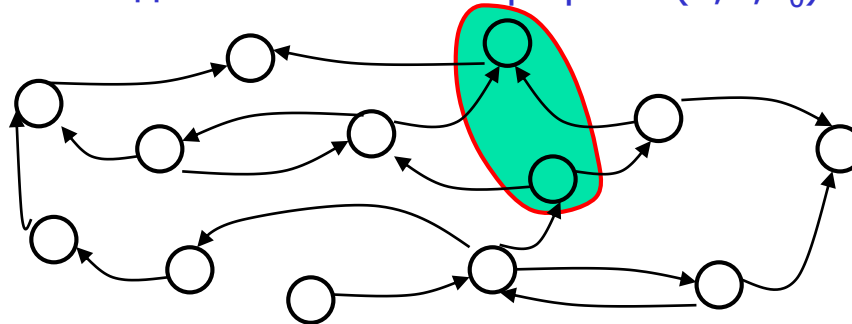
Выбор моделей

Разработка
(использование)
теории

Выбор структур
данных

Разработка
алгоритмов

Решение
проблемы



1. Представляем множества S_0 и R в форме BDD, т.е. определяем множество v переменных, кодирующих состояния из S , а также функции $\chi_{S_0}(v)$ и $\chi_{R(v,v')}$ в форме BDD

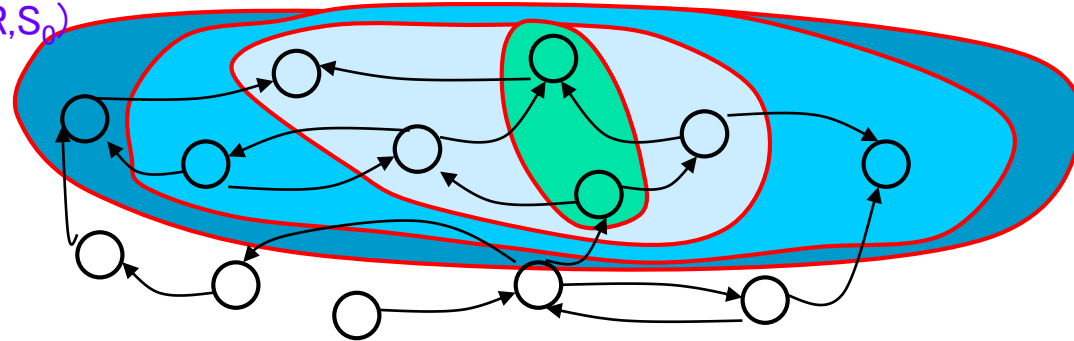
2. Полагаем $S_i = S_0$, т.е. $\chi_{S_i}(v) = \chi_{S_0}(v)$

3. Итеративно вычисляем ограничение R на S_i

$\chi_{R|S_i}(v,v') = \chi_{S_i}(v) \& \chi_{R(v,v')}$,
и куда можем перейти из S_i :
 $\chi_{S_{i+1}}(v) = [\chi_{S_i}(v) \vee \exists v'. \chi_{R|S_i}(v,v') (v / v')]$

Решение проблем с помощью BDD

Пример: анализ достижимости на графе $\Gamma=(S,R,S_0)$



1. Представляем множества S_0 и R в форме BDD, т.е. определяем множество v переменных, кодирующих состояния из S , а также функции $\chi_{S_0}(v)$ и $\chi_R(v,v')$ в форме BDD

2. Полагаем $S_i = S_0$, т.е. $\chi_{S_i}(v) = \chi_{S_0}(v)$

3. Итеративно вычисляем ограничение R на S_i

$\chi_{R|S_i}(v,v') = \chi_{S_i}(v) \& \chi_R(v,v')$,
и куда можем перейти из S_i :
 $\chi_{S_{i+1}}(v) = [\chi_{S_i}(v) \vee \exists v'. \chi_{R|S_i}(v,v') (v / v')]$

4. Алгоритм останавливается, когда $\chi_{S_{i+1}}(v) = \chi_{S_i}(v)$

Используем характеристические булевы функции для S_0 и R , а не работаем с каждой вершиной и ребром - сложность **полиномиальна на структурах, растущих экспоненциально**

Пост.проблемы

Выбор моделей

Разработка
(использование)
теории

Выбор структур
данных

Разработка
алгоритмов

Решение
проблемы

Randal Bryant (Carnegie Mellon Uni)



Randal Bryant

Carnegie Mellon University

Formal verification, binary decision diagrams

Подтвержден адрес электронной почты в домене cs.cmu.edu -

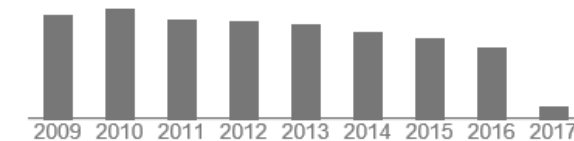
[Главная страница](#)

Подписаться

Google Академия

Индексы цитирований

	Все	Начиная с 2012 г.
Статистика цитирования	25268	4622
h-индекс	52	21
i10-индекс	125	43



Соавторы Все соавторы...

Jinbo Huang

Название	1-20	Процитировано	Год
Graph-based algorithms for boolean function manipulation		10357	1986
RE Bryant Computers, IEEE Transactions on 100 (8), 677-691			
Symbolic Boolean manipulation with ordered binary-decision diagrams		2567	1992
RE Bryant ACM Computing Surveys (CSUR) 24 (3), 293-318			
Efficient implementation of a BDD package		1587 *	1990
KS Brace, RL Rudell, RE Bryant ACM/IEEE Proc. 27th DAC, 40-45			
Semantics-aware malware detection		714	2005
M Christodorescu, S Jha, SA Seshia, D Song, RE Bryant Security and Privacy, 2005 IEEE Symposium on, 32-46			

Статья про BDD 1986 г. более 10 тыс раз использовалась в публикациях



Заключение

- БФ удобны для решения многих задач, но реально с классическими представлениями БФ можно работать только с 4-6 переменными
- BDD – новая компактная форма представления полностью определенных БФ в виде направленного графа. Сложность BDD зависит от порядка переменных
- Для большинства БФ представление в BDD линейно или полиномиально. Но существуют классы БФ, для которых представление в BDD экспоненциально
- BDD имеют множество привлекательных свойств:
 - каноническое представление
 - логические операции с BDD выполняются эффективно (линейно)!!!
 - многие задачи (например, SAT) решаются эффективно
 - BDD может представлять любые конечные структуры данных на основе использования булевых характеристических функций множеств, отношений, функций и т.п.
 - ...
- Основное свойство BDD – возможность работы с полиномиальным временем на структурах, растущих экспоненциально
- Существует много расширений BDD



Спасибо за внимание!