Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Казанский (Приволжский) федеральный университет»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ
И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Михайлов В.Ю.

ЛОГИКА ВЫСКАЗЫВАНИЙ КАК ОСНОВА СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

КАЗАНЬ - 2025

УДК 51-74

Учебное пособие публикуется по решению
учебно-методической комиссии Института вычислительной математики
и информационных технологий КФУ
Протокол № 7 от 21 марта 2025 г.

заседания кафедры системного анализа
и информационных технологий
Протокол № 6 от 14 марта 2025 г.

Автор-составитель
к.ф.-м.н. Михайлов В.Ю.

Рецензент
к.ф.-м.н. Пшеничный П.В.

Логика высказываний как основа создания интеллектуальных систем**:** Учебное пособие / Михайлов В.Ю.– Казань: Казанский университет, 2025, – 103 с.

Учебное пособие предназначено для студентов, изучающих курс «Математическая логика и теория алгоритмов», а также для преподавателей, ведущих лекционные и практические занятия по данному курсу.

СОДЕРЖАНИЕ

Введение

Настоящее пособие предназначено для студентов, начинающих своё знакомство с такой обширной и глубокой дисциплиной как математическая логика. Мотивационным посылом начального знакомства с языками и методами математической логики, как правило, является способность после освоения небольшого начального фрагмента успешно решать задачи, которые раньше казались студенту сложными.

По ходу продвижения в изучении материала в сознании студента традиционная задача математики: “заменить вычисления рассуждениями”, согласно работам Лейбница, инвертируется и превращается в задачу математической логики: “заменить рассуждения вычислениями”.

Студенты начальных курсов привыкли, что решение любой содержательной задачи состоит из следующих этапов:

1) Запись на некотором языке знаний, имеющихся у нас, на основе которых мы будем пытаться искать решение задачи. Как правило, это знания о предметной области задачи и условиях самой задачи (что дано?).

2) Точная запись вопроса задачи и точное понимание структуры объекта, который мы должны предъявить в качестве решения задачи.

3) Нахождение и реализация некоторой последовательности действий по построению промежуточных утверждений и объектов, на основе которых будет строиться искомый объект и обосновываться то, что он является решением исходной задачи.

Заметим, что для реализации первого и второго этапов каждый студент использует свой неформальный язык - некоторую смесь естественного языка, математических терминов и обозначений. Иногда по таким записям студентов бывает трудно понять, какую задачу описал и пытается решить студент.

Реализация же самого третьего этапа представляет для студента основную сложность и требует от него смекалки и так называемых ‘математических способностей’.

Настоящее пособие пропагандирует подход к решению задач, основанный на использовании математической логики, в частности, логики высказываний.

1) Языком для записи знаний о предметной области и знаний об условиях задачи выбирается язык логики высказываний. Знания о предметной области и условия задачи записываются в виде формул логики высказываний F1,…,Fk.

2) Вопрос задачи также записывается в виде формулы логики высказываний G.

3) Формулируется стандартная логическая задача, которая служит эквивалентом содержательной задачи.

В данном пособии к таким стандартным логическим задачам относятся:

проверка общезначимости формулы F; проверка выполнимости формулы F; проверка того, что формула G является логическим следствием формул F1, …, Fk.

Для решения стандартных логических задач применяются методы решения такие, как построение таблицы истинности, построение дерева Куайна, построение дизъюнктивной нормальной формы, вывод в секвенциальном исчислении высказываний, метод резолюций, по найденным решениям которых мы можем строить решения исходных содержательных задач.

Таким образом, основными целями пособия является:

1) Помощь студентам в изучении языка логики высказываний и отработки у них навыка записи на нем условий задачи, вопроса задачи и содержательных знаний о предметной области задачи.

2) Помощь студентам в освоении методов решения стандартных логических задач, таких как проверка общезначимости, выполнимости формул и задачи проверки логического следствия. Такими методами, достаточно подробно изложенными в пособии, являются поиск вывода в секвенциальном исчислении высказываний, метод резолюций построение BDD для формул логики высказываний. При изложении материала в пособии основной аспект делается на демонстрации решений задач. Теоремы, доказательства которых содержат теоретический материал, не содержащий описаний методов

решения задач, приводятся без доказательств. В качестве важных приложений описанных методов рассматриваются решение логических задач, концептуальное программирование Э.Х.Тыугу, символьное представление множеств.

.

Глава 1 . Язык логики высказываний. Синтаксис и Семантика

Язык логики высказываний = Формулы логики высказываний.

1. 1. Высказывания. Логические связки. Семантика связок

Элементарные высказывания принимают одно из двух значений {0,1}.

Пропозициональными переменными (обозначаются буквами) называются переменные, значениями которых являются элементарные высказывания.

1.1.1. Унарная связка

Отрицание: ¬A (читается: не A)

Смысл отрицания определяется таблицей:

x	¬x
0	1
1	0

1.1.2. Бинарные связки

Конъюнкция: A&B (читается A и B)

Дизъюнкция: Av B (A или B)

Импликация: A→B (из A следует B)

Результат импликации ложен только тогда, когда исходное (A) высказывание истинно, а результат (B) ложен.

Пояснение: (x делится на 4) → (x делится на 2) – истина при любом x. Поэтому (5 делится на 4) → (5 делится на 2) и (2 делится на 4) → (2 делится на 2) должны быть истинами.

Эквивалентность: A ≡ B (A равносильно B)

Результат эквивалентности есть истина, если A и B одновременно истинны, либо одновременно ложны (иными словами, если A=B).

Сложение по модулю 2: A⊕B (либо A, либо B)

Смысл этих связок определяется таблицей:

x1	**x2**	**x1&x2**	**x1vx2**	**x1→x2**	**x1 ≡ x2**	**x1⊕x2**
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0.	0	1	0	0	1
1	1	1	1	1	1	0

1.2. Формулы логики высказываний. Таблица истинности. Метод Куайна

Определение формулы логики высказываний. Формулой логики высказываний (пропозициональной формулой) называется строка символов, которая является либо пропозициональной переменной, либо совпадает с одной из строк (¬A), (A&B), (AvB), (A→B), (A ≡B), (A⊕B), где A и B – формулы.

Для сокращения числа скобок в формуле принято опускать скобки, не влияющие на результат. Соглашение о порядке выполнения (приоритете, силе связывания) операций, позволяет отбросить скобки, связывающие разные операции. Порядок выполнения логических операций следующий:

сначала выполняются операции в скобках, затем операция отрицания, далее - конъюнкция, дизъюнкция, импликация, эквивалентность, сложение по модулю 2.

Таблица истинности.

Смысл сложной формулы P логики высказываний определяется таблицей истинности этой формулы, в которой для каждого набора значений истинности входящих в неё элементарных высказываний указывается значение истинности самой формулы P.

Пример1.

Построим таблицу для формулы $p = x \wedge \bar{y} \to \bar{x} \vee y$

x	y	¬x	¬y	x∧¬y	¬x∨y	p
1	1	0	0	0	1	1
1	0	0	1	1	0	0
0	1	1	0	0	1	1
0	0	1	1	0	1	1

Для двух элементарных высказываний x и y перечисляются все возможные значения (их четыре), и затем для каждого набора, пользуясь таблицами связок, последовательно вычисляются значения подформул данной формулы.

Пример2.

Построим таблицу истинности для формулы Φ= (P→Q)&(R→S)&(P&R)→(Q&S).

P	Q	R	S	P→Q	R→S	P&R	Q&S	Φ
0	0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	0	1
0	0	1	0	1	0	0	0	1
0	0	1	1	1	1	0	0	1
0	1	0	0	1	1	0	0	1
0	1	0	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1
0	1	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1
1	0	1	0	0	0	1	0	1
1	0	1	1	0	1	1	0	1
1	1	0	0	1	1	0	0	1

P	Q	R	S	P→Q	R→S	P&R	Q&S	Φ
1	1	0	1	1	1	0	1	1
1	1	1	0	1	0	1	0	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

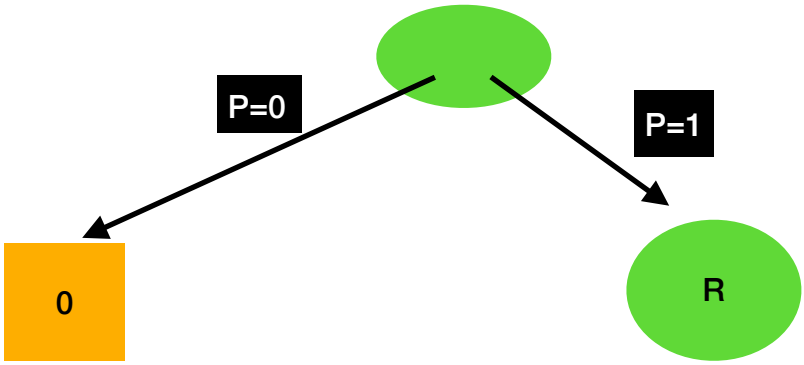
Заметим, что бездумное полное построение таблицы истинности, во-первых, занимает много времени, а во-вторых, содержит массу избыточных шагов, которые связаны с рассмотрением наборов, на которых одна из формул посылок очевидно принимает значение 0. Так на всех наборах, на которых P=0, третья посылка P&R принимает значение 0, а поэтому значения остальных посылок и заключения не имеют значения для решения задачи. Рассмотрим оставшиеся наборы, на которых P=1. Аналогично, на наборах, где R=0, значение третьей посылки равно 0, а стало быть значения остальных посылок и заключения не влияют на решение задачи. Осталось рассмотреть четыре набора на которых P=1, R=1. Здесь опять справедливо следующее соображение. Как только мы определили, что одна из посылок принимает значение 0, то значения других посылок и заключения мы можем не вычислять, т.к. они не влияют на решение задачи. Оформим эти рассуждения в виде сокращенной таблицы:

P	Q	R	S	P→Q	R→S	P&R	Q&S	Φ
0	-	-	-	-	-	0	-	1
1	-	0	-	-	-	0	-	1
1	0	1	0	0	-	-	-	1
1	0	1	1	0	-	-	-	1
1	1	1	0	1	0	-	-	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

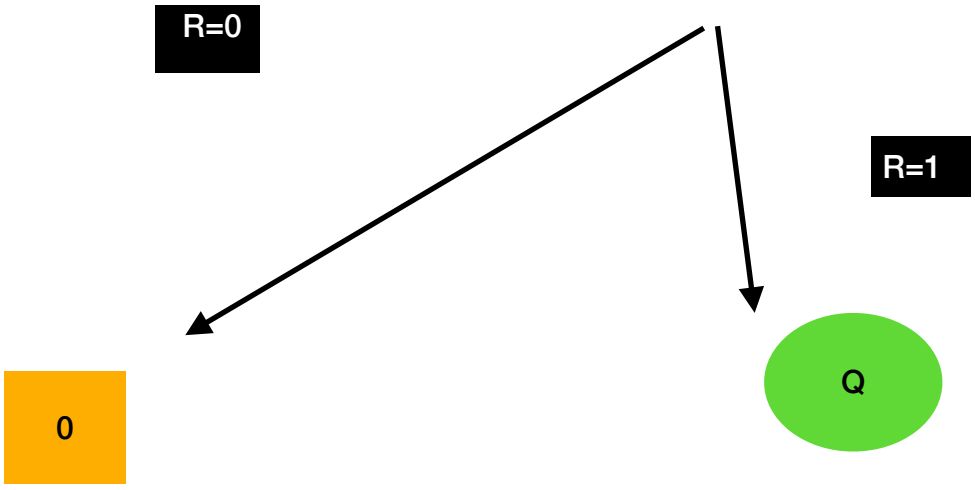
Построение сокращенной таблицы истинности осуществляется по **методу Куайна**. Он продемонстрирован на следующем рисунке. Мы фактически строим таблицу истинности формулы в виде дерева. Начальной вершине дерева приписывается формула, которая проверяется на выполнимость. Далее, к каждой вершине, к которой приписана некоторая формула Φ применяется следующее действие: вебирается некоторая переменная «расщепления» X, входящая в Φ, и у рассматриваемой вершины строятся две дочерние вершины, одна соответствует значению X=0 (левая), а вторая X=1 (правая). Левой вершине приписывается результат частичного вычисления Φ при X=0, а правой результат частичного вычисления Φ при X=1.

Метод Квайна - «splitting algorithm

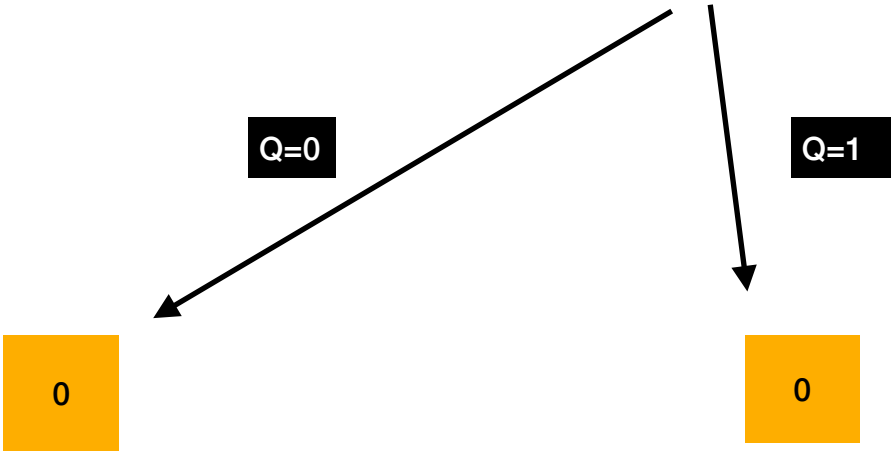
$\Phi = (P \to Q)\&(R \to S)\&(P\&R)\& \neg(Q\&S)$



$\Phi|_{p=1} = (1 \to Q)\&(R \to S)\&(1\&R)\& \neg(Q\&S) = (Q)\&(R \to S)\&(R)\& \neg(Q\&S)$



$\Phi|_{p=1|r=1} = (Q)\&(1 \to S)\& \neg(Q\&S) = (Q)\&(S)\&\neg(Q\&S)$



Если вершине приписывается формула 0 или 1, то ниже неё дерево уже не строится (говорят, что соответствующая ветвь дерева усекается). Процесс заканчивается построением так называемого замкнутого дерева (называемого часто сокращенным семантическим деревом Куайна). Каждой ветви замкнутого дерева соответствует частичный или полный набор значений переменных, входящий в сокращенную таблицу истинности.

Если в замкнутом дереве имеется вершина, которой приписана формула 1, то исходная формула является выполнимой.

Эффективность метода Квайна полностью определяется порядком выбора расщепляющих переменных. При удачно выбранном порядке расщепления переменных многие ветви могут быстро обрываться, что сделает размер построенного замкнутого дерева практически приемлемым.

1.3. Общезначимость. Выполнимость. Логическое следование

Определим ряд важных свойств формул логики высказываний.

Формула $\Phi(X_1, X_2, ..., X_n)$ логики высказываний называется *выполнимой*, если существует набор значений переменных $X_1, X_2, ..., X_n$, при котором Ф принимает значение 1.

Формула $\Phi(X_1, X_2, ..., X_n)$ называется *общезначимой* (*тавтологией*, *тождественно истинной*), если при любой подстановке на места переменных $X_1, X_2, ..., X_n$, конкретных значений формула $\Phi(X_1, X_2, ..., X_n)$ принимает значение 1. Обозначение тавтологии $\models \Phi$ (читается Φ –тавтология).

Формула $\Phi(X_1, X_2, ..., X_n)$ называется *опровержимой*, если найдутся значения переменных $X_1, X_2, ..., X_n$, при которых формула Ф принимает значение 0.

Формула $F(X_1, X_2, ..., X_n)$ называется *противоречием* (тождественно ложной формулой), если если при любой подстановке на места переменных $X_1, X_2, ..., X_n$, конкретных значений формула $\Phi(X_1, X_2, ..., X_n)$ принимает значение 0.

Формула логики высказываний G называется *логическим следствием* формул F1, F2, …, Fm**,** если для любых наборов значений переменных, входящих в формулы F1,F2, …, Fm, G, для которых истинны все формулы F1,F2,…, Fm**,** формула G тоже истина. Обозначение F1, F2, …, Fm |=G

Теорема о логическом следствии. Формула логики высказываний G является логическим следствием формулы F тогда и только тогда, когда формула F→G является тавтологией. Т.е. F |=G тогда и только тогда, когда |= F→ G

Обобщѐнная теорема о логическом следствии.

F1, F2, … , Fm |=G тогда и только тогда, когда

|=F1&F2& …& Fm→ G .

1.4. Основные задачи логики высказываний

1. Найти способ, позволяющий для любой формулы определить, является ли она выполнимой (так называемая проблема SAT).

2. Найти способ, позволяющий определить, является ли формула логики высказываний G логическим следствием формулы F.

3. Найти способ, позволяющий определить, является формула логики высказываний G логическим следствием формул F1, F2, …, Fm .

Очевидно, что все эти задачи эквивалентны и могут быть решены путѐм построения таблицы истинности для соответствующих формул логики высказываний. Существуют и другие способы решения этих задач (но на сегодняшний день неизвестны более эффективные). Для определенных классов формул логики высказываний такие эффективные методы разработаны.

Глава 2. Алгебра логики высказываний

2.1. Основные законы алгебры логики высказываний.

В логике высказываний довольно часто приходится проводить преобразования формул, сохраняющие эквивалентность. Для таких преобразований используются так называемые *основные законы логики высказываний*. Приведем список этих законов.

Пусть F,G и H – некоторые формулы логики высказываний. Тогда следующие формулы равносильны (эквивалентны):

1)$F\&1 \sim F$; 2) $F\vee 1 \sim 1$; 3) $F\&0 \sim 0$; 4) $F\vee 0 \sim F$; упрощающие законы с константами

5) $F\&F\sim F$; 6) $F\vee F \sim F$; законы идемпотентности

7) $F\&G \sim G\&F$; 8) $F\vee G \sim G\vee F$; законы комутативности

9)$F\&(G\&H)\sim(F\&G)\&H$; 10) $F\vee(G\vee H)\sim(F\vee G)\vee H$; законы ассоциативности

11) $F \,\&\, (G\vee H) \sim(F\&G) \vee (F\&H)$; 12) $F \vee (G\&H) \sim (F\vee G) \,\&\, (F\vee H)$; законы дистрибутивности

13) $F \,\&\, (F\vee G) \sim F$; 14) $F \vee (F\&G) \sim F$; законы поглощения

15) $F\&\neg F \sim 0$; 16) $F\vee\neg F \sim 1$; закон противоречия и закон исключенного третьего

17) $\neg(F\&G) \sim \neg F \vee \neg G$; 18) $\neg(F\vee G) \sim \neg F \,\&\, \neg G$; законы де Моргана

19) $\neg\neg F \sim F$; закон снятия двойного отрицания

20) $F\to G \sim \neg F \vee G$; закон выражения импликации через дизъюнкцию и отрицание

21) $F\leftrightarrow G \sim (F\to G) \,\&\, (G\to F)$; Закон выражения эквивалентности через импликацию и конъюнкцию

2.2. Нормальные формы в логике высказываний

Среди множества формул, равносильных данной, выделяют формулы, имеющие ту или иную нормальную форму. Дадим необходимые определения.

Определение. *Литералом* называется атомарная формула, кроме *1* и *0*, или ее отрицание. *Элементарной конъюнкцией* называется литерал или конъюнкция литералов.

Определение. Формула G имеет *дизъюнктивную нормальную форму* (сокращенно: ДНФ), если она является элементарной конъюнкцией или дизъюнкцией элементарных конъюнкций.

Например, формулы X, $\neg Y$, $X\&\neg Y$, $(X\&\neg Y) \vee (\neg X\&Z)$ находятся в ДНФ, а формулы $\neg(X\&Y)$, $X\vee Y\vee 1$, $X\to Y$ нет.

Теорема. Для всякой формулы F существует формула G, равносильная F и находящуюся в дизъюнктивный нормаль- ной форме.

Теорема легко следует из рассмотрения следующего алгоритма, который по данной формуле F выдает одну из формул G, удовлетворяющую условию теоремы.

Прежде, чем привести алгоритм, условимся не различать формулы, которые получаются одна из другой применением коммутативности и ассоциативности конъюнкции и дизъюнкции, т.е. законов 7–10.

Алгоритм приведения произвольной формулы логики высказываний к ДНФ:

Шаг 1. Используя законы 21 и 20, исключить из исходной формулы эквиваленцию и импликацию.

Шаг 2. С помощью законов 17–19 пронести отрицание в глубь формулы к атомарным формулам.

Шаг 3. Если формула содержит подформулу вида H_1 & ($H_2 \vee H_3$),

то заменить ее на равносильную формулу ($H_1 \& H_2$) ∨ ($H_1 \& H_3$).

Пример 1. Применение алгоритма проиллюстрируем на примере формулы

$F = \neg(X \leftrightarrow Y)$ & X.

Выполним первый шаг. Для этого, используя закон 21,

заменим $X \leftrightarrow Y$ равносильной ей формулой ($X \to Y$) & ($Y \to X$). Затем в полученной формуле с помощью закона 20 исключим связку →. Мы получим формулу

$F_1 = \neg[(\neg X \vee Y)$ & $(\neg Y \vee X)]$ & X.

Перейдем ко второму шагу. Применение закона 17, приведет к формуле

$F_2 = [\neg(\neg X \vee Y) \vee \neg(\neg Y \vee X)]$ & X.

Затем дважды воспользуемся законом 18 и снимем двойное отрицание (закон 19), получим формулу

$F_3 = [(X \& \neg Y) \vee (Y \& \neg X)]$ & X. Шаг 2 выполнен.

Выполнение шага 3 состоит из применения дистрибутивности к формуле F_3. Это дает нам формулу

$F_4 = (X \& \neg Y \& X) \vee (Y \& \neg X \& X)$.

Алгоритм на этом завершен. Формула F_4 имеет дизъюнктивную нормальную форму. Но эту формулу можно упростить. Действительно, формула $Y \& \neg X \& X$ в силу законов 15 и 3 равносильна *0*, а формула $X \& \neg Y \& X$ равносильна $X \& \neg Y$ (закон 5). Следовательно, формула F_4 равносильна формуле $F_5 = X \& \neg Y$.

Формула F_5, как и F_4, имеет ДНФ и равносильна исходной формуле F.

Нетрудно доказать, что если формула F невыполнима, т.е. при любой интерпретации F принимает значение *0*, то после приведения F к ДНФ каждая элементарная конъюнкция будет содержать хотя бы одну пару противоположных литералов X и $\neg X$. Но в таком случае на шаге 5 все элементарные конъюнкции будут вычеркнуты и получится пустая ДНФ.

На этом факте (что у каждой формулы существует эквивалентная ей ДНФ) основывается один из методов проверки выполнимости формул логики высказываний.

Теорема. Для всякой формулы F существует формула G, равносильная F и имеющая конъюнктивную нормальную форму.

Доказательство теоремы легко следует из анализа алгоритма приведения к КНФ, который в свою очередь получается из алгоритма приведения к ДНФ, если шаг 3 заменить на следующий

Шаг 3. Если формула содержит подформулу вида $H_1 \vee (H_2 \& H_3)$,

то заменить ее на равносильную ей формулу ($H_1 \vee H_2$) & ($H_1 \vee H_3$).

Нетрудно доказать, что если формула F неопровержима, т.е. при любой интерпретации F принимает значение *1*, то после приведения F к КНФ каждая элементарная дизъюнкция будет содержать хотя бы одну пару противоположных литералов X и $\neg X$. Но в таком случае на шаге 5 все элементарные конъюнкции будут вычеркнуты и получится пустая КНФ.

2.3. Построение нормальных форм формул логики высказываний, как способ решения задач справедливости логического следования

Упражнение 2.3.1. Используя определение логического следования, выяснить, выполняется ли логическое следование: $P\to Q, R\to S, P\&R \models Q\&S$.

Решение. Выясним, будет ли формула

((P → Q) & (R → S) & (P & R)) → (Q & S) тавтологией.

Пользуясь эквивалентностями

X→Y~¬X∨Y, ¬ (X∨Y)~¬X& ¬Y, ¬(X& Y)~¬X∨¬Y, ¬¬X~X, X&(X∨Y)~X, X∨(X&Y)~X, ¬X∨(X&Y)~¬X∨Y,

построим КНФ для данной формулы.

((P→Q)&(R→S)&(P&R))→(Q&S~

¬((¬ P ∨ Q) & (¬ R ∨ S) & (P & R)) ∨ (Q & S)~

(P & ¬ Q) ∨ (R & ¬S) ∨ ¬P ∨ ¬ R ∨ (Q & S) ~

¬ P ∨ ¬Q ∨ ¬ S ∨ ¬ R ∨ S~ 1

Так как¬ S∨S ~1, то (¬ P∨¬Q ∨¬ S ∨¬ R ∨S) ~1 и у нас получилась пустая КНФ.

Отсюда следует, что формула $((P\to Q)\&(R\to S)\&(P\&R))\to(Q\&S)$ является тавтологией, а формула $Q\&S$ логически следует из формул $P\to Q$, $R\to S$, $P\&R$.

Рассмотрим еще один способ проверки логического следования *методом от противного*. Пусть требуется выяснить, будет ли формула $G(X_1,X_2,...,X_n)$ логическим следствием формул $F_1(X_1,X_2,...,X_n),..., F_m(X_1,X_2,...,X_n)$. Предположим, что G не является логическим следствием формул $F_1,F_2,...,F_m$. Тогда, найдутся такие высказывания $P_1,P_2,...,P_n$, что высказывание $G(P_1,P_2,...,P_n)$ будет ложным, в то время как все высказывания $F_1(P_1,P_2,\ldots,P_n)$,..., $F_m(P_1,P_2,...,P_n)$ будут истинны.

Упражнение 2.3.2. Используя метод от противного, выяснить, выполняется ли логическое следование:

P→Q, R→S, P&R ⊨ Q&S.

Выясним, является ли формула (P→Q)&(R→S)& (P&R)&¬ (Q&S) выполнимой. Для этого построим ДНФ этой формулы, используя известные эквивалентности.

(P→Q)&(R→S)& (P&R)&¬ (Q&S) ~

(¬ P ∨ Q) & (¬ R ∨ S) & P & R & (¬ Q ∨¬ S) ~

Q & S & P & R & (¬ Q ∨¬ S) ~

Q & S & P & R &¬ S ~ 0.

Получили пустую ДНФ. Отсюда следует, что формула (P→Q)&(R→S)& (P&R)&¬ (Q&S) не выполнима, а логическое следование P→Q, R→S, P&R ⊨ Q&S справедливо.

Заметим однако, что применение законов дистрибутивности при построении ДНФ: (A ∨ B) ∧ (C ∨ D) ≡ (A ∧ C) ∨ (A ∧ D) ∨ (B ∧ C) ∨ (B ∧ D) может привести к экспоненциальному взрыву. Следовательно, с точки зрения решения проблемы выполнимости, построение ДНФ не лучше построения таблицы истинности.

2.4. Задачи для самопроверки

I. Выяснить является ли формула выполнимой:

а) ((A&B)→C)→(A∨B); б) ((A→B)→A)→B;

в) (A→B)→(B→A)→(B→A);

г) (A&(B∨C))→((C→(A→B))↔(B→(C→A))); д) ((A∨B)→B))&(A∨B);

е) (((B&C)→A)→(C→(A∨B)))&A.

II. Выяснить, какие из формул являются тавтологиями:

а) A∨B→A&B;

б) (A→B)→(B→A);

в) (A→(B→C))→((A→B)→(A→C)); г) (A↔B)↔((A→B)&(B→A));

д) ((A→B)→A)→A;

е) (A→C)→((B→C)→((A∨B)→C))

III.Установить, верно ли логическое следствие.

1. F→ G, ((F ∨ L) & H) →M, L →H |= ((F ∨L) & G) → ¬M
2. (P →Q) → (R &S), ¬R |= P& ¬Q
3. (F & G) → ¬R, (F&H) → K, F → ¬K, (F & ¬G) →H |= F→ ¬R
4. (P ∨ Q) → (R → S), ¬P & ¬Q |= R& ¬S
5. F→ G, ¬K → ¬L, S → H, ¬F → ¬K, H → L |= S → G
6. P → Q, R → S, P ∨ S |= Q ∨ R
7. (F & G) → H, (H & K) → L, ¬M → (K & L) |= (F & G) → M
8. P → Q, R → S, P & R |= Q & S
9. F → (G → H), (H & K) → L, ¬M → (K & ¬L) |= F → (G → M)
10. P → Q, P ∨ R |= (P ∨ R) → P & Q
11. (F ∨ G) → (H & K), (K ∨ L) → M |= F → M
12. (P → Q) → R |= P ∨ Q ∨ R
13. F → (G & H), ¬G ∨ K, (L → ¬M) → ¬K, G → (F & ¬L) |= G → L
14. P → (Q → R) |= (P → Q) → R
15. (F → G) & (H→K), (G→L) & (K → M), ¬(L&M), F → H |= ¬F

Методы решения: построение таблицы истинности, дерева Квайна, ДНФ

Глава 3. Выразительные возможности языка логики высказываний

3.1. Примеры задач

В занимательной книге Р. Смальяна «Принцесса или тигр» приведены ряд задач, решение которых требует от читателя некоторой смекалки и так называемого логического мышления. Сейчас мы опишем способ решения этих задач, основанный на стандартном подходе, принятом в математике. По условиям каждой задачи мы будем составлять уравнения, решения которых будут соответствовать решению исходной задачи.

Задачи 3.1.1. Принцесса или тигр

Во всех задачах этой группы узник помещается в комнату, в которой имеются две двери, за каждой из которых может находиться либо принцесса, либо тигр. Вполне может статься, что в обеих комнатах сидят тигры или там окажутся одни лишь принцессы (считается, что в первом случае узнику не повезло и никакая смекалка ему не поможет). Узник должен угадать, в какой комнате находится принцесса, а в какой — тигр. На двери каждой комнаты весит по одной табличке с некоторой информацией, а заключенному кое-что говорится об истинности этих сведений. Если узник не глуп и способен рассуждать логически, он сумеет сохранить себе жизнь и в придачу заполучить прелестную невесту.

3.1.1.1. Узник заходит в комнату и на дверях видит таблички следующего содержания:

T1: В этой комнате находится принцесса, а в другой комнате сидит тигр

Т2: В одной из этих комнат находится принцесса; кроме того, в одной из этих комнат сидит тигр

Далее узнику говорится, что на одной табличке сведения верные, на другой – нет. Какую дверь должен открыть узник? (Конечно, если он предпочитает принцессу тигру.)

Решение.

Введем две пропорциональные переменные П1 и П2.

Считаем, что П1 будет принимать значение 1, если высказывание «В первой комнате находится принцесса» истинно, и 0, если это высказывание ложно. П2 будет принимать значение 1, если высказывание «Во второй комнате находится принцесса» истинно, и 0, если это высказывание ложно.

Заметим, что в этом случае истинность логической формулы ¬П1 будет соответствовать истинности высказывания «В первой комнате находится тигр», а истинность формулы ¬ П2 - истинности высказывания «Во второй комнате находится тигр»

Значение переменных П1 и П2 мы должны определить.

Информация, записанная на первой табличке, может быть описана логической формулой

Т1 = П1&¬ П2, а информация на второй табличке – формулой Т2 = П1⊕ П2.

Тогда по условию задачи формула Т1⊕Т2 должна принимать значение 1.

Таким образом, решив логическое уравнение (П1&¬ П2)⊕ (П1⊕ П2) =1 ,т.е. найдя подходящие значение переменных П1 и П2, мы найдем решение исходной задачи.

Т.о. задача свелась к типовой задаче логики высказываний: проверить выполнимость логической формулы

(П1&¬ П2)⊕ (П1⊕ П2) .

Существует несколько способов решения подобных логических уравнений.

Самым простым в данном случае будет разбор всех возможных вариантов значений переменных П1 и П2, которые могут быть сведены в одну таблицу, которую принято называть таблицей истинности формулы:

П1	П2	П1&¬ П2	П1⊕ П2	(П1&¬ П2)⊕ (П1⊕ П2)
0	0	0	0	0
0	**1**	**0**	**1**	**1**
1	0	1	1	0
1	1	0	0	0

Таким образом, существует единственный набор значений переменных **П1=0 и П2=1**, при котором формула принимает значение 1. А это означает, что в описанной ситуации в первой комнате сидит тигр, а во второй- принцесса.

Мы видим, что единственным набором значений переменных, при котором формула примет значение 1 будет П1=1, П2=0, т.е. принцесса находится в первой комнате.

3.1.1.2. Узник заходит в комнату и на дверях видит таблички, на которых написано:

Т1: Что выбрать — большая разница

Т2: Лучше выбрать другую комнату

Относительно первой комнаты узнику было сказано: «Если в этой комнате находится принцесса, то утверждение на табличке истинно, если же тигр, ложно». Относительно второй комнаты было сказано: утверждение на табличке ложно, если в комнате находится принцесса, и истинно, если в комнате сидит тигр.

Какую дверь должен выбрать узник?

Решение.

Аналогично решению предыдущей задачи введем две пропозициональные переменные П1 и П2, с теми же значениями. Тогда содержание первой таблички может быть описано формулой

Т1≔П1⊕П2 (что означает, что содержимое комнат разное), а содержание второй таблички – формулой Т2≔П1&¬П2 (единственный вариант, при котором надо предпочесть первую комнату, соответствует случаю, когда в первой комнате находится принцесса, а во второй – тигр).

По условию задачи две следующие формулы должны быть истинными: Т1↔П1 и Т2↔¬П2 .

Таким образом, нам необходимо решить систему из двух логических уравнений:

(П1⊕П2)↔П1 = 1 и (П1&¬П2)↔¬П2 =1.

А для этого необходимо проверить, выполнима ли формула

((П1⊕П2)↔П1) & ((П1&¬П2)↔¬П2) ?

В данном случае также самым простым будет разбор всех возможных вариантов значений переменных П1 и П2, т.е. построение таблицы истинности:

П1	П2	(П1⊕П2)↔П1	(П1&¬П2)↔¬П2
0	0	1	0
0	1	0	1
1	**0**	**1**	**1**
1	1	0	1

Мы видим, что единственным набором значений переменных, при котором формула примет значение 1 будет П1=1, П2=0, т.е. принцесса находится в первой комнате.

Задача 3.1.2. Проверить совместимость множества утверждений.

1) Если курс ценных бумаг растет или процентная ставка снижается, то либо падает курс акций, либо налоги не повышаются.

2) Курс акций понижается тогда и только тогда, когда растет курс ценных бумаг и налоги растут.

3) Если процентная ставка снижается, то либо курс акций не понижается, либо курс ценных бумаг не растет.

4) Либо повышаются налоги, либо курс акций понижается и снижается процентная ставка.

Для этого необходимо представить предложения в виде пропозициональных формул и затем проверить, является ли их конъюнкция противоречием.

Обозначим элементарные высказывания:

В - курс ценных бумаг растет,

Р- процентная ставка снижается,

А- падает курс акций,

Н- налоги не повышаются.

Тогда

F1= (B v P) → (A⊕H); F2= A ≡ (B&¬H);

F3= P→ (¬A⊕¬B); F4=¬H⊕(A&P).

Задача сводится к следующей задаче логики высказываний: определить, является ли формула F1 & F2 & F3 & F4 выполнимой, т.е. имеются ли такие значения переменных B, P, A, H, при которых данная формула примет значение 1 (истина).

Для этого достаточно построить ДНФ для этой формулы. Построим сначала ДНФ для каждой из формул F1, F2 , F3 , F4.

F1= (B v P) → (A⊕H) ~ ¬ (B v P) v (A⊕H) ~ ¬ B &¬P v A &¬ H v ¬A & H

F2= A ≡ (B&¬H) ~ A & B&¬H v ¬ A & ¬(B&¬H) ~ A & B&¬H v ¬ A &(¬B v H) ~ A & B&¬H v ¬ A & ¬B v ¬ A &H

F3= P→ (¬A⊕¬B) ~ ¬P v (¬A&B) v (A&¬B)

F4= ¬ H⊕(A&P) ~ H&A&P v ¬ H& ¬(A&P) ~ H&A&P v ¬ H& (¬A v ¬P)) ~ H&A&P v ¬H&¬A v ¬ H & ¬P

Теперь заметим что, применяя законы дистрибутивности к формуле F1 & F2 & F3 & F4, мы легко обнаружим, что ДНФ этой формулы будет содержать элементарную конъюнкцию *¬B& ¬P& ¬A & ¬ H* , которая получается перемножением подчеркнутых элементарных конъюнкций в ДНФ формул F1 , F2 , F3 , F4. Отсюда следует, что при значениях переменных B=0, P=0,A=0,H=0 каждая из формул F1 , F2 , F3 , F4 примет значение 1, что и требовалось определить.

Задача 3.1.3. Проверить правильность рассуждения:

В бюджете возникнет дефицит, если не повысят пошлины (F1). Если в бюджете возникнет дефицит, то расходы на социальные нужды сократятся (F2). Следовательно, если повысят пошлины, то расходы на социальные нужды не сократятся (G).

Обозначим элементарные высказывания:

Б- в бюджете возникнет дефицит,

П- повысят пошлины,

С- расходы на социальные нужды сократятся.

Тогда F1= ¬П → Б, F2=Б → С, G=П→ ¬С.

Таким образом, задача сводится к проверке справедливости логического следствия

¬П → Б, Б → С |= П→ ¬ С.

Для этого построим ДНФ формулы F1&F2& ¬G и проверим её на пустоту:

(¬П → Б) & (Б → С) & ¬(П→ ¬С) ~

(¬¬П v Б) & (Б v С) & ¬(¬П v ¬С) ~

(П v Б) & (Б v С) & П & С ~

П & С

Мы получаем, что при П=1 и С=1 обе посылки F1, F2 принимают значение 1, а заключение G принимает значение 0. Вывод: рассматриваемое рассуждение логически не верно.

Задача 3.1.4. Четыре друга – Антонов (*A*), Вехов (*B*), Сомов (*C*) и Деев (*D*) решили провести каникулы в четырех различных городах: Москве, Одессе, Киеве и Ташкенте. Они договорились, что каждый из друзей будет отдыхать в

одном городе, и в каждом городе будет отдыхать лишь один из друзей. Определить, в какой город должен поехать каждый, если имеются ограничения:

(P) если *A* не едет в Москву, то *C* не едет в Одессу;

(Q) если *B* не едет ни в Москву, ни в Ташкент, то *A* едет в Москву;

(R) если *C* не едет в Ташкент, то *B* едет в Киев;

(S) если *D* не едет в Москву, то *B* не едет в Москву;

(T) если *D* не едет в Одессу, то *B* не едет в Москву.

Рассмотрим таблицу

	Москва	**Одесса**	**Киев**	**Ташкент**
Антонов				
Вехов				
Сомов				
Деев				

В ячейку на пересечении строки X и города У впишем пропорциональную переменную XУ , смысл которой «друг X едет в город У». Т.о. В ячейке с номером <3,2> , будет стоять переменная CL, соответствующая высказыванию «Сомов едет в Одессу», в ячейке с номером <2,4> , будет стоять переменная BT, соответствующая высказыванию «Вехов едет в Ташкент», и т.д.

Общие требования в задаче следующие:

- Каждый из друзей едет отдыхать только в один город, т.е. в каждой строке имеется ровно одна переменная принимающая значение 1,
- В каждом городе может отдыхать только один из друзей, т.е. в каждом столбце имеется ровно одна переменная, принимающая значение 1.

Отсюда следует, что конъюнкция любых двух переменных из какой-либо строки равна 0 и конъюнкция любых двух переменных из какого-либо столбца тоже равна 0.

Конкретные требования задачи могут быть записаны в виде формул:

$$P \equiv \neg A_M \to \neg C_L \cong A_M \vee \neg C_L;$$
$$Q \equiv (\neg B_M \wedge \neg B_T) \to A_M \cong B_M \vee B_T \vee A_M;$$
$$R \equiv \neg C_T \to B_K \cong C_T \vee B_K;$$
$$S \equiv \neg D_M \to B_M \cong D_M \vee B_M;$$
$$T \equiv D_L \to \neg B_M \cong D_L \vee \neg B_M.$$

Будем последовательно вычислять конъюнкцию этих формул:

$$P \wedge Q \equiv (A_M \vee \neg C_L) \wedge (B_M \vee B_T \vee A_M) \cong (A_M \wedge B_M) \vee (\neg C_L \wedge B_M) \vee$$
$$\vee (A_M \wedge B_T) \vee (\neg C_L \wedge B_T) \vee A_M \vee (\neg C_L \wedge A_M) \cong (A_M \wedge B_T) \vee A_M \vee (\neg C_L \wedge$$
$$\wedge B_M) \vee (\neg C_L \wedge B_T) \cong A_M \vee (\neg C_L \wedge B_M) \vee (\neg C_L \wedge B_T);$$
$$P \wedge Q \wedge R \equiv (A_M \vee (\neg C_L \wedge B_M) \vee (\neg C_L \wedge B_T)) \wedge (C_T \vee B_T) \cong (A_M \wedge$$
$$\wedge (C_T \vee B_K)) \vee ((\neg C_L \wedge B_M) \wedge (C_T \vee B_K)) \vee (\neg C_L \wedge B_T \wedge (C_T \vee B_T))) \cong$$
$$\cong (A_M \wedge C_T) \vee (A_M \wedge B_K) \vee (\neg C_L \wedge B_M \wedge C_T) \vee (\neg C_L \wedge B_M \wedge B_K) \vee (\neg C_L \wedge$$
$$\wedge B_T \wedge C_T) \vee (\neg C_L \wedge B_T \wedge B_K) \cong (A_M \wedge C_T) \vee (A_M \wedge B_K) \vee (\neg C_L \wedge B_M \wedge C_T);$$
$$P \wedge Q \wedge R \wedge S \equiv ((A_M \wedge C_T) \vee (A_M \wedge B_K) \wedge (\neg C_L \wedge B_M \wedge C_T)) \wedge (D_M \vee$$

$$\vee B_M) \cong (A_M \wedge C_T (\wedge D_M \vee B_M)) \vee (A_M \wedge C_T \wedge D_M) \vee (A_M \wedge C_T \wedge B_T) \vee (A_M \vee$$
$$\wedge B_K \wedge D_M) \vee (A_M \wedge B_K \wedge B_M) \vee (\neg C_L \wedge B_M \wedge C_T \wedge D_M) \vee (\neg C_L \wedge B_M \wedge C_T \wedge$$
$$\wedge D_M) \vee (A_M \wedge B_K \wedge B_M) \vee (\neg C_L \wedge B_M \wedge C_T \wedge D_M) \vee (\neg C_L \wedge B_M \wedge C_T \wedge$$
$$\wedge B_M) \cong \neg C_L \wedge B_M \wedge C_T.$$

Отсюда следует, что Вехов едет в Москву, Сомов едет в Ташкент. Из условия Т следует, что Деев едет в Одессу, а, следовательно, Антонов едет в Киев.

Интересный прием здесь – **неполная формализация** - не используются все 16 переменных вида XY и явно не формализуются ограничения о том, что один человек не может одновременно ехать в несколько городов, и что в одном городе не будут отдыхать несколько друзей. Вместо них используются дополнительные правила на выполнение операции конъюнкции над переменными.

3.2.Краткие сведения об алгоритме DPLL решения SAT-проблемы, одном из самых быстрых алгоритмов проверки выполнимости, которые были разработаны до настоящего времени.

Он использует идеи метода Куайна и свойства КНФ. Этот алгоритм часто называют алгоритмом Дэвиса-Патнем в честь авторов оригинальной статьи, в которой он был опубликован, Мартина Дэвиса и Хилари Патнем. Затем он фактически стал одной из версий алгоритмов, описанных Дэвисом, Логеманом и Лавлендом, поэтому он обычно называется DPLL по первым буквам фамилий всех четырех авторов.

Алгоритм DPLL принимает на входе некоторую формулу в конъюнктивной нормальной форме, представленную как множество дизъюнктов. На выходе он выдает набор значений переменных, при котором исходная формула выполнима, если такой набор существует, и «нет», если таких наборов нет.

Произвольный набор значений переменных, входящих в исходную формулу будем называть моделью. Модель может быть полной (если в нее входят все переменные рассматриваемой формулы) и частичной.

Алгоритм DPLL на семантическом дереве производит поиск выполняющей модели данной КНФ. Фактически он осуществляет рекурсивный перебор в глубину всех возможных моделей с возвратом в последнюю точку ветвления в случае неудачи. При этом в DPLL реализованы три описанных ниже усовершенствования.

1. ***Раннее завершение.*** Алгоритм на каждом этапе поиска пытается определить, является ли данная КНФ истинной или ложной, уже на построенной частичной модели. Элементарный дизъюнкт является истинным, если истинен любой его литерал, даже при том, что для других литералов еще не определены истинностные значения; поэтому об истинности всей КНФ в целом можно судить еще до того, как модель будет составлена полностью. Например, КНФ (AvB)&(AvC) является истинной, если истинен литерал A, независимо от значений литералов B и C. Аналогичным образом, КНФ является ложной, если ложен любой ее элементарный дизъюнкт, а это происходит, если каждый литерал какого-то дизъюнкта является ложным. Опять-таки, такая ситуация может возникнуть задолго до того, как модель будет полностью составлена. Раннее завершение позволяет обойтись без исследования целых поддеревьев в пространстве поиска.

2. ***Эвристика чистого символа.*** Чистым символом называется символ, который всегда появляется с одним и

тем же "знаком" во всех дизъюнктах. Например, в трех дизъюнктах (Av¬B), (¬Bv¬C), (CvA) символ A является чистым, поскольку он появляется только в виде положительных литералов; чистым можно также считать символ B, который появляется только в виде отрицательных литералов, а символ C считается нечистым.

Можно легко показать, что если некоторая КНФ имеет выполняющую модель и A – чистый символ, то имеется выполняющая модель, в которой символу A присвоено значение true, если все его вхождения положительны и присвоено значение false, если все его вхождения отрицательны. Следует отметить, что при определении чистоты символа алгоритм может игнорировать выражения, в отношении которых уже известно, что они истинны в модели, составленной до сих пор. Например, если модель содержит присваивание B= false, то дизъюнкт (¬Bv¬C) уже является истинным, а символ C становится чистым, поскольку присутствует только в дизъюнкте (CvA).

3. ***Эвристика единичного дизъюнкта.*** Единичный дизъюнкт это дизъюнкт с одним литералом. В контексте алгоритма DPLL единичные дизъюнкты возникают из дизъюнктов, в которых в построенной частичной модели всем литералам, кроме одного, уже было присвоено значение false. Например, если модель содержит присваивание B=false, то дизъюнкт (Bv¬C) становится единичным дизъюнктом, поскольку становится эквивалентным дизъюнкту ¬C. Очевидно, для того, чтобы этот дизъюнкт принял истинное значение, литералу C должно быть присвоено значение false. Эвристика единичного дизъюнкта предусматривает присваивание значений всем входящим в них символам до того, как произойдет переход к обработке оставшейся части КНФ. Следует отметить, что присваивание значения одному единичному дизъюнкту может привести к созданию еще одного единичного дизъюнкта; например, после присваивания символу C значения false единичным становится дизъюнкт (CvA), что влечет за собой присваивание истинного значения символу A=true. Такое "каскадное" распространение форсированных присваиваний называется распространением единичных дизъюнктов. Оно напоминает процесс прямого логического вывода (прямой волны) с применением хорновских формул. В действительности, если рассматриваемая формула в конъюнктивной нормальной форме содержит только хорновские выражения, то алгоритм DPLL по сути сводится к алгоритму прямого логического вывода.

Архитектура современных эффективных SAT-решателей. SAT- решателями называют программы, решающие проблему выполнимости формул логики высказываний. Здесь мы кратко опишем основные особенности архитектуры наиболее эффективных SAT-решателей. Подавляющее большинство современных SAT-решателей, показывающих высокие результаты на специализированных конкурсах, использует в качестве ядра алгоритм DPLL.

В некоторых случаях в процессе поиска выполняющей модели могут возникать (и возникают) так называемые конфликты. Конфликтом называется ситуация, когда из угаданных на текущий момент значений переменных для некоторой другой булевой переменной x выводятся одновременно x=0 и x=1. Если такое происходит, то вступает в действие процедура разрешения конфликта.

Переменные, значения которых угадываются, называют переменными уровней решений (decision level), уровни решений при этом нумеруются. На самом деле иерархию уровней решений можно представить в виде бинарного дерева, корнем которого служит первый уровень, помечаемый соответствующей пропорциональной переменной. Потомки корня соответствуют переменным, значения которых угадываются или индуцируются на последующих уровнях. Ветви, выходящие из произвольной вершины, помечаются значениями соответствующей переменной. Листья дерева помечаются как sat, либо как conflict. Путь из корня в лист, помеченный как sat, определяет (совместно с индуцированными на данном пути присвоениями) набор, выполняющий исходную КНФ. Путь из корня в лист, помеченный как conflict, определяет последовательность угадываний, из которой по правилу единичного дизъюнкта был выведен конфликт. Несложно понять, что если КНФ выполнима, то для некоторой альтернативы, исходящей из

корня данного дерева, найдется путь в лист, помеченный как sat. Если же КНФ невыполнима, то все пути из любой альтернативы корня будут оканчиваться листьями, помеченными как conflict. Очевидно, что для дерева поиска необходимо определить процедуры отсечения его поддеревьев, в которых любой путь из корня оканчивается листом, помеченным как conflict. Возможности нетривиальных отсечений приводят к сокращению перебора. Первоначально в DPLL был принят элементарный механизм разбора конфликтов — хронологический бэктрекинг. При хронологическом бэктрекинге разрешение конфликта происходит за счет изменения решения на последнем (перед конфликтом) уровне — значение угаданной на этом уровне переменной меняется на противоположное. Позже была сформулирована концепция нехронологического бэктрекинга (или бэкджампинга), которая привела к созданию по-настоящему скоростных SAT-решателей. Ключевой механизм бэкджампинга — процедура Clause Learning (далее — CL-процедура), позволяющая запоминать информацию о конфликтах. В общем случае использование CL-процедуры позволяет точно выявлять присвоения, ответственные за конфликт, вследствие чего возможны откаты не к последнему уровню принятого решения (как в хронологическом бэктрекинге), а к более ранним (в иерархии угадываний) уровням. В этом и состоит основной конструктивный момент нехронологического бэктрекинга (бэкджампинг). Следует отметить, что использование CL-процедуры приводит к росту объема памяти, задействуемой SAT-решателем. Помимо перечисленных, важнейшими компонентами современных эффективных SAT-решателей являются эвристики выбора переменных и специальные структуры данных, позволяющие совершать возвраты, оперируя при этом минимальным объемом информации. Эвристики выбора позволяют при выборе очередной переменной уровня решения руководствоваться некоторыми “разумными” предположениями. Первые эвристики выбора не использовали информации о ходе поиска и получили название статических. Позже были введены динамические эвристики, используемые в настоящее время в большинстве эффективных SAT-решателей. Первая динамическая эвристика выбора переменной на определенном уровне решения получила название VSIDS (Variable State Independent Decaying Sum). Основная ее идея заключается в присвоении переменным специальных индексов активности — активность переменной тем выше, чем чаще данная переменная принимает участие в конфликтах. Обычно динамические эвристики выбора переменных применяются в сочетании с рестартами. Согласно данной концепции, процесс поиска разбивается рестартами на фрагменты (этапы). После каждого рестарта угадывание переменных уровней решения происходит в соответствии с построенной на предыдущих этапах поиска таблицей активности переменных (таблица активности в каждом фрагменте поиска модифицируется). Описанный подход позволяет сохранять и эффективно использовать предысторию поиска. Были также разработаны так называемые быстрые структуры данных, предназначенные для эффективной реализации стратегии распространения булевых ограничений и организации возвратов с наименьшими вычислительными затратами. Можно увидеть, что после включения всех описанных выше усовершенствований алгоритм DPLL, несмотря на свой почтенный возраст, становится одним из самых быстрых алгоритмов проверки выполнимости, которые когда-либо были разработаны. В частности, реализация zchaff этого алгоритма используется для решения задач проверки качества аппаратного обеспечения с миллионом переменных.

Проблематика построения эффективных программных SAT-решателей становится в последние годы весьма интенсивно развивающимся направлением. В Internet регулярно проводятся конкурсы SAT-решателей и обновляются библиотеки тестовых примеров. Наиболее эффективные на данный момент SAT-решатели — это ***minisat, zchaff, berkmin.***

3.3. Задачи для самопроверки

I. Решение логических задач из книги Р. Смальяна

День второй испытаний

Во всех испытаниях этого дня относительно левой комнаты (комната I) король говорил вот что:

— Если в этой комнате находится принцесса, то утверждение на табличке истинно, если же тигр, ложно.

В правой же комнате (комната II) все было наоборот: утверждение на табличке ложно, если в комнате находится принцесса, и истинно, если в комнате сидит тигр. Ну и опять же, вполне может статься, что в обеих комнатах находятся принцессы или в них сидит по тигру, либо, наконец, в одной комнате пребывает принцесса, а в другой — тигр.

Четвертое испытание.

Объявив эти правила следующему узнику, король указал на две новые таблички:

I В обеих комнатах находятся принцессы

II В обеих комнатах находятся принцессы

Какую из комнат следует выбрать на этот узнику?

Испытание пятое.

Условия те же, а таблички вот какие:

I По крайней мере в одной из комнат находится принцесса

II Принцесса— в другой комнате

Испытание шестое.

Этой задачкой король особенно гордился, равно как и следующей за ней.

I Что ни выберешь — все едино

II Принцесса — в другой комнате

Как должен поступить узник?

Испытание седьмое.

Теперь на табличках было написано:

I Что выбрать — большая разница

II Лучше выбрать другую комнату

Испытание восьмое.

— На дверях же нет никаких табличек! — воскликнул следующий узник.

— Совершенно верно, — заметил король. — Их только что изготовили и не успели повесить.

— Так как же мне выбирать? — спросил узник.

— А вот эти таблички, — ответил король.

В этой комнате сидит тигр

В обеих комнатах сидят тигры

— Очень мило, — обеспокоился узник, — а какую куда?

Король призадумался.

— А тебе это знать вовсе не обязательно, — сказал он наконец. — Задача решается и так. Только не забудь, конечно, — добавил он, — что если принцесса в левой комнате, то утверждение на табличке у этой двери будет истинным, а если там тигр, то ложным. Для правой же комнаты — все наоборот.

Каково решение задачи в этом случае?

Раздел Психиатрическая лечебница

Однажды инспектора Крейга из Скотланд-Ярда срочно откомандировали во Францию для проверки одиннадцати лечебниц для умалишенных, где, по слухам, дела обстояли не слишком-то хорошо. В каждой из лечебниц единственными обитателями были пациенты и врачи — причем последние составляли весь персонал этих медицинских учреждений. Каждый обитатель лечебницы, будь то пациент или доктор, либо находился в здравом уме, либо был лишен рассудка. Кроме того, нормальные обитатели были абсолютно нормальны и на сто процентов уверены в том, что они говорят, они твердо знали, что все истинные утверждения действительно являются истинными, а все ложные — на самом деле ложными. В то же время безумные обитатели лечебниц придерживались совершенно противоположных представлений: все истинные утверждения они считали ложными, а все ложные утверждения — истинными. Наконец, надо полагать, что все обитатели лечебниц во всех случаях остаются честными — они всегда верят в то, что говорят.

1. Первая лечебница.

В первой же лечебнице, которую посетил Крейг, он беседовал по очереди с двумя обитателями, которых звали Джонс и Смит.

- Не могли бы вы рассказать мне, — обратился инспектор к Джонсу, — что вам известно о мистере Смите?

- Вам следовало бы называть его доктор Смит, — поправил Джонс. — Ведь это один из врачей нашей больницы.

Позже Крейг задал Смиту вопрос:

— Что вам известно о Джонсе? Он здесь пациент или доктор?

— Он пациент, — ответил Смит.

Поразмыслив некоторое время, инспектор смекнул, что дела в этой лечебнице и в самом деле идут не блестяще: либо один из докторов лишился рассудка и, значит, ему не следует продолжать работу в больнице умалишенных, либо, что еще хуже, один из пациентов является нормальным человеком и вообще не должен находиться здесь.

Как Крейг догадался об этом?

2. Во второй лечебнице.

В другой лечебнице, которую посетил Крейг, один из ее обитателей сообщил инспектору нечто такое, из чего тот смог сделать вывод, что говоривший был пациентом, но во вполне здравом уме, и потому его нужно было выпустить оттуда. Инспектор сразу же предпринял шаги для его освобождения. Не могли бы вы предложить пример такого сообщения?

3. В третьей лечебнице.

В следующей лечебнице некий обитатель высказал утверждение, из которого Крейг смог сделать вывод, что тот является лишившимся рассудка доктором.

Не могли бы вы сформулировать такое утверждение?

4. В четвертой лечебнице.

В следующей лечебнице Крейг спросил одного из ее обитателей:

— Вы пациент?

На что тот ответил: — Да.

Как обстоят дела в этой лечебнице?

Раздел Трансильвания

Одну часть населения Трансильвании составляют люди, а другую — упыри, причем люди всегда говорят правду, а упыри всегда лгут. Ситуация в этой стране крайне осложняется еще и тем, что половина всех жителей Трансильвании лишена рассудка и придерживается совершенно превратных представлений об окружающем их мире: так, все истинные суждения они считают ложными, а все ложные утверждения — истинными. Другая половина жителей психически здорова и абсолютно безупречна в своих суждениях, а именно: все истинные утверждения, по их мнению, являются истинными, про ложные же утверждения они знают, что те ложны.

Если же ложное суждение высказывает трансильванец, то он может делать это как просто из заблуждения, так и умышленно. Люди в здравом уме и упыри, лишившиеся рассудка, изрекают только истины; люди, лишившиеся рассудка, и упыри, находящиеся в здравом уме, всегда лгут. К примеру, если вы спросите у жителя Трансильвании, круглая ли Земля (или она плоская), то человек в здравом уме, зная, что Земля круглая, так и скажет. Человек же, лишившийся рассудка, считает, что Земля не является круглой, и потому, правдиво высказывая свое мнение, будет утверждать, что Земля плоская. Упырь в здравом уме знает, что Земля круглая, но поскольку он всегда лжет, то будет говорить, что это вовсе не так. В то же время лишившийся рассудка упырь уверен, будто Земля плоская и поскольку он всегда лжет, то станет утверждать, что Земля круглая. Таким образом, ответы упыря, лишившегося рассудка, совпадают с высказываниями нормального человека, в то время как утративший разум человек будет отвечать на задаваемые ему вопросы точно так же как и упырь, находящийся в здравом уме.

Первые пять расследований

В каждом из этих дел фигурировало по два обитателя Трансильвании. При этом заранее было известно, что один из них — человек, а второй — упырь, хотя и не было установлено кто же именно. По поводу состояния психики обитателей (исключая, впрочем, дело № 5) также не было никаких сведений. Задача расследования: определить, кто из персонажей является человеком, а кто упырем?

1. Дело Люси и Минны. По первому делу проходили две сестры, которых звали Люси и Минна. Крейгу предстояло определить, кто из сестер является упырем. Как уже отмечалось ранее, относительно состояния их психики ничего известно не было. Ниже приведена запись беседы инспектора с сестрами.

Крейг(обращаясь к Люси). Расскажите что-нибудь о себе и вашей сестре.

Люси. Мы обе не в своем уме.

Крейг (обращаясь к Минне). Это правда?

Мина. Конечно же, нет!

Исходя из этих ответов, Крейг, к всеобщему удовлетворению сразу сумел догадаться, которая из сестер является упырем. Кто же это был?

2. Дело братьев Лугози. Следующее дело было связано с братьями Лугози. Обоих братьев звали Бела, только один из них был упырем, а второй нет. Братья высказывали следующие утверждения.

Бела-старший. Я человек.

Бела-младший. Я человек.

Бела-старший. Мой брат вполне нормален.

Кто из них является упырем?

3. Дело Михаэля и Петера Карлофф. В следующем расследовании перед инспектором вновь предстали два брата — на этот раз Михаэль и Петер Карлофф. Вот что они заявили.

Михаэль Карлофф. Я упырь.

Петер Карлофф. Я человек.

Михаэль Карлофф. Психическое состояние моего брата совпадает с моим.

Кто из них упырь?

4. Дело де Роганов. В следующем расследовании оказались замешаны отец и сын де Роганы. Вот как выглядит запись беседы Крейга с ними.

Крейг (обращаясь к отцу). Вы оба в здравом уме или оба лишились рассудка? Или, может, вы отличаетесь друг от друга в этом отношении?

Отец. По крайней мере один из нас безумец.

Сын. Совершенно верно.

Отец. Но я-то, конечно, не упырь.

Кто из них является упырем?

5. Дело Карла и Марты Дракула. В последнем деле этой группы фигурировали двое близнецов — Карл и Марта Дракула (смею вас уверить, что в родстве со знаменитым графом они не состояли). Самое интересное в данном случае заключалось в том, что Крейгу было известно не только то, что один из них человек, а другой упырь, но и то, что один из близнецов в здравом уме, а другой лишился рассудка, хотя инспектор не имел ни малейшего представления, кто же именно. Вот запись их беседы.

Карл. Моя сестра — упырь.

Марта. Мой брат сошел с ума!

Кто из них является упырем?

II. Справедливы ли следующие рассуждения:

1. Если подозреваемый совершил кражу, то либо кража была тщательно подготовлена, либо имелся соучастник. Если бы кража была тщательно подготовлена, то был бы соучастник. Значит, подозреваемый не виновен в краже.

2. Намеченная атака удастся, только если захватить противника врасплох или же если позиции его плохо защищены. Захватить его врасплох можно только, если его позиции плохо защищены. Значит, атака не удастся.

3. Если бы у нее было много денег, она бы ездила в институт на такси и тогда бы никогда не опаздывала. Она постоянно опаздывает. Значит, у нее по-прежнему мало денег.

4. Если бы он хорошо знал английский язык или хотя бы она говорила помедленней, то он бы ее понял. Но он ее не понял. Значит, она как всегда говорила слишком быстро.

5. Муравей поднимет соломинку, если ее вес не превышает собственный вес муравья более, чем в 10 раз. Муравей не будет поднимать соломинку, если она ему не нужна. Муравей не стал поднимать соломинку. Значит, либо соломинка слишком тяжелая, либо муравью не нужна соломинка.

6. Если человек обедает в кафе быстрого питания, то он голоден и куда-то торопится. Человек не обедает в кафе быстрого питания, хотя и очень торопится. Значит, он не голоден.

7. Незнание правил дорожного движения не освобождает от ответственности в случае их несоблюдения. При нарушении правил водитель несет ответственность.. Следовательно, знать правила нужно.

8. Намеченная атака удастся, если захватить противника врасплох или его позиции плохо защищены. Захватить противника врасплох можно только, если он беспечен. Он не будет беспечен, если его позиции плохо защищены. Следовательно, намеченная атака не удастся.

9. Если Джонс не встречал этой ночью Смита, то Смит был убийцей или Джонс лжет. Если Смит не был убийцей, то Джонс не встречал Смита этой ночью, и убийство произошло после полуночи. Если убийство произошло после полуночи, то Смит был убийцей или Джонс лжет. Эксперты установили, что убийство произошло до полуночи. Следовательно, Смит был убийцей.

10. Если губернатор не имеет соответствующего авторитета или если он не желает принимать на себя ответственность, то порядок не будет восстановлен и волнения не прекратятся до тех пор, пока участникам волнений это не надоест, и власти не начнут примирительные действия. Следовательно, если губернатор не желает взять на себя ответственность и участникам волнений это не надоест, то волнения не прекратятся.

Порядок решения каждой задачи:

1. Записать рассуждение в логической символике и выписать формулу выполнимость которой необходимо проверить

2. Построить полную или усеченную таблицу истинности по методу Квайна

3. Проверить правильность рассуждения методом построения ДНФ

III. Проверить совместимость каждого из множества утверждений.

Для этого представить предложения в виде пропозициональных форм и затем построением ДНФ проверить, является ли их конъюнкция противоречием.

1. Если вечер скучен, то или Алиса начинает плакать, или Анатоль рассказывает смешные истории. Если Сильвестр приходит на вечер, то или вечер скучен, или Алиса начинает плакать. Если Анатоль рассказывает смешные истории, то Алиса не плачет. Сильвестр приходит на вечер тогда и только тогда, когда Анатоль рассказывает смешные истории. Если Алиса начинает плакать, то Анатоль рассказывает смешные истории.

2. Если курс ценных бумаг растет или процентная ставка снижается, то либо падает курс акций, либо налоги не повышаются. Курс акций понижается тогда и только тогда, когда растет курс ценных бумаг и налоги растут. Если процентная ставка снижается, то либо курс акций не понижается, либо курс ценных бумаг не растет. Либо повышаются налоги, либо курс акций понижается и снижается процентная ставка.

3. Либо свидетель не был запуган, либо, если Генри покончил жизнь самоубийством, то записка была найдена. Если свидетель был запуган, то Генри не покончил жизнь самоубийством. Если записка была найдена, то Генри покончил жизнь самоубийством.

IV. Задачи на частичную формализацию

1. Шесть спортсменов - Адамов, Белов, Ветров, Глебов, Дронов, Ершов - в проходившем соревновании заняли шесть первых мест, причем ни одно место не было разделено между ними. О том, кто какое место занял, были получены такие высказывания:

1) Кажется, первым был Адамов, а вторым-Дронов;

2) Нет, на первом месте был Ершов, а на втором – Глебов;

3) Вот так болельщики! Ведь Глебов был на третьем месте, а Белов – на четвертом;

4) И вовсе не так: Белов был пятым, а Адамов – вторым;

5) Все вы перепутали: пятым был Дронов, перед ним –Ветров.

Известно, что в высказывании каждого болельщика одно утверждение истинное, а другое ложное. Определите, какое место занял каждый из спортсменов.

Указание. Рассмотрите высказывания : Ai –«Адатов занял i-тое место» (i=1,2,3,4,5,6). Аналогичные значения

имеют символы Bi, Vi, Gi, Di, Ei. Высказывания болельщиков представьте в виде дизъюнкций. Все они должны быть истинными. Рассмотрите конъюнкцию этих истинных дизъюнкций. Преобразуйте эту конъюнкцию и, учитывая ее истинность, выведите распределение мест между спортсменами.

2. Для четырех дружинников, фамилии которых начинаются буквами A, E, P, C, необходимо составить график дежурств на четыре вечера подряд, учитывая, что:

1) C и P не могут дежурить в первый вечер в связи с командировкой;

2) если C выйдет во второй вечер или P – в третий, то E сможет подежурить в четвертый;

3) если A не будет дежурить в третий вечер, то E согласен дежурить во второй вечер;

4) если A или P будут дежурить во второй вечер, то C сможет пойти в четвертый вечер;

5) если P в четвертый вечер уедет на конференцию, то A придется дежурить в первый, а C в третий вечер.

3. При составлении расписания уроков на один день учителя математики, истории и литературы высказали следующие пожелания: математик просил поставить ему или первый, или второй урок; историк- или первый, или третий; учитель литературы – или второй, или третий. Как составить расписание, чтобы учесть все пожелания?

4. Для полярной экспедиции из восьми претендентов A, B, C, D, E, F, G и H надо отобрать шесть специалистов: биолога, гидролога, синоптика, радиста, механика и врача. Обязанности биолога могут выполнять E и G, гидролога – B и F, синоптика – F и G, радиста – C и D, механика – C и H, врача – A и D. Хотя некоторые претенденты владеют двумя специальностями, в экспедиции каждый сможет работать по одной специальности. Кого и кем следует взять в экспедицию, если F не может ехать без B, D- без H и без C, C не может ехать одновременно с G , а A не может ехать вместе с B ?

5. Некий остров населен жителями, каждый из которых либо всегда говорит правду, либо всегда лжет. Все жители отвечают на вопросы только «да» или «нет». К развилке дорог, из которых только одна ведет в столицу острова, подходит путешественник. Никаких знаков, указывающих, куда ведет каждая дорога, у развилки нет. Но здесь стоит местный житель, некто N. Какой вопрос, предусматривающий ответ «да» или «нет», должен задать ему путешественник, чтобы определить, какая дорога ведет в столицу острова?

Глава 4. Доказуемость в секвенциальном варианте исчисления высказываний

4.1. Поиск контрпримера для формулы

Пропозициональная формула общезначима, тогда и только тогда, когда не существует интерпретации языка логики высказываний, в которой эта формула ложна. Пусть A — формула, M — интерпретация языка логики высказываний; если M $\not\models$A, то M назовём контрпримером для A. Продемонстрируем метод систематического поиска контрпримера для формулы.

Пример 1. Рассмотрим формулу $\neg(p \wedge q) \supset (\neg p \vee r)$. Попробуем найти контрпример, т.е. интерпретацию, в которой эта формула примет значение 0. Будем записывать формулы, которые должны быть ложны в искомой интерпретации-контрпримере справа от знака →, а формулы, которые должны быть истинны — слева.

$\rightarrow \neg(p \wedge q) \supset (\neg p \vee r)$ - мы хотим эту формулу сделать ложной;

$\neg(p \wedge q) \rightarrow (\neg p \vee r)$ - чтобы импликация была ложной необходимо посылку сделать истиной, а формулу-заключение ложной;

$\rightarrow (\neg p \vee r), (p \wedge q)$ - для того, чтобы отрицание формулы сделать истинным необходимо саму формулу сделать ложной;

Чтобы конъюнкция формул p ∧ q стала ложной необходимо, чтобы одна из формул p, q стала ложной

→(¬p∨r),q →(¬p∨r),p;

Чтобы дизъюнкция двух формул стала ложной необходимо, чтобы обе эти формулы стали ложными

→ ¬p, r, q → ¬p, r, p ;

Чтобы отрицание формулы стало ложным необходимо, чтобы сама формула стала истиной

p → r, q p → r, p .

По выражению p → r,q можно задать контрпример M: M(p) = 1, M(r) = 0, M(q) = 0, остальным переменным интерпретация M сопоставляет какие угодно истинностные значения, для определённости 0.

Пример 2.

Рассмотрим формулу ¬(p ∧ q) ⊃ (¬p ∨ ¬q). Снова попробуем найти для нее контрпример. Но теперь запишем наши рассуждения в обратном порядке (снизу вверх):

q → ¬p, q p → ¬q, p

→ ¬p,¬q,q → ¬p,¬q,p

→(¬p∨¬q),q →(¬p∨¬q),p

→ (¬p ∨ ¬q), (p ∧ q)

¬(p∧q)→(¬p∨¬q)

→ ¬(p ∧ q) ⊃ (¬p ∨ ¬q)

Мы убеждается, что для исходной формулы не существует контрпримера, что равносильно общезначимости этой формулы. Поиск контрпримера для формулы осуществляется по нескольким правилам, точнее по двум правилам для каждой логической связки. Эти правила можно формализовать, задав так называемое исчисление.

4.2. Секвенциальное исчисление.

Сформулируем исчисление G, называемое секвенциальным исчислением высказываний (генценовского типа).

Секвенцией называется выражение вида $A_1,A_2,...,A_m \to B_1,B_2,\ldots,B_n$, где все $A_1,A_2,\ldots,A_m$, $B_1,B_2,...,B_n$ являются формулами. Список формул $A_1,A_2,...,A_m$ ($B_1,B_2,...,B_n$), который может быть пустым, называется *антецедентом (сукцедентом)* этой секвенции. Порядок формул в антецеденте и сукцеденте не имеет значения. Каждая формула A_i, i = 1,2,...,m (B_j, j = 1,2,...,n) называется членом антецедента (сукцедента). Любой член антецедента и сукцедента называется членом секвенции. Интерпретация называется *контрпримером* для секвенции S, если в этой интерпретации истинны все члены антецедента S и ложны все члены сукцедента S.

Представление секвенции в виде формулы. Для секвенции S вида $A_1,A_2,...,A_m \to B_1,B_2,...,B_n$ определим формулу Φ(S) так, что всякая интерпретация M является контрпримером для S, если и только если M является контрпримером для Φ(S).

Представление секвенции S в виде формулы Φ(S):

1) $A_1 \wedge A_2 \wedge ... \wedge A_m \supset B_1 \vee B_2 \vee \ldots \vee B_n$, если m>0 и n>0;

2) $1 \supset B_1 \vee B_2 \vee \ldots \vee B_n$, если m=0 и n>0;

3) $A_1 \wedge A_2 \wedge ... \wedge A_m \supset 0$,если m>0 и n=0;

4) 1⊃0, если m=n=0.

Таким образом, при $m > 0$ и $n > 0$ секвенцию S можно содержательно понимать так: из $A_1 \wedge A_2 \wedge \ldots \wedge A_m$ следует хотя бы одна из формул $B_1, B_2, \ldots, B_n$.

Назовём секвенцию S общезначимой, если формула Ф(S) общезначима. Очевидно, секвенция общезначима тогда и только тогда, когда не существует контрпримера для этой секвенции.

Правила вывода исчисления G

Пусть Γ, Δ обозначают любые конечные списки формул, A, B — любые формулы. Следующие фигуры являются правилами вывода в исчислении G.

$$\frac{\Gamma \to \Delta, A}{\Gamma, \neg A \to \Delta} \quad (\neg\to),$$

$$\frac{\Gamma, A \to \Delta}{\Gamma \to \Delta, \neg A} \quad (\to\neg),$$

$$\frac{\Gamma \to \Delta, A; \quad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B} \quad (\to\wedge),$$

$$\frac{\Gamma, A, B \to \Delta}{\Gamma, A \wedge B \to \Delta} \quad (\wedge\to),$$

$$\frac{\Gamma, A \to \Delta; \ \Gamma, B \to \Delta}{\Gamma, A \vee B \to \Delta} \quad (\vee\to),$$

$$\frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B} \quad (\to\vee),$$

$$\frac{\Gamma \to \Delta, A; \ \Gamma, B \to \Delta.}{\Gamma, A \supset B \to \Delta} \quad (\supset\to),$$

$$\frac{\Gamma, A \to \Delta, B}{\Gamma \to \Delta, A \supset B} \quad (\to\supset).$$

В левом столбце приведены правила введения логических связок в антецедент секвенции: правило введения отрицания в антецедент, правило введения конъюнкции в антецедент и т.д. В правом столбце приведены правила введения логических связок в сукцедент секвенции: правило введения отрицания в сукцедент и т.д.

Правила вывода и поиск контрпримера

Каждое правило вывода исчисления G представляет собой некоторую функцию f , сопоставляющую секвенцию одной или двум секвенциям (f не обязательно всюду определена).

Если $f(S_1;\ldots;S_n) = S$ (где n=1 или n=2), то говорят, что S получена из $S_1; \ldots; S_n$ по правилу вывода f , или что S является результатом применения правила вывода f к $S_1;\ldots;S_n$. Каждую секвенцию S_i (i = 1, . . . , n) называют посылкой, а секвенцию S — заключением данного применения правила. f называют n-посылочным правилом вывода.

Поиск контрпримера для формулы осуществлялся в соответствии с выше приведёнными правилами вывода. Подробнее, для каждого правила вывода и любой интерпретации M верно следующее: M является контрпримером для секвенции-заключения этого правила вывода тогда и только тогда, когда M является контрпримером хотя бы для одной

секвенции-посылки этого правила.

Аксиомы исчисления G

Пусть Γ обозначает любой конечный список формул, A — любую формулу. Аксиомами исчисления G являются секвенции следующих видов:

$\Gamma, A \to \Delta, A$; $\Gamma, 0 \to \Delta$; $\Gamma \to \Delta, 1$.

Очевидно, ни для какой аксиомы исчисления G не существует контрпримера.

Вывод в исчислении G

Выводом (или доказательством) в исчислении G называется конечная последовательность секвенций S_1 ; S_2 ; . . . ; S_k , в которой каждая секвенция S_i является аксиомой исчисления G или получена по некоторому правилу вывода исчисления G из некоторых секвенций S_{i_1} ; . . . ; S_{i_n} (где n = 1 или n =2) этой последовательности таких, что каждое $i_1,...,i_n$ меньше i. Число k называется длиной этого вывода.

Выводом секвенции S в исчислении G называется вывод $S_1;...;S_k$, в котором S_k совпадает с S.

Если существует вывод секвенции S в исчислении G, то секвенцию S называют выводимой в исчислении G (или теоремой исчисления G) и обозначают это как $\vdash_G S$ ($\vdash S$). Если секвенция S' не является выводимой в исчислении G, то S' называют невыводимой в G и обозначают это как $\nvdash_G S'$ ($\nvdash S'$).

4.3. Анализ вывода. Формулы, выводимые в исчислении G

Зачастую полезно дополнять вывод его анализом: каждую секвенцию в выводе снабжать номером и указанием на то, что она является аксиомой, или из каких секвенций и по какому правилу вывода получена данная секвенция. (Можно условиться дополнять вывод анализом другого вида, если это потребуется). Формула A называется выводимой в исчислении G, если секвенция $\to A$ выводима. Под выводом формулы A в исчислении G будем понимать вывод секвенции $\to A$.

Контрприменение правила вывода

Вывод заданной секвенции удобно искать, применяя правила от заключений к посылкам, аналогично тому как производился поиск контрпримера для формулы, только теперь можно действовать формально, в соответствии с правилами вывода, и не вспоминать о семантике. Нахождение посылок некоторого применения правила вывода по известному заключению S (говоря менее точно, применение этого правила от заключения к посылкам), называется контрприменением этого правила к S, а найденные посылки — посылками этого контрприменения.

Пример-вывод формулы.

Рассмотрим формулу $\neg(p \wedge q) \supset (\neg p \vee \neg q)$.

1) $\to \neg(p \wedge q) \supset (\neg p \vee \neg q)$ получается из секвенции 2 по правилу ($\to\supset$);

2) $\neg(p \wedge q) \to (\neg p \vee \neg q)$ получается из секвенции 3 по правилу ($\neg \to$);

3) $\to(\neg p \vee \neg q),(p \wedge q)$ получается из секвенций 4 и 5по правилу ($\to \wedge$);

4) $\to (\neg p \vee \neg q), p$ получается из секвенции 6 по правилу ($\to \vee$);

5) $\to (\neg p \vee \neg q), q$ получается из секвенции 8 по правилу ($\to \vee$);

6) $\to \neg p, \neg q, p$ получается из секвенции 7 по правилу ($\to \neg$);

7) $p \to \neg q, p$ —аксиома;

8) $\to \neg p, \neg q, q$ получается из секвенции 9 по правилу ($\to \neg$);

$q \to \neg p, q$ —аксиома.

Секвенция 1 (а также исходная формула) выводима, и последовательность секвенций 9; 8; . . . ; 2; 1 является выводом секвенции 1.

Дерево поиска вывода. Дерево вывода

Конечное дерево, узлами которого являются секвенции, называется деревом поиска вывода секвенции S, если корнем этого дерева является секвенция S, и для каждого внутреннего узла S' этого дерева сыновними узлами S' являются секвенции-посылки некоторого контрприменения правила вывода к S'.

Дерево поиска вывода традиционно изображается растущим снизу вверх.

Если все листья дерева поиска вывода секвенции S являются аксиомами, то это дерево называется деревом вывода секвенции S.

По дереву вывода произвольной заданной секвенции можно записать вывод этой секвенции в виде последовательности и наоборот.

4.4. Теоремы о корректности и полноте исчисления G

Теорема (корректность исчисления G).

Любая выводимая в исчислении G секвенция общезначима.

Теорема (полнота исчисления G).

Любая общезначимая секвенция выводима в исчислении G.

Пример поиска вывода не выводимой секвенции.

Попробуем найти вывод формулы ¬(p ∧ q) ⊃ (¬p ∨ r).

1) → ¬(p ∧ q) ⊃ (¬p ∨ r) получается из секвенции 2 по правилу (→⊃);

2) ¬(p ∧ q) → (¬p ∨ r) получается из секвенции 3 по правилу (¬ →);

3) →(¬p∨r),(p∧q) получается из секвенции 4 по правилу(→∨);

4) →¬p,r,(p∧q) получается из секвенции 5 по правилу(→¬);

5) p→r,(p∧q) получается из секвенций 6 и 7 по правилу(→∧);

6) p→r, p —аксиома;

7) p → r , q не выводима.

Секвенция 7 не является аксиомой и не содержит ни одной логической связки, поэтому секвенция 1, а вместе с ней и исходная формула, не выводима, т.е. не является общезначимой.

Дерево поиска вывода:

p → r, p p → r, q

p → r , (p ∧ q)

→ ¬p, r , (p ∧ q)

→ (¬p ∨ r), (p ∧ q)

¬(p ∧ q) → (¬p ∨ r)

→ ¬(p ∧ q) ⊃ (¬p ∨ r)

Вывод: исходная не является общезначимой. Контрпример: p=1, r=0, q=0.

Глава 5. Метод резолюций для логики высказываний

5.1. Основные понятия метода резолюций для логики высказываний

Этот раздел посвящен рассмотрению метода доказательства того, что формула G является логическим следствием формул F1,F2,…,Fk. Этот метод называется *методом резолюций*. Отметим, что задача о логическом следствии сводится к задаче о выполнимости. Действительно, формула G есть логическое следствие формул F1,F2,...,Fk тогда и только тогда, когда множество формул {F1,F2,...,Fk,¬G} невыполнимо. Метод резолюций, если говорить более точно, устанавливает невыполнимость. Это первая особенность метода. Вторая особенность метода состоит в том, что он оперирует не с произвольными формулами, а с дизъюнктами (или элементарными дизъюнкциями).

Напомним, что в логике высказываний литералом мы называли пропозициональную переменную или ее отрицание, дизъюнктом – дизъюнкцию литералов. Дизъюнкт может состоять из одного литерала. *На дизъюнкт мы будем смотреть, как на множество литералов,* т.е. не будем различать дизъюнкты, которые получаются один из другого с помощью коммутативности, ассоциативности и идемпотентности дизъюнкции. Последнее означает, например, что дизъюнкты X∨¬Y∨X и X∨¬Y будут для нас равны. Нам понадобится особый дизъюнкт – *пустой*, т.е. дизъюнкт, не содержащий литералов. Его мы будем обозначать "квадратиком" □. Будем считать, что пустой дизъюнкт ложен при любой интерпретации переменных. Это означает, что формула F&□ равносильна □, а формула F∨□ равносильна F. Пустой дизъюнкт есть фактически то же самое, что и атомарная формула 0, но в контексте метода резолюций принято использовать □.

Определение. Литералы L и ¬L называются *противоположными (контрарными)*.

Метод резолюций в логике высказываний основан на правиле резолюций.

Определение. *Правилом резолюций в логике высказываний* называется следующее правило: из дизъюнктов X∨F и ¬X∨G выводим дизъюнкт F∨G.

Например, из дизъюнктов ¬X∨Y∨Z и X∨¬Y выводим дизъюнкт Y∨Z∨¬Y. Обратим внимание на то, что в первых двух дизъюнктах есть еще одна пара противоположных литералов. Условимся, что можно применять правило резолюций не обязательно к самым левым литералам (поскольку мы не различаем дизъюнкты, отличающиеся порядком записи литералов). Тогда правило резолюций, примененное к контрактной паре Y и ¬Y, даст ¬X∨Z∨X. Условимся еще о следующем: в дизъюнктах не писать повторяющиеся литералы и не писать □ , если есть другие литералы.

Определение. Пусть S – множество дизъюнктов. *Выводом* из S называется последовательность дизъюнктов

D,D2,…,Dn такая, что каждый дизъюнкт этой последовательности принадлежит S или следует из предыдущих по правилу резолюций. Дизъюнкт D *выводим из* S, если существует вывод из S, последним дизъюнктом которого является D.

Например, если S={¬X∨Y∨Z, ¬Y∨U, X}, то последовательность

D1=¬X∨Y∨Z,

D2=¬Y∨U,

D3=¬X∨Z∨U,

D4=X,

D5=Z ∨U

– вывод из S. Дизъюнкт Z∨U выводим из S.

Применение метода резолюций основано на следующем утверждении, которое называется теоремой о полноте

метода резолюций.

Теорема. Множество дизъюнктов логики высказываний S невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Проверка невыполнимости логической формулы, представленной в виде КНФ.

Пусть формула F представлена в КНФ.

F = (x1 ∨ ¬x2) & (¬x1 ∨ x2) & (x2 ∨¬ x3) & (x3 ∨ ¬x2) & (x1 ∨ x3) & (¬ x1 ∨ ¬ x3)

Давайте проверим, является ли она невыполнимой, т.е. F ˜ 0?

Представим F как набор дизъюнктов (элементарных дизъюнкций).

S={x1 ∨ ¬x2, ¬x1 ∨ x2, x2 ∨¬ x3, x3 ∨ ¬x2, x1 ∨ x3, ¬ x1 ∨ ¬ x3}

Построим резолютивный вывод пустого дизъюнкт из S.

Исходные дизъюнкты 1-6:

1. x1 ∨ ¬x2

2. ¬x1 ∨ x2

3. x2 ∨ ¬x3

4. x3 ∨ ¬x2

5. x1 ∨ x3

6. ¬x1 ∨ ¬ x3

7. x1 ∨ ¬x3 (резольвента 1 и 3)

8. x1 (резольвента 7 и 5)

9. x2 (резольвента 2 и 8)

10. x3 (резольвента 10 и 4)

11. ¬x1 (резольвента 10 и 6)

12. □ (резольвента 11 и 8)– пустой дизъюнкт

Этот резолюционный вывод показывает, что исходное множество дизъюнктов не выполнимо.

Ответ: формула F - невыполнима.

Итак, для доказательства того, что формула G является логическим следствием множества формул F1,…,Fk метод резолюций применяется следующим образом. Сначала составляется множество формул T={F1,…,Fk,¬G}. Затем каждая из этих формул приводится к конъюнктивной нормальной форме и в полученных формулах зачеркиваются знаки конъюнкции. Получается множество дизъюнктов S. И, наконец, ищется вывод пустого дизъюнкта из S. Если пустой дизъюнкт выводим из S, то формула G является логическим следствием формул F1,…,Fk. Если из S нельзя вывести □, то G не является логическим следствием формул F1,…,Fk.

Проиллюстрируем сказанное на примере. Покажем, что формула G=Z является логическим следствием формул F1=¬X∨Y→X&Z, F2=¬Y→Z. Сформируем множество формул T={F1,F2,¬G}. Приведем формулы F1 и F2 к КНФ (формула ¬G сама имеет эту форму). Мы получим, что F1 равносильна X&(¬Y∨Z), F2 равносильна (Y∨Z).

Тогда множество дизъюнктов S равно {X, ¬Y∨Z, Y∨Z, ¬Z}.

Из множества S легко выводится пустой дизъюнкт:

1. ¬YvZ, исх. дизъюнкт из S

2.¬Z, исх. дизъюнкт из S

3.¬Y, резольвента 1 и 2

4.Y∨Z, исх. дизъюнкт из S

5.Y, резольвента 2 и 4

6.□ резольвента 3 и 5

Следовательно, формула G является логическим следствием формул F1, и F2.

Пример: Логический вывод в игре “Сапер”

Рассмотрим следующее положение в популярной игре Сапёр. Здесь необходимо определить, в каких клетках, обозначенных малыми латинскими буквами, стоят мины.Будем рассматривать a,b,c,d,t,f,g,h,i как пропозициональные переменные, соответствующими предложению, что в данной клетке находится мина. T.e. a=1, если в клетке a находится мина, и a=0, если в этой клетке мины нет. И так для каждой переменной a,b,c,d,t,f,g,h,i .Запишем теперь правила игры Сапер в виде пропорциональных формул Fi .

Запишем теперь правила игры Сапер в виде пропорциональных формул Fi .

1 ⇒ F1: a¬b ∨ ¬ab

2 ⇒ F2: ab¬c ∨ a¬bc ∨ ¬abc

1 ⇒ F3: b¬c¬d ∨ ¬bc¬d ∨ ¬b¬cd

2 ⇒ F4: cd¬e ∨ c¬de ∨ ¬cde

1 ⇒ F5: d¬e¬f ∨ ¬de¬f ∨ ¬d¬ef

2 ⇒ F6: ef¬g ∨ e¬fg ∨ ¬efg

1 ⇒ F7: f¬g¬h ∨ ¬fg¬h ∨ ¬f¬gh

2 ⇒ F8: gh¬i ∨ g¬hi ∨ ¬ghi

1 ⇒ F9: h¬i ∨ ¬hi

Формула R: **a** соответствует вопросу есть ли мина в клетке a? $\bigwedge_i F_i$ зависит от 9 переменных. ТИ этой функции - 512 строк). Ответ: “ДА!” (в поле ‘a’ находится мина) получен методом резолюции за 4 шага после преобразования $\wedge_i F_i$ в КНФ.

1. F11: a ∨ b

2. F12: ¬b ∨ ¬a

3. F21: ¬a ∨ ¬b ∨ ¬c

4. F22: a ∨ c

5. F23: b ∨ c

6. F24: a ∨ b

7. F31: ¬b ∨ ¬d

… … …

32.¬R: ¬a (отр. следствия)

33. b (рез. 32 и 1)

34. c (рез. 32 и 4)

35. ¬c (рез. 33 и 9)

36. □ (рез.34 и 35)

Центральным этапом применения метода резолюций является последовательное получение бинарных резольвент из исходных дизъюнктов и вновь получаемых дизъюнктов. На каждом шаге выбор из полученного множества дизъюнктов пары дизъюнктов для применения правила резолюции не является детерминированным. Часто для обеспечения полноты поиска вывода пустого дизъюнкта мы должны рассматривать все возможные пары, что делает применение метода резолюций для логики высказываний очень трудоемким и практически неэффективным.

Как правило, при решении практических задач применяются различные эвристики для сокращения выбора пар дизъюнктов для резольвирования. Если эти эвристики сохраняют полноту метода, то они называются *стратегиями.*

Кроме того, метод резолюций оказывается эффективным для ряда случаев, в которых исходные дизъюнкты имеют специфический вид.

5.2. Метод резолюций для хорновских формул

Определения. Для положительных литералов $P, L_i, i \in \{1, 2, \ldots n\}$ хорновские дизъюнкты могут иметь один из следующих видов:

1. L_1
2. $\neg L_1 \vee \neg L_2 \vee \cdots \vee \neg L_n$
3. $P \vee \neg L_1 \vee \neg L_2 \vee \cdots \vee \neg L_n$

Дизъюнкт Хорна с ровно одним положительным литералом называется ***определяющим дизъюнктом.*** Дизъюнкт Хорна без положительных литералов называется **целью или запросом**.

Формула Хорна — конъюнкция дизъюнктов Хорна, то есть формула в конъюнктивной нормальной форме, все дизъюнкты которой являются хорновскими.

Пример (определяющего) дизъюнкта Хорна:

$$\neg p \vee \neg q \vee \cdots \vee \neg t \vee u.$$

Эта формула может быть преобразована в эквивалентную формулу с импликацией:

$$(p \wedge q \wedge \cdots \wedge t) \rightarrow u$$

Пропозициональные формулы Хорна представляют интерес в теории сложности вычислений, где задача HORNSAT поиска множества истинностных значений, выполняющих конъюнкцию дизъюнктов Хорна, является P-полной. Это вариант из класса P для SAT — важнейшей NP-полной задачи.

Заметим, что резолюция двух определяющих хорновских дизъюнктов является хорновским дизъюнктом. Кроме того, резолюция цели и определяющего дизъюнкта также является хорновским дизъюнктом. Применение метода резолюций для проверки выполнимости хорновских формул или при решении задач логического следствия для хорновских формул оказывается очень эффективным в силу того, что в этом случае может применяться *стратегия* предпочтения одночленных дизъюнктов: *при выборе пары дизъюнктов для получения резольвенты один из дизъюнктов должен состоять из одной литеры.* Эта стратегия называется также *стратегией прямой волны*.

Далее, программная реализация стратегии прямой волны может быть организована таким образом, что время решения соответствующих задач будет линейным от общей длины исходных хорновских формул.

5.2.1. Стратегия прямой волны

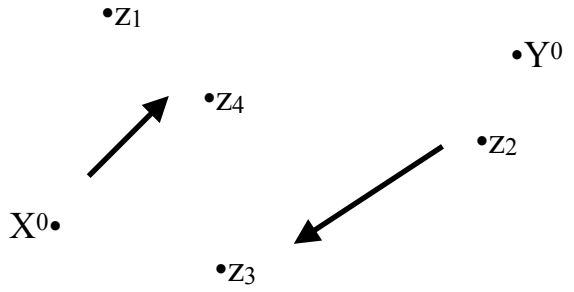
Пусть есть набор аксиом:

$$\begin{cases} x_1^1 \,\&\, x_2^1 \,\&\, \ldots \,\&\, x_{n_1}^1 \supset Y^1 & A_1 \\ \ldots\ldots\ldots\ldots\ldots & \vdots \\ x_1^N \,\&\, x_2^N \,\&\, \ldots \,\&\, x_{n_N}^N \supset Y^N & A_N \end{cases}$$

Задача : $x_1^0 \,\&\, x_2^0 \,\&\, \ldots \,\&\, x_{n_0}^0 \supset Y^0$ B

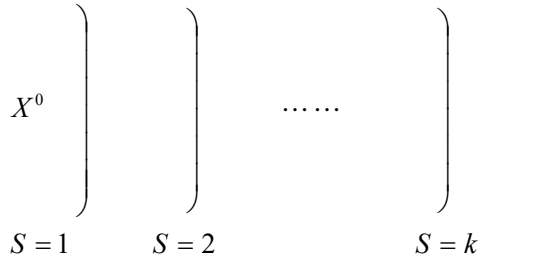
Доказать : $A_1, \ldots, A_N \models$ B.

Наводящее соображение: Предположим, что все ni=1. Рассмотрим список всех различных переменных {z1,…, zm}, имеющих вождения в аксиомы и задачу. Тогда набор аксиом может быть описан графом. Каждой переменной соответствует вершина.



Если имеется аксиома X ⊃ Y, то проводим ребро из вершины, соответствующей X, в вершину, соответствующую Y. Очевидно, что выводимость из аксиом формулы $X^0 \supset Y^0$ равносильна ответу на вопрос, есть ли путь в графе между вершинами, соответствующими X^0 и Y^0 ? То есть, наша задача сводится к поиску пути между двумя вершинами в ориентированном графе.

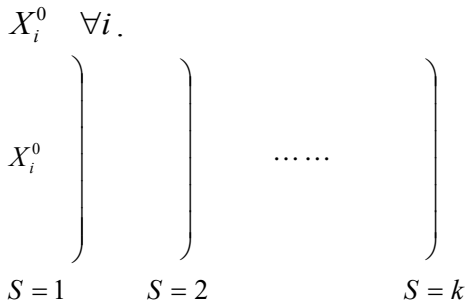
Задача решается с помощью алгоритма **прямой волны**. Источник X^0 (отсюда ищем путь). Выписываем все переменные, в которые можно перейти из X^0 за один шаг (фронт волны). Если фронт пустой, то задача не имеет решения. Иначе, если Y^0 попадает во фронт, то задача решена положительно. Если Y^0 не попадает во фронт, то продолжаем работу алгоритма.



k<m

Если на на каком-то шаге очередной фронт волны не содержит новых вершин и вершина Y^0 в него не попадает, то мы говорим, что нужного нам пути не существует. Если фронт пополнился новыми вершинами, а вершина Y^0 в него не попала, то строим новый фронт волны, добавляя в него новые вершины, в которые можно попасть из вершин текущего фронта за один шаг. Очевидно, что алгоритм всегда закончит свою работу. Обратим внимание, что построение нового фронта волны можно проинтерпретировать как применение правила резолюции : из дизъюнтктов $X \supset Y$ и X получаем резольвенту Y . Тогда алгоритм прямой волны для задачи поиска пути в графе соответствует применению метода резолюций со стратегией прямой волны.

Алгоритм переносится на общий случай, но в качестве ядра берется множество всех



На каждом этапе алгоритма просматриваем все аксиомы. Если все посылки некоторой аксиомы уже выведены, а заключение ещё нет, то включаем его во фронт. Если новый фронт пустой, то задача не имеет решения. Если Y^0 попадает во фронт, то задача решена положительно. Если Y^0 не попадает во фронт, то продолжаем работу алгоритма. Сложность алгоритма прямой волны:

$O(mL) \approx O(L^2)$, где $\sum_{i=0}^{n} n_i = L$, m - число этапов (фронтов волны), L- общая длина записи аксиом т.е. общее число вхождений переменных в запись аксиом.

Вывод: простая реализация стратегии прямой волны метода резолюций для решения задач на хорновских моделях работает квадратичное время.

Теперь рассмотрим более практичную реализацию этой стратегии.

5.2.2.Алгоритм со счётчиками

Пусть M – число переменных, N – число аксиом. Для каждой аксиомы Ai вводим счётчик Si, который на каждом этапе будет равен количеству ещё не выведенных переменных в посылке этой аксиомы.

В начале работы

$S_1=n_1$, $S_2=n_2$, …, $S_N=n_N$

Заведем следующие массивы:

1) ПЕР_В_ЛЕВ_Ч [1:M] значение i-той компоненты - список номеров аксиом, в левую часть которых входит переменная zi ;

2) ПЕР_ПРАВ_Ч [1:N] значение i-той компоненты - номер переменной, которая является правой частью i- той аксиомы;

3) **ВЫВ_П [1:M]** - булевский массив. i- тая компонента равна 1, если переменная zi выведена, и равна 0, если переменная zi еще не выведена.

4) **П_В_СПИСКЕ [1:M]** –булевский массив. i- тая компонента равна 1, если переменная zi выведена, но не «обработана» и включена в список для «обработки», и равна 0, если переменная zi не находится в списке для обработки.

Алгоритм:

Составляем список S для обработки переменных, куда в начале помещаем все переменные из

$x_1^0 \,\&\, x_2^0 \,\&\, \ldots \,\&\, x_{n_0}^0$

Отметим их в массиве **П_В_СПИСКЕ [1:M]**, чтобы не повторятся, и в массиве **ВЫВ_П [1:M]** , что они уже выведены. Берём последнюю переменную **p** из списка **S** и «обрабатываем» ее, т.е. находим в массиве **ПЕР_В_ЛЕВ_Ч** номера аксиом, в левую часть которых она входит, и уменьшаем на 1 соответствующие счётчики. Если некоторый счетчик становится равным 0, то берём переменную из правой части соответствующей аксиомы (в массиве **ПЕР_ПРАВ_Ч**). Если она не в списке для обработки, то метим ее, как выведенную, и помещаем в список для обработки. После этого переменная **p** исключается из списка **S**, и т.д. Алгоритм работает, пока список **S** не станет пустым.

Если по окончании алгоритма переменная Y^0 помечена как выведенная в массиве **ВЫВ_П[1:M]**, то «успех», иначе – «неудача».

Время работы этого алгоритма равно числу вхождений переменных в левую часть всех аксиом. Таким образом, алгоритм работает линейное время.

5.3. Концептуальное программирование (Э.Х. Тыугу)

Тыугу Э.Х. предложил представлять знания об предметной области в виде системы концептов. Концепт описывает некоторый объект и представляет из себя набор элементов объекта и набор взаимосвязей между ними, выраженных в форме уравнений.

Рассмотрим предметную область «планиметрия» и понятие «треугольник». Создадим концепт «треугольник». Для этого перечислим известные нам элементы треугольника: стороны a,b,c; углы α, β, γ; площадь S, периметр p, высоты ha, hb, hc, медианы, биссектрисы, радиусы вписанной и описанной окружностей и т.д. Для этого выделим набор пропорциональных переменных Δ (a,b,c, α, β, γ,S,p, ha, hb, hc,…) - высказывания, что соответствующий параметр треугольника может быть вычислен. Известные геометрические формулы переводятся в логические:

$$p = a + b + c \quad : \begin{cases} a \,\&\, b \,\&\, c \supset p \\ a \,\&\, b \,\&\, p \supset c \\ b \,\&\, c \,\&\, p \supset a \end{cases}$$

$$\alpha + \beta + \gamma = 180^\circ \quad : \begin{cases} \alpha \,\&\, \beta \supset \gamma \\ \alpha \,\&\, \gamma \supset \beta \\ \gamma \,\&\, \beta \supset \alpha \end{cases}$$

$$S = \frac{1}{2} h_a a \quad : \quad \begin{cases} h_a \,\&\, a \supset S \\ S \,\&\, a \supset h \\ S \,\&\, h_a \supset a \end{cases}$$

$$S = \frac{1}{2} ab \sin\gamma \quad : \begin{cases} a \,\&\, b \,\&\, \gamma \supset S \\ \ldots\ldots\ldots\ldots\ldots \\ S \,\&\, a \,\&\, b \supset \gamma \end{cases}$$

Таким образом, концепт «треугольник» представляет из себя набор хорновских формул вида

$$\begin{cases} x_1^1 \,\&\, \ldots \,\&\, x_{n_1}^1 \supset Y^1 \\ \ldots\ldots\ldots\ldots\ldots \\ x_1^N \,\&\, \ldots \,\&\, x_{n_N}^N \supset Y^N \end{cases}$$

которые мы будем называть аксиомами концепта. Решения вычислительных задач типа: «по известным значениям одних элементов треугольника вычислить значения других элементов» сводится к проверке логического следствия формулы вида

$$x_1^0 \,\&\, \ldots \,\&\, x_{n_0}^0 \supset Y^0$$

которую мы будем называть задачей, из аксиом концепта. С каждой аксиомой концепта обычно связывается вычислительная процедура, которая позволяет реально вычислять численное значение переменной-заключение по численным значением переменных-посылок. Таким образом, следующая последовательность шагов позволяет эффективно решать многие практически важные задачи:

- Построить модель предметной области - систему концептов.
- Сформулировать решаемую задачу в виде хорновской формулы.
- Обосновать возможность положительного решения задачи, установив ее логическое следствие из аксиом предметной области (тоже хорновских формул). Обоснование представляет из себя резолютивный вывод пустого дизьюнкта, возникающий при положительном решении задачи логического следствия.
- Используя идеи теоремы Клини-Нельсона, построить вычислительную процедуру (программу на заданном языке программирования), которая по численным значениям переменных из посылки будет вычислять численное значение переменной из заключения задачи.

Под названием концептуальное программирование мы понимаем процесс создания концептов предметной области.

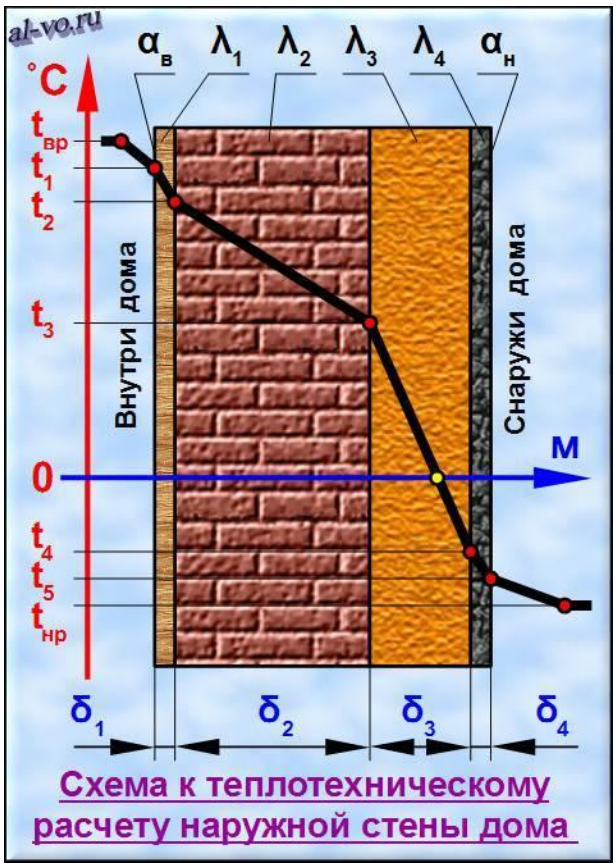
Процесс создания базовых концептов мы уже описали. Сложные концепты могут быть составлены из более простых с помощью несложных операций. Так в предметной области «планиметрия» после создания концепта «треугольник» концепт «параллелограмм» может быть описан, как два изоморфных треугольника, у которых одна сторона общая. Затем последовательно могут быть созданы концепты «трапеция», «прямоугольник», «квадрат», «ромб» и т.д.

Таким образом построены концептуальные модели многих практически важных предметных областей: радиотехника, военное дело, уход за животными и растениями и др.

Приведем **пример** базового концепта из предметной области строительство зданий. Назовем его «наружная стена».

Расчет стен жилых помещений - это процесс определения количества тепла, которое проходит через стены здания извне или наружу. Это важный аспект проектирования здания, который позволяет определить оптимальную конструкцию стен и выбрать правильные материалы для их изготовления для достижения наилучшей теплоизоляции.

Расчеты учитывают различные параметры, такие как теплопроводность материала стенки, толщина, коэффициент теплопередачи, температура внутри и снаружи комнаты и так далее. Давайте введем обозначения, которые используются для решения этой задачи:



R0 tr - базовое значение требуемого сопротивления теплопередачи конструкции ограждения (m2℃ /вт);

R0 усл (расчетное) сопротивление теплопередачи однородной части фрагмента теплозащитной оболочки здания (м2℃ /вт) ;

mp – коэффициент учитывающий особенности региона строительства;

ГСОП – градусо-сутки отопительного периода (℃×сут) ;

tot – средняя температура наружного воздуха (℃) ;

zot – продолжительность отопительного периода (сут/год) ;

Тв – расчетная температура внутреннего воздуха здания (℃) ;

a,b – расчетные коэффициенты, взятые из таблицы ;

ав – коэффициент теплопередачи внутри поверхности ограждающей конструкции (вт/m2℃) ;

ан -коэффициент теплопередачи наружной поверхности ограждающей конструкции (вт/ m2℃) ;

Rs - термическое сопротивление слоя однородной части фрагмента (м2℃ /вт);

δs - толщина слоя (м) ;

λs - расчетная теплопроводность материала слоя (W m2℃) ;

ys у.э. - коэффициент условий эксплуатации слоя материала слоя. Подробную информацию о расчетах и принципах определения значений для них можно найти в документах SP 50.13330.2012 «Термическая защита зданий», SP «Строительная климатология», также дополнительную информацию можно найти в видео на YouTube.

Опишем концепт:

$$\Delta(R_0^{\text{тр}}, R_{\text{о}}^{\text{усл}}, \text{ГСОП}, a, b, m_p, t_{\text{в}}, t_{\text{от}}, z_{\text{от}}, a_{\text{в}}, a_{\text{н}}, Sum_{R_s}, R_1, \delta_1 \lambda_1 y_1^{\text{у.э}} \dots)$$

- высказывания, что соответствующий параметр может быть вычислен

Известные физические формулы переводятся в логические:

$$\begin{cases} \text{ГСОП} = (t_{\text{в}} - t_{\text{от}}) \times z_{\text{от}} \\ t_{\text{в}} = \dfrac{\text{ГСОП}}{z_{\text{от}}} + t_{\text{от}} \\ t_{\text{от}} = t_{\text{в}} - \dfrac{\text{ГСОП}}{z_{\text{от}}} \\ z_{\text{от}} = \dfrac{\text{ГСОП}}{t_{\text{в}} - t_{\text{от}}} \end{cases} \qquad \begin{cases} t_{\text{в}} \wedge t_{\text{от}} \wedge z_{\text{от}} \supset \text{ГСОП} \\ \text{ГСОП} \wedge z_{\text{от}} \wedge t_{\text{от}} \supset t_{\text{в}} \\ t_{\text{в}} \wedge \text{ГСОП} \wedge z_{\text{от}} \supset t_{\text{от}} \\ \text{ГСОП} \wedge t_{\text{в}} \wedge t_{\text{от}} \supset z_{\text{от}} \end{cases}$$

$$\begin{cases} R_{\text{о}}^{\text{тр}} = (a \times \text{ГСОП} + b) \times m_p \\ a = \dfrac{R_{\text{о}}^{\text{тр}}}{\text{ГСОП} \times m_p} - \dfrac{b}{\text{ГСОП}} \\ \text{ГСОП} = \dfrac{R_{\text{о}}^{\text{тр}}}{a \times m_p} - \dfrac{b}{a} \\ b = \dfrac{R_{\text{о}}^{\text{тр}}}{m_p} - a \times \text{ГСОП} \\ m_p = \dfrac{R_{\text{о}}^{\text{тр}}}{a \times \text{ГСОП} + b} \end{cases} \qquad \begin{cases} a \wedge \text{ГСОП} \wedge b \wedge m_p \supset R_{\text{о}}^{\text{тр}} \\ R_{\text{о}}^{\text{тр}} \wedge \text{ГСОП} \wedge m_p \wedge b \supset a \\ R_{\text{о}}^{\text{тр}} \wedge a \wedge m_p \wedge m_p \supset \text{ГСОП} \\ R_{\text{о}}^{\text{тр}} \wedge m_p \wedge a \wedge \text{ГСОП} \supset b \\ R_{\text{о}}^{\text{тр}} \wedge a \wedge \text{ГСОП} \wedge b \supset m_p \end{cases}$$

$$\begin{cases} R_{\text{о}}^{\text{усл}} = \dfrac{1}{a_{\text{в}}} + Sum_{R_s} + \dfrac{1}{a_{\text{н}}} \\ a_{\text{в}} = \dfrac{1}{R_{\text{о}}^{\text{усл}}} - \dfrac{1}{Sum_{R_s}} - a_{\text{н}} \\ Sum_{R_s} = R_{\text{о}}^{\text{усл}} - \dfrac{1}{a_{\text{в}}} - \dfrac{1}{a_{\text{н}}} \\ a_{\text{н}} = \dfrac{1}{R_{\text{о}}^{\text{усл}}} - a_{\text{в}} - \dfrac{1}{Sum_{R_s}} \end{cases} \qquad \begin{cases} a_{\text{в}} \wedge Sum_{R_s} \wedge a_{\text{н}} \supset R_{\text{о}}^{\text{усл}} \\ R_{\text{о}}^{\text{усл}} \wedge Sum_{R_s} \wedge a_{\text{н}} \supset a_{\text{в}} \\ R_{\text{о}}^{\text{усл}} \wedge a_{\text{в}} \wedge a_{\text{н}} \supset Sum_{R_s} \\ R_{\text{о}}^{\text{усл}} \wedge a_{\text{в}} \wedge Sum_{R_s} \supset a_{\text{н}} \end{cases}$$

$$\begin{cases} Sum_{R_s} = R_1 + R_2 + R_3 + R_4 \\ R_1 = Sum_{R_s} - R_2 - R_3 - R_4 \\ R_2 = Sum_{R_s} - R_1 - R_3 - R_4 \\ R_3 = Sum_{R_s} - R_1 - R_2 - R_4 \\ R_4 = Sum_{R_s} - R_1 - R_2 - R_3 \end{cases} \qquad \begin{cases} R_1 \wedge R_2 \wedge R_3 \wedge R_4 \supset Sum_{R_s} \\ Sum_{R_s} \wedge R_2 \wedge R_3 \wedge R_4 \supset R_1 \\ Sum_{R_s} \wedge R_1 \wedge R_3 \wedge R_4 \supset R_2 \\ Sum_{R_s} \wedge R_1 \wedge R_2 \wedge R_4 \supset R_3 \\ Sum_{R_s} \wedge R_1 \wedge R_2 \wedge R_3 \supset R_4 \end{cases}$$

$$\begin{cases} R_s = \dfrac{\delta_s}{\lambda_s} \times y_s^{\text{у.э}} \\ \delta_s = \dfrac{R_s \times \lambda_s}{y_s^{\text{у.э}}} \\ \lambda_s = \dfrac{\delta_s \times y_s^{\text{у.э}}}{R_s} \\ y_s^{\text{у.э}} = \dfrac{R_s \times \lambda_s}{\delta_s} \end{cases} \qquad \begin{cases} \delta_s \wedge \lambda_s \wedge y_s^{\text{у.э}} \supset R_s \\ R_s \wedge \lambda_s \wedge y_s^{\text{у.э}} \supset \delta_s \\ \delta_s \wedge y_s^{\text{у.э}} \wedge R_s \supset \lambda_s \\ R_s \wedge \lambda_s \wedge \delta_s \supset y_s^{\text{у.э}} \end{cases}$$

Таким образом, для данного концепта построена Хорновская модель вида

$$\begin{cases} x_1^1 \wedge \dots \wedge x_{n_1}^1 \supset \text{Y}^1 \\ \dots \dots \dots \dots \dots \dots \dots \dots \\ x_1^N \wedge \dots \wedge x_{n_1}^N \supset \text{Y}^{\text{N}} \end{cases}$$

На которой необходимо решать задачи об истинности формул вида

$$x_1^0 \wedge \dots \wedge x_{n_0}^0 \supset \text{Y}^0$$

На приведенном концепте мы можем решать различные задачи:

1) При строительстве мы готовы выделить N см для определенного слоя материалов, в силу каких-то конструктивных соображений. Какой коэффициент λ должен иметь материал для наших целей?

2) Также в одном из слоев может быть произведена замена материала, например, в процессе перестройки. Как это повлияет на теплопроводность?

3) Имеется готовая конструкция стены с приведенными параметрами. Необходимо проверить, удовлетворяет ли она нормативным требованиям? и так далее.

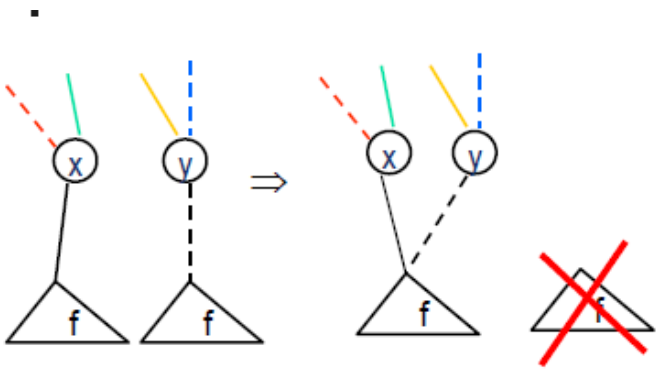
Глава 6. BDD - каноническая форма представления формул логики высказываний

6.1. Определение BDD

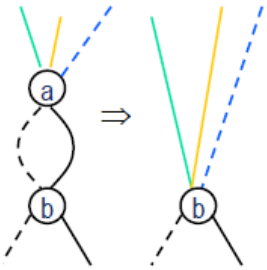
Булеву функцию $f(x_1, x_2, \ldots, x_n)$ можно представить в виде направленного ациклического графа, состоящего из нескольких внутренних узлов решений и двух терминальных узлов, называемых 0-терминальный узел и 1-терминальный узел. Каждый внутренний узел решений на уровне i помечен булевой переменной x_i и имеет по два потомка, называемых младшим потомком и старшим потомком. Переход от внутренного узла x_i к младшему или старшему потомку выполняется в зависимости от значения переменной x_i (0 или 1 соответственно). Для заданных значений $x_1, x_2, \ldots, x_n$ путь от корневого узла до 1-терминального узла соответствуют тому, что на этих входных значениях булева функция $f(x_1, x_2, \ldots, x_n)$ принимает значение 1.

BDD называется *упорядоченной*, если переменные появляются в одном и том же порядке во всех путях от корневого узла графа до одной из терминальных вершин. При этом, некоторые переменные в путях могут отсутствовать, если над графом ранее была произведена операция сокращения. Основная идея построения *сокращенной* BDD состоит в устранении из построенного варианта всей избыточной информации о функции. Это достигается применением следующих двух правил сокращения (в любом порядке, до тех пор, пока ни одно из них окажется не применимым).

П1. Слияние любых изоморфных подграфов.



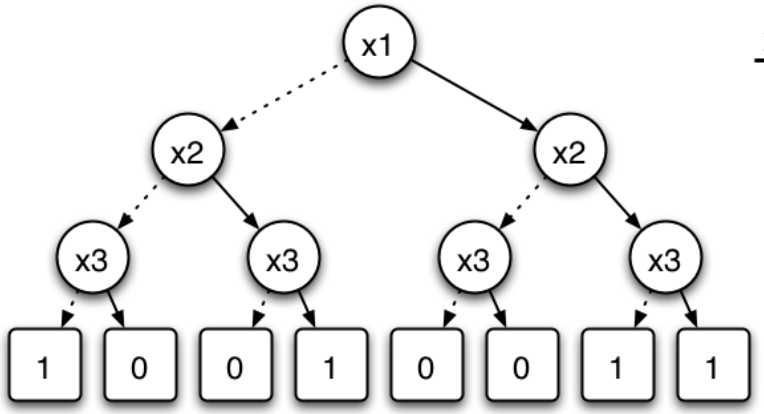
П2. Удаление всех узлов с изоморфными потомками.



Эта процедура получила название Reduce.

В большинстве случаев под BDD понимают именно сокращенную упорядоченную бинарную диаграмму решений. Преимущество сокращенной упорядоченной BDD в том, что она является уникальной для конкретной функции и заданного порядка переменных. Т.е. BDD являются каноническим представлением булевых функций, наподобие с СКНФ, СДНФ, полиномам Жегалкина. Это свойство делает BDD полезной для проверки функциональной эквивалентности.

Пример.

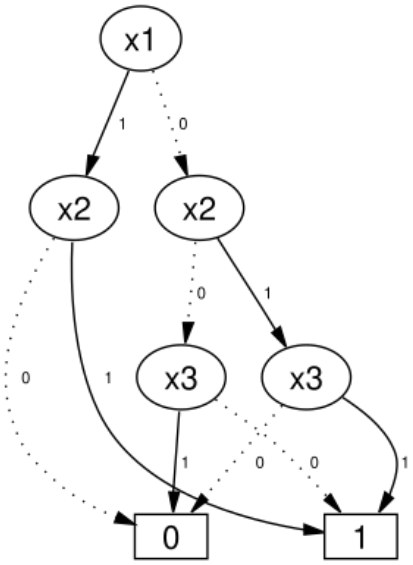


x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

На рисунке изображено бинарное дерево принятия решений (без применения правил сокращения), соответствующее приведенной на этом же рисунке таблице истинности для булевой функции $f(x_1, x_2, x_3)$. Для заданных входных значений x_1, x_2, x_3 можно определить значение булевой функции, двигаясь по дереву от корневого узла дерева к терминальным узлам, выбирая направление перехода из узла x_i в зависимости от входного значения x_i. Пунктирными линиями на рисунке изображены переходы к младшему потомку, а непрерывными линиями изображены переходы к старшему потомку. Например, если заданы входные значения ($x_1 = 0, x_2 = 1, x_3 = 1$), то из корневого узла x_1 необходимо перейти по пунктирной линии влево (так как значение x_1 равно 0), после этого необходимо перейти по непрерывным линиям вправо (так как

значения x_2 и x_3 равны 1). В результате мы окажемся в 1-терминальном узле, то есть значение $f(x_1 = 0, x_2 = 1, x_3 = 1)$ равно 1.

Бинарное дерево принятия решений на рисунке можно преобразовать в бинарную диаграмму решений путём применения двух правил сокращения. Результирующая BDD изображена на рисунке ниже.

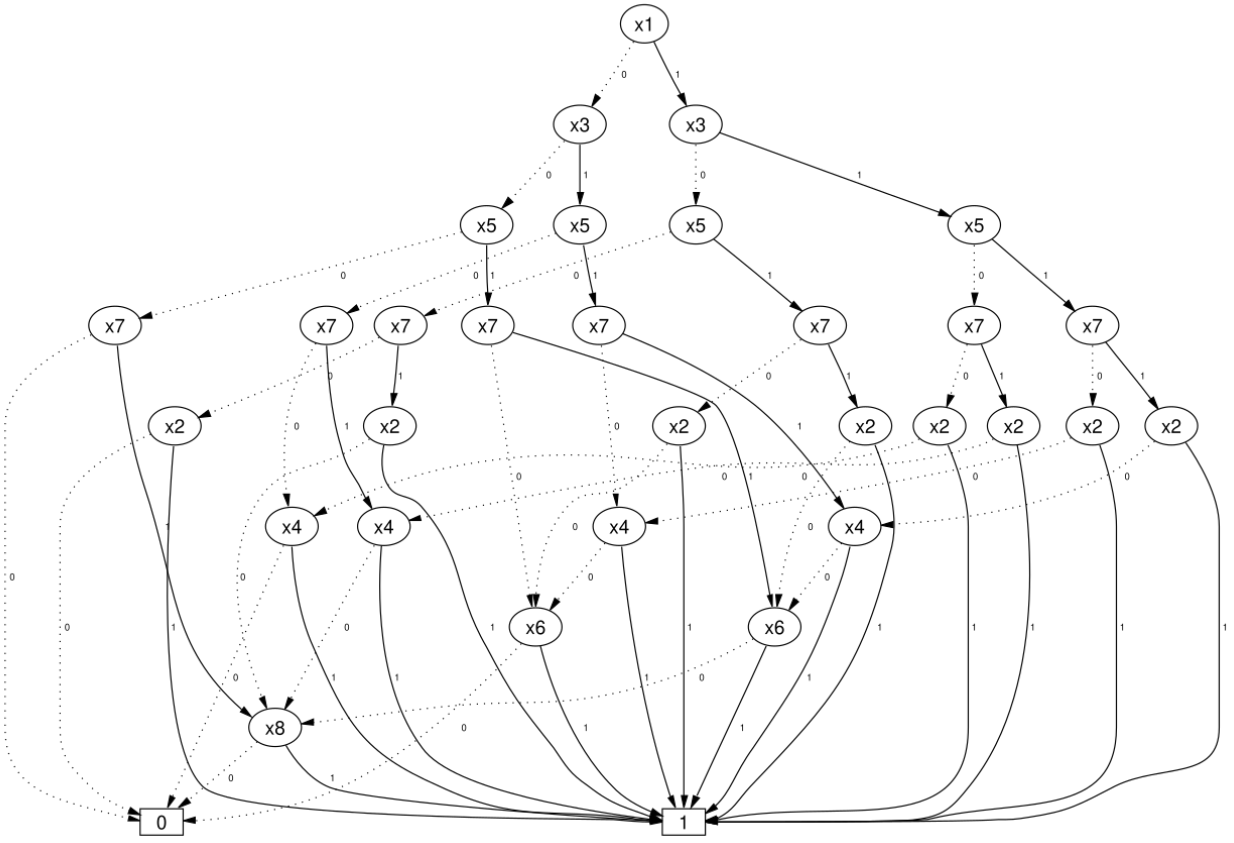


Основной идеей для создания такой структуры данных послужило разложение Шеннона. Любую булеву функцию по одной из входных переменных можно разделить на две подфункции, называемых положительным и отрицательным дополнением, из которых по принципу *if-then-else* выбирается только одна подфункция в зависимости от значения входной переменной (по которой выполнялось разложение Шеннона). Представляя каждую такую подфункцию в виде поддерева и продолжая разложение по оставшимся входным переменным, можно получить дерево принятия решений, сокращение которого даст *бинарную диаграмму решений*. Информация об использовании бинарных диаграмм решений или бинарных ветвящихся программ была впервые опубликована в 1959 году в техническом журнале «Bell Systems Technical Journal». Потенциал для создания эффективных алгоритмов, основанных на использовании этой структуры данных, исследовал Randal Bryant из университета Карнеги — Меллон: его подход заключался в использовании фиксированного порядка переменных (для однозначности канонического представления каждой булевой функции) и повторного использования общих подграфов (для компактности). Применение этих двух ключевых концепций позволяет повысить эффективность структур данных и алгоритмов для представления множеств и отношений между ними.Дональд Кнут в своей видеолекции Fun With Binary Decision Diagrams (BDDs) назвал бинарные диаграммы решений «*одной из действительно фундаментальных структур данных, которые появились за последние двадцать пять лет*» и отметил, что публикация Randal Bryant от 1986 года, осветившая использование бинарных диаграмм решений для манипуляции над булевыми функциями, являлась некоторое время наиболее цитируемой публикацией в компьютерной науке.

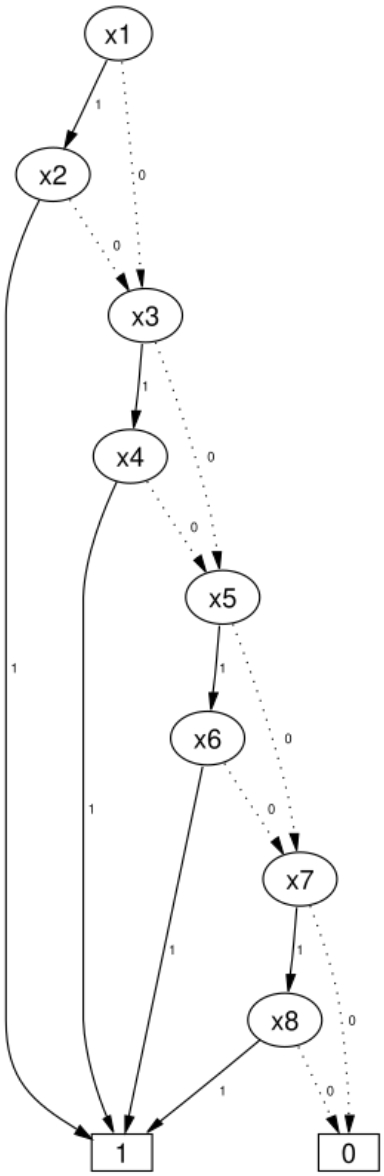
Порядок переменных.

Размер BDD определяется как самой булевой функцией, так и выбором порядка входных переменных. При составлении графа для булевой функции $f(x_1, \ldots, x_n)$ количество узлов в лучшем случае может линейно зависеть от n, а в худшем случае зависимость может быть экспоненциальной при неудачном выборе порядка входных переменных.

Например, дана булева функция $f(x_1, \ldots, x_{2n}) = x_1 x_2 + x_3 x_4 + \cdots + x_{2n-1} x_{2n}$. Если использовать порядок переменных $x_1 < x_3 < \cdots < x_{2n-1} < x_2 < x_4 < \cdots < x_{2n}$, то понадобится 2^{n+1} узлов для представления функции в виде BDD. Иллюстрирующая BDD для функции от 8 переменных изображена на рисунке.



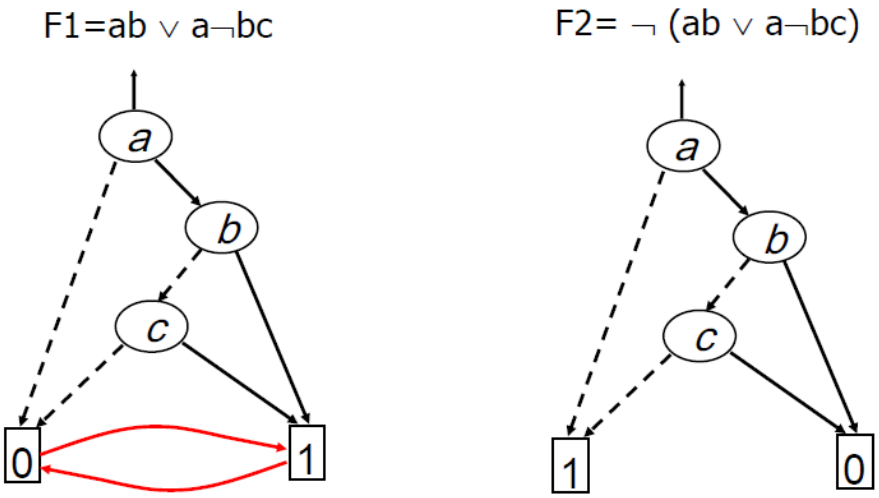
А если использовать порядок $x_1 < x_2 < x_3 < x_4 < \cdots < x_{2n-1} < x_{2n}$, то можно получить BDD той же функции из $2n+2$ узлов. Иллюстрирующая БДР для функции от 8 переменных изображена на следующем рисунке.



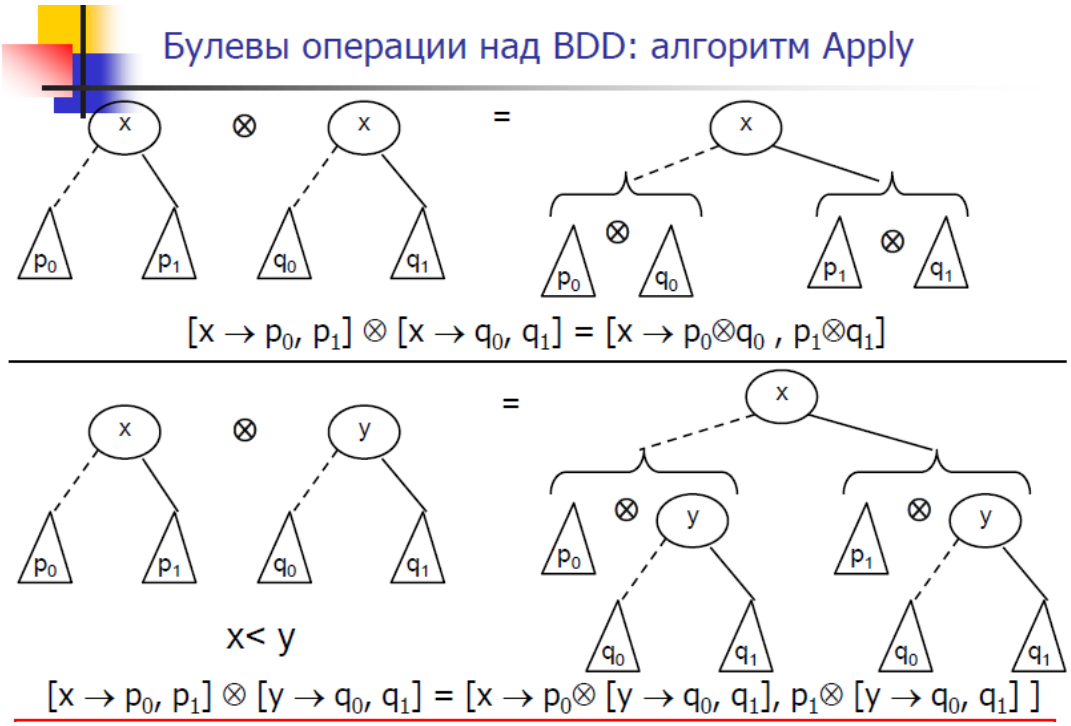
Выбор порядка переменных является критически важным при использовании таких структур данных на практике. Проблема нахождения лучшего порядка переменных является NP-полной задачей. Более того, NP-полной является даже задача нахождения субоптимального порядка переменных, при котором для любой константы $c > 1$ размер BDD не более чем в c раз больше оптимального. Однако существуют эффективные эвристические методы для решения этой проблемы.Кроме того, существуют такие функции, для которых размер графа всегда растет экспоненциально с ростом числа переменных, независимо от порядка переменных.

6.2. Логические операции над бинарными диаграммами решений

1. Отрицание.



2. Бинарные булевы операции (конъюнкция, дизъюнкция, импликация, эквивалентность, (исключающее или) реализуются по одинаковой схеме:

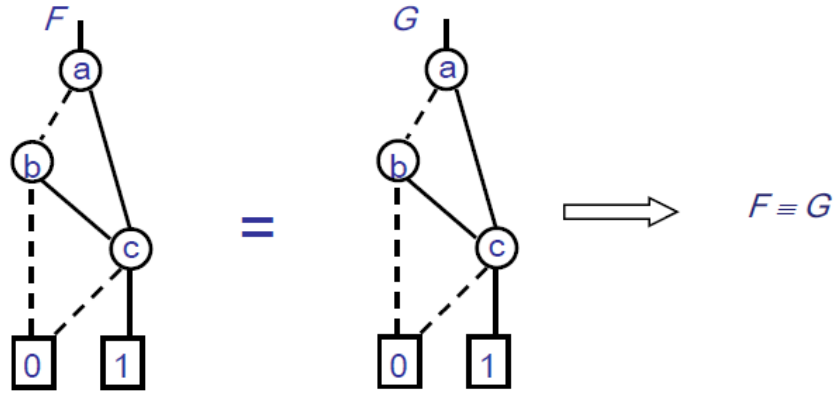


Мы видим, что булевы операции над булевыми функциями, представленными в виде BDD, реализуются просто.

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

3. Проверка эквивалентности булевых функций

$F = a\neg bc \vee abc \vee ab\neg c$ $\quad G = ac \vee bc$ $\quad F \equiv G\ (?)$

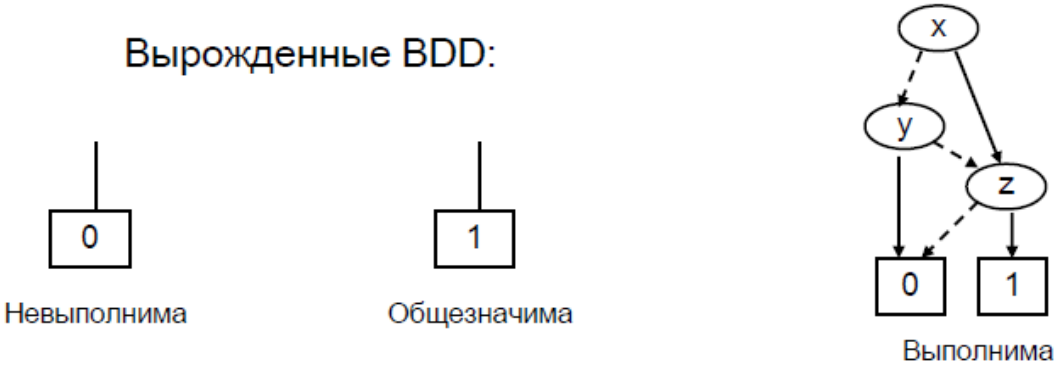


Поскольку BDD – каноническое представление, проверка равнозначности двух функций, представленных в BDD, сводится к проверке совпадения направленных графов

6.3. Проверка выполнимости (общезначимости) булевых функций

Теорема Кука. Проблема выполнимость булевых функций NP-полна**.**

Но проблема выполнимости, невыполнимости и общезначимости для BDD тривиальны - достаточно проверить, не является ли BDD вырожденной (значения 0 или 1). Поэтому проблема построения BDD для булевых формул является NP- полной.



Базовые логические операции (конъюнкция, дизъюнкция, отрицание и т.д.) могут быть выполнены непосредственно над BDD с помощью алгоритмов, выполняющих манипуляции над графами за полиномиальное время. Однако повторение этих операций множество раз, например, при формировании конъюнкций или дизъюнкций набора BDD, могут привести к экспоненциально большой BDD в худшем случае. Это происходит из-

за того, что результатом любых предшествующих операций над двумя BDD в общем случае может быть BDD с размером, пропорциональным произведению предшествующих размеров, поэтому для нескольких BDD размер может увеличиваться экспоненциально.

Мы видели, что сложность получающихся BDD зависит от порядка переменных. Разные порядки могут приводить к BDD разной сложности (от линейной до экспоненциальной). Проблема нахождения оптимального порядка переменных NP-трудна. Но разработаны многочисленные эвристики, обеспечивающие приемлемую сложность строящихся BDD. В настоящее время специалисты сходятся во мнении, что почти для всех практически важных функций можно построить BDD линейной сложности.

6.4. Калькулятор для построения двоичных решающих диаграмм

Построение BDD по формуле алгебры-логики в задачах не очень большой размеренности может быть проделано с помощью программы *калькулятор BDD*.

Калькулятор доступен по ссылке: https://programforyou.ru/tests/CalculatorBDD/CalculatorBDD

Что умеет калькулятор:

построить BDD по заданной формуле алгебры логики или булевому вектору

- задавать и изменять порядок переменных
- вывести пошаговое решение
- выводить таблицы истинности и некоторые дополнительные данные
- позволяет перемещать строительные узлы в построенной BDD.

Пример работы калькулятора.

Калькулятор для построения BDD поформуле алгебры логики

Выражение: x1 + x2 ^ x3 ⊕¬(x1→x2)→(x1∧x2≡x3)∨x3 Построить

∨	∧	¬	⊕	→	≡	↓	|
0	1	a	b	c	x	y	z
(	)	x1	x2	x3	x4	x5	x6

Порядок переменных: x1 x2 x3

Введённое выражение: x1 ∨ x2 ⊕ x3 ⊕¬(x1→x2)→(x1∧x2≡x3)∨x3

Распаршенное выражение: ((((x1 ∨ x2) ⊕ x3) ⊕ ¬(x1 → x2)) → (((x1 ∧ x2) ≡ x3) ∨ x3))

Вектор функции: 11111101

Таблица истинности:

x1	x2	x3	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Разбиение по переменной x1:

TRUE: (x2 → x3)
FALSE: 1

Разбиение по переменной x2:

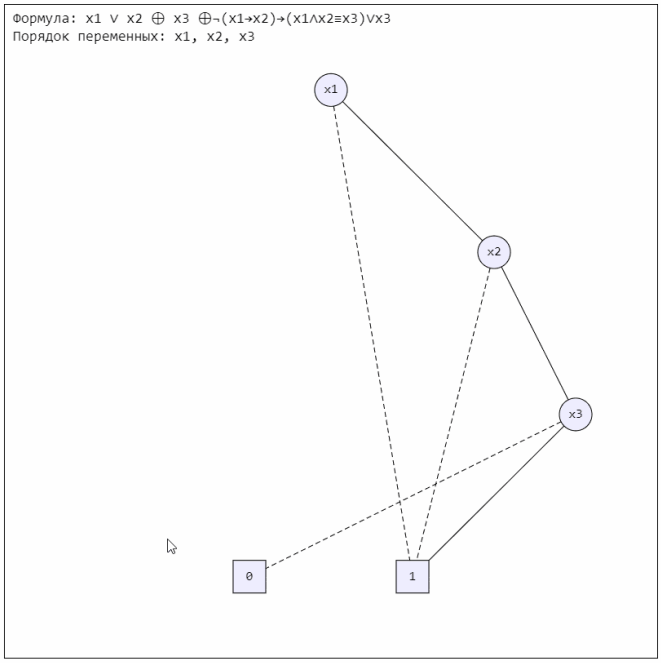
TRUE: (x1 → x3)
FALSE: 1

Разбиение по переменной x3:

TRUE: 1
FALSE: (x2 → ((x1 ∧ x2) ≡ 0))

Построение ROBDD:

Apply(x1 ∨ x2 ⊕ x3 ⊕¬(x1→x2)→(x1∧x2≡x3)∨x3) = Reduce(Compose(x1, Apply((x2 → x3)), [1]))
Apply((x2 → x3)) = Reduce(Compose(x2, Apply(x3), [1]))
Apply(x3) = Reduce(Compose(x3, [1], [0]))



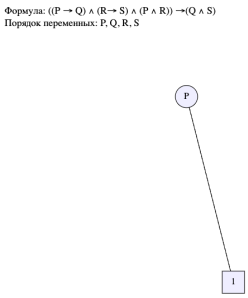
Используем калькулятор для решения задачи «проверка логического следствия».

Задача. Проверить, справедливо ли логическое следствие: $P \to Q, R \to S, P\&R \models Q\&S$.

Решение. Выясним, будет ли формула

$((P \to Q) \& (R \to S) \& (P \& R)) \to (Q \& S)$ тавтологией.

Для этого попытаемся построить BDD для данной формулы. Выберем следующий порядок переменных: P Q R S. Калькулятор выдаст следующую BDD



Видно, что формула является тавтологией, а следовательно, данное логическое следствие справедливо.

6.5. Приложения BDD

Использование BDD оказалось очень эффективным в ряде важных приложений.

6.5.1. Символьное представление множеств

Пусть S – конечное множество из m элементов.

Закодируем все элементы S наборами двоичных переменных x1, …, xk.

Пример: S - из 8 элементов: {a0, a1, …, a7}.

Кодируем (k=3): a0 ⇔ 000, a1 ⇔ 001, … , a7 ⇔⊆ 111.

Пусть P ⊆ S, P= {a0, a1, a2, a6}. Соответствующее множество двоичных кодов { 000, 001, 010, 110}. Построим булеву функцию χ_P, которая равна 1 на наборах, кодирующих элементы, принадлежащие множеству P.

χ_P называется характеристической функцией множества P.

Пусть v = { x_1, x_2, x_3 }– разряды кодировки. Тогда булева функция:

x1	x2	x3	S	χ_P
0	0	0	a0	1
0	0	1	a1	1
0	1	0	a2	1
0	1	1	a_3	0
1	0	0	a_4	0
1	0	1	a_5	0
1	1	0	a_6	1
1	1	1	a_7	0

χ P = $\neg x1 \neg x2 \neg x3 \vee \neg x1 \neg x2\, x3 \vee \neg x1 x2 \neg x3 \vee x1 x2 \neg x3$ задает P = {000, 001, 010, 110}. Это СДНФ функции.

Таким образом любое подмножество конечного множества можно задавать булевой формулой, представляющей характеристическую функцию.
Иногда удобно писать $\chi P(v)$, если χP представлена как функция от переменных v=(x1, x2, x3, ...), чтобы зафиксировать переменные кодировки. Булевы функции как и формулы логики высказываний могут быть представлены множеством способов. Такие представления конечных множеств называются символьными представлениями.

Символьные вычисления.

Определение. *Символьными вычислениями* называются вычисления, при которых операции над множествами представляются операциями над характеристическими двоичными функциями, представляющими эти множества.
Сведение проблем, связанных с конечными множествами, к операциям с БФ:
Включение множеств:

$P \subseteq Q$ тогда и только тогда, когда функция $\chi P \Rightarrow \chi Q$ общезначима.

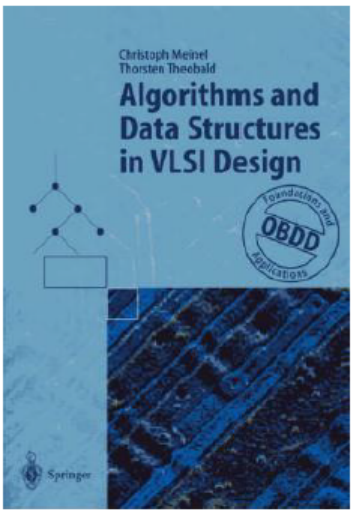
$P \subset Q$ тогда и только тогда, когда функция $(\chi P \rightarrow \chi Q) \wedge \neg(\chi Q \rightarrow \chi P)$ общезначима.
Равенство множеств:

$P = Q$ тогда и только тогда, когда функции χP и χQ равносильны.
Проверить равносильность БФ можно по каноническому представлению функций, например, по ТИ, СДНФ, СКНФ, по полиному Жегалкина. И, конечно, по представлениям функций в виде BDD.
Важно заметить, что при символьном представлении для решения не нужно перебирать элементы множеств. Все проблемы решаются выполнением операций над характеристическими двоичными функциями множеств.

Любое подмножество множества из 10**20 состояний может быть представлено булевой функцией от 70 переменных (2**10~10**3, 10**20 ~ 2**70).
Операции над такими характеристическими булевыми функциями, представленными в виде BDD, выполняются менее, чем за секунду.
При верификации систем (очень важный класс алгоритмов) с числом состояний 10**20 нужно:
- множество начальных состояний (1 функция от 70 переменных)
- множество переходов (1 функция от 140 переменных)
- для каждого атомарного предиката – множество состояний, в которых этот предикат истинен (1 функция от 70 переменных)
- для каждой подформулы проверяемой формулы необходимо определить множество состояний, в которых эта формула истинна (1 функция от 70 переменных).
Реальные практически важные системы, которые необходимо верифицировать, содержат обычно > 10**100 состояний. Тогда для представления подмножеств состояний нужны БФ от ~300 двоичных переменных, а для представления отношений нужны БФ от ~600 переменных, что при представлении БФ в форме BDD вполне выполнимо.

6.5.2.

BDD и синтез СБИС



- Подзаголовок книги:

OBDD – Foundations and Applications

Из введения:
Для сверхбольших схем (СБИС) проблема представления и преобразования их функционального поведения становится почти неразрешимой. Проблема была разрешена в результате установления плодотворной связи с одним из базовых результатов теоретической вычислительной науки в области построения структур данных и эффективных алгоритмов их обработки. Этот результат - использование OBDD - привел к драматическому улучшению производительности и прорыву во многих CAD проектах во всем мире

BDD являются основной моделью для представления и обработки информации в современной технологии разработки СБИС

Рекомендуемая литература по BDD.

- D. E. Knuth, «The Art of Computer Programming Volume 4, Fascicle 1: Bitwise tricks & techniques; Binary Decision Diagrams» (Addison-Wesley Professional, March 27, 2009) viii+260pp, ISBN 0-321-58050-8. Draft of Fascicle 1b available for download.
- H. R. Andersen «An Introduction to Binary Decision Diagrams», Lecture Notes, 1999, IT University of Copenhagen.
- Ch. Meinel, T. Theobald, «Algorithms and Data Structures in VLSI-Design: OBDD — Foundations and Applications», Springer-Verlag, Berlin, Heidelberg, New York, 1998. Complete textbook available for download.
- *Rüdiger Ebendt; Görschwin Fey; Rolf Drechsler.* Advanced BDD optimization (англ.). — Springer[англ.], 2005. — ISBN 978-0-387-25453-1.
- *Bernd Becker; Rolf Drechsler.* Binary Decision Diagrams: Theory and Implementation (англ.). — Springer[англ.], 1998. — ISBN 978-1-4419-5047-5.
- O. A. Finko, K. S. Meretukov, Systems of Boolean functions: numerical decomposition in Z_m ring, OP&PM Surveys on Applied and Industrial Mathematics. Proceedings II International Baltic Symposium on Applied and Industrial Mathematics (Svetlogorsk, June 12 – 18, 2016), 23, TVP, Moscow, 2016, 167-168.

Ссылки на библиотеки, позволяющие работать с BDD с большим количеством переменных.

- ABCD: The ABCD package by Armin Biere, Johannes Kepler Universität, Linz.
- CMU BDD, BDD package, Carnegie Mellon University, Pittsburgh
- CUDD: BDD package, University of Colorado, Boulder
- Installing CUDD in Windows/Visual Studio environments.
- Biddy: multi-platform academic BDD package, University of Maribor, Slovenia
- JavaBDD, a Java port of BuDDy that also interfaces to CUDD, CAL, and JDD
- The Berkeley CAL package which does breadth-first manipulation
- A. Costa BFunc, includes a BDD boolean logic simplifier supporting up to 32 i

Литература

Основная

1. Клини С. Математическая логика. М.:Мир, 1973, 480 с.
2. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.:Мир, 1983. 360 с.
3. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. Москва, "Физико-математическая литература", 1995 г., 250 с.
4. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. Санкт-Петербург, «БХВ-Петербург», 2010. 552с.
5. Захаров В.А. Лекции по математической логике и теории алгоритмов. https://mk.cs.msu.ru/index.php/Участник:ZakharovVA
6. Герасимов А.С. Лекции по математической логике. Секвенциальное исчисление высказываний. http://gas-teach.narod.ru

Дополнительная

1. Мендельсон Э. Введение в математическую логику. М.:Наука, 1984. 319 с.
2. Верещагин Н.К., Шень А. Языки и исчисления. 2004.
3. Успенский В.А., Верещагин Н.К., Плиско В.Е. Вводный курс математической логики. 2004. 128 с.
4. Колмогоров А.Н., Драгалин А.Г. Математическая логика. Серия "Классический университетский учебник". Изд.3, 2006, 240 с.
5. Непейвода Н. Н. Прикладная логика. Новосибирск. 2000 г. 529 с.
6. Ковальский Р. Логика в решении проблем. М.: Наука, 1990. 277 с.
7. Логический подход к искусственному интеллекту (от модальной логики к логике баз данных). М.:Мир, 1998. 495 с.