Формальные методы в программной инженерии

Проверка моделей

Символьные представления

Введение

- Идея символьной проверки на модели
  - Представить множество состояний системы (формулу) компактным образом
  - Иметь возможность вычислять значения операций на множествах-формулах
    - дополнение-отрицание
    - объединение-дизъюнкция
    - пересечение-конъюнкция
    - предусловие (модальности)
      - обратное отношение достижимости
  - Вычислять семантику формулы изнутри
    - начать с семантики атомарных высказываний
    - использовать соответствующие операции на множествах

Введение

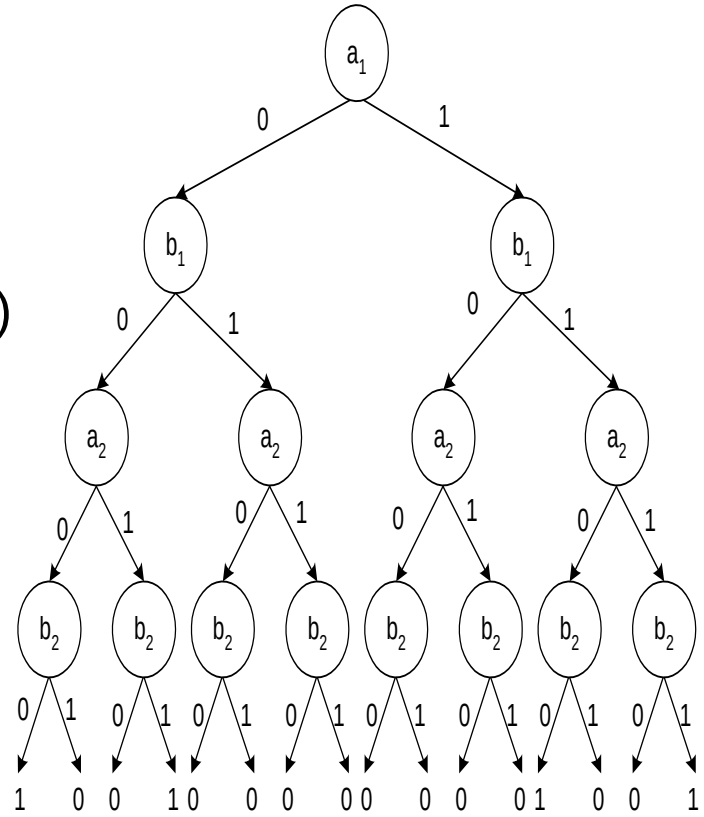
- Символьные представления
  - Булевские модели
    - Бинарные Разрешающие Диаграммы BDD (Binary Decision Diagram)
      - Множество состояний, переход — булевская формула
      - Закодировать булевскую формулу как BDD
  - Целочисленные модели
    - Формулы арифметики Пресбургера
      - Множество состояний, переход — формула арифметики Пресбургера

BDD. Представление булевых функций

- Булевы функции определяются на значениях 0 и 1
  - 0 — логическая константа False
  - 1 — логическая константа True.
- Упорядоченные двоичные разрешающие диаграммы
  - OBDD — Ordered Binary Decision Diagrams
  - каноническая форма представления булевых функций
  - Более компактны, чем традиционные нормальные формы
    - дизъюнктивная нормальная форма
    - конъюнктивная нормальная форма
  - Эффективность операций
- Широкое применение в автоматическом проектировании
  - символьное имитационное моделирование
  - верификация комбинационных логических схем
  - верификация параллельных систем с конечным числом состояний

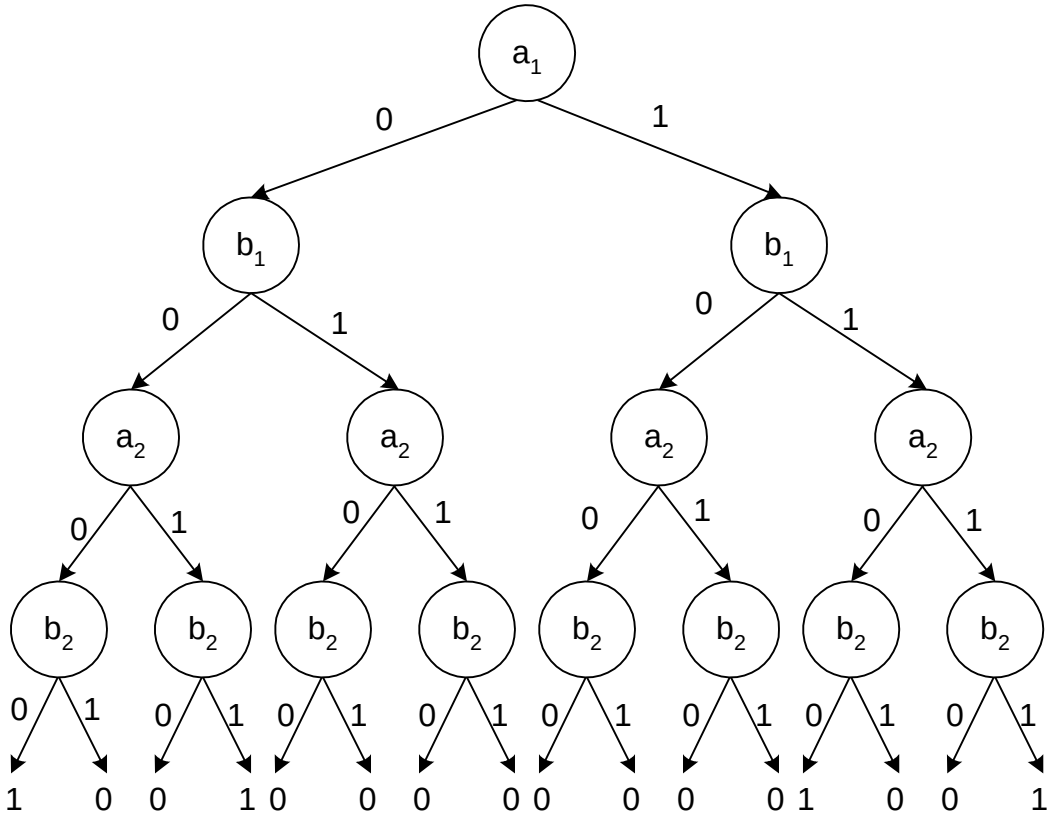
BDD. Представление булевых функций

- Двоичное разрешающее дерево
  - корневое ориентированное дерево
  - вершины
    - нетерминальная вершина v
      - помечена переменной var(v)
      - последователь low(v)
        - var(v) = 0
      - последователь high(v)
        - var(v) = 1
    - терминальная вершина v (лист)
      - помечена value(v)
        - 0 или 1



- Двоичное разрешающее дерево для двухбитового компаратора
  - $f(a_1, a_2, b_1, b_2) = (a_1 \leftrightarrow b_1) \wedge (a_2 \leftrightarrow b_2)$.

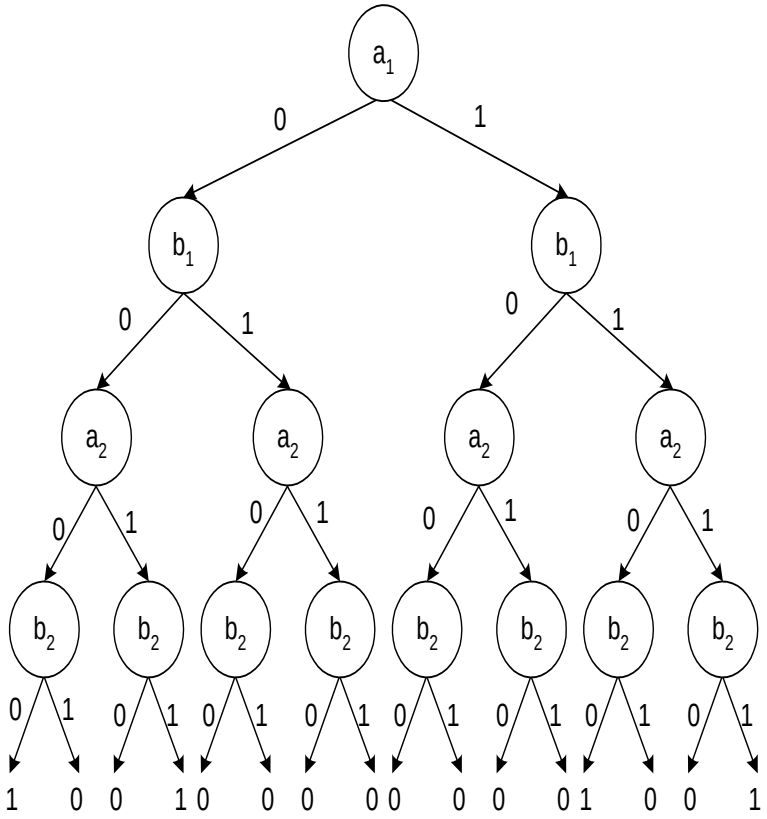
1. BDD. Представление булевых функций



Двоичное разрешающее дерево для двухбитового компаратора

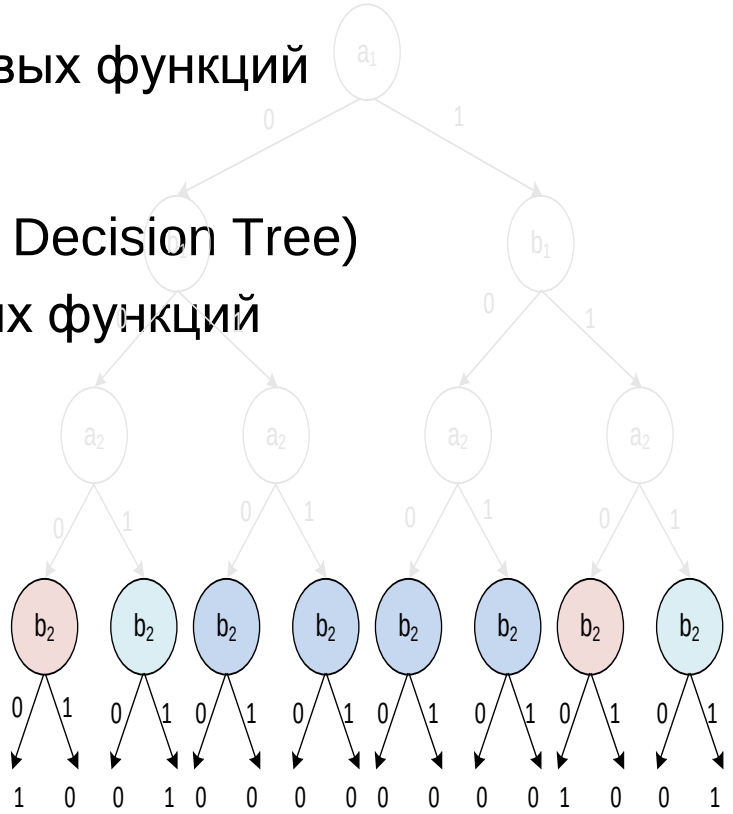
1. BDD. Представление булевых функций

- Значение формулы при заданном наборе значений переменных
  - спуск по дереву из корня до листа
    - если var(v) = 0
      - следующая вершина на пути — low(x)
    - если var(v) = 1
      - следующая вершина на пути — high(x)
  - значение функции
    - пометка листа
- Оценка ($a_1 \leftarrow 1$, $a_2 \leftarrow 0$, $b_1 \leftarrow 1$, $b_2 \leftarrow 1$)
  - приводит к листу, помеченному 0
    - при таком означивании формула ложна



1. BDD. Представление булевых функций

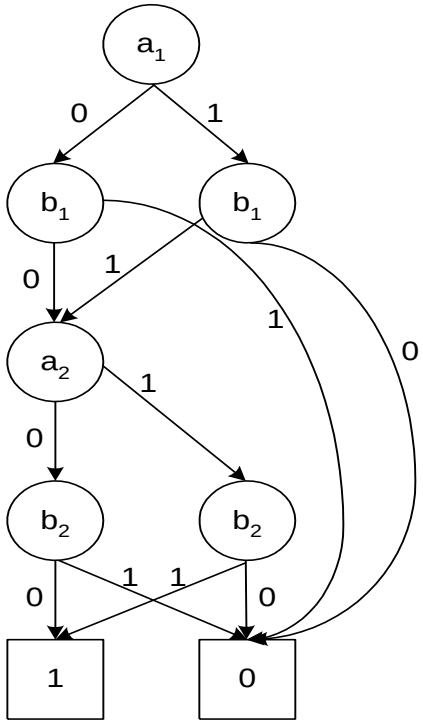
- Двоичные разрешающие деревья (Binary Decision Tree)
  - не компактное представление булевых функций
  - размер таблицы истинности
- Избыточность BDT
  - Дерево компаратора содержит
    - восемь поддеревьев с корнем b_2,
      - только три попарно различны
  - Более компактное представление
    - слияние изоморфных поддеревьев
- Двоичная разрешающая диаграмма BDD
  - корневой ориентированный ациклический граф
    - терминальные и нетерминальные вершины
    - результат слияния изоморфных поддеревьев BDT



1. BDD. Представление булевых функций

- Двоичная разрешающая диаграмма
  - нетерминальная вершина v
    - помечена переменной var(v)
    - вершины-последователи low(v) и high(v)
  - терминальная вершина
    - помечена value(v)$\in$ {0, 1}
- Двоичная разрешающая диаграмма B
  - с корнем в вершине v
  - определяет булеву функцию $f_v(x_1, \ldots, x_n)$:

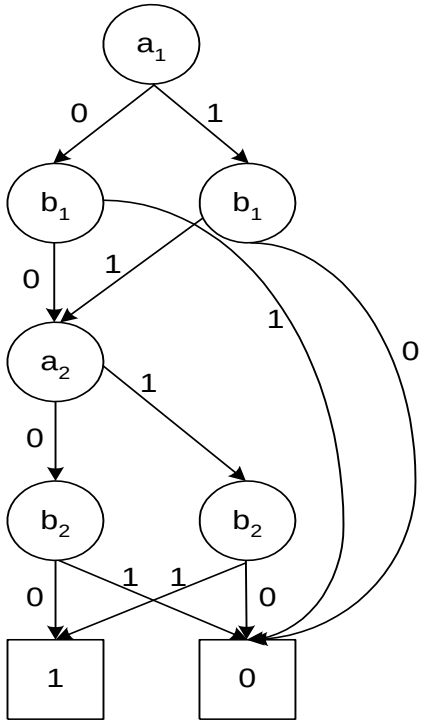
1. если v — терминальная вершина, то
   a) если value(v) = 1, то $f_v(x_1, \ldots, x_n) = 1$
   b) если value(v) = 0, то $f_v(x_1, \ldots, x_n) = 0$
2. если v — нетерминальная вершина и var(v) = x_i
   - $f_v(x_1, \ldots, x_n) = (\neg x_i \wedge f_{low(v)}(x_1, \ldots, x_n)) \vee (x_i \wedge f_{high(v)}(x_1, \ldots, x_n))$.



1. BDD. Представление булевых функций

- Каноническое представление булевых функций
  - обеспечивает однозначность
  - две булевы функции логически эквивалентны ⇔ они имеют изоморфные представления.
    - проверка эквивалентности двух формул
    - проверка выполнимости формулы
- Две двоичные разрешающие диаграммы изоморфны
  - если есть взаимно однозначное отображение h:

    $T \cup NT \rightarrow T \cup NT$
    - $\forall v \in T$:
      - $h(v) \in T$
        - value(v) = value(h(v))
    - $\forall v \in NT$:
      - $h(v) \in NT$
        - var(v) = var(h(v))
        - h(low(v)) = low(h(v)) и h(high(v)) = high(h(v)).



1. BDD. Представление булевых функций

- Канонические представления булевых функций
  - Ограничения на структуру BDD
    1. По каждому пути из корня в лист переменные следуют в одном и том же порядке
       - линейный порядок < на множестве переменных
         - для каждой нетерминальной вершины u с нетерминальным последователем v верно

           var(u) < var(v)
    2. в диаграмме нет изоморфных поддеревьев и избыточных вершин
       - последовательно применить три правила преобразования, сохраняющих функцию, представленную диаграммой
       - `Reduce`

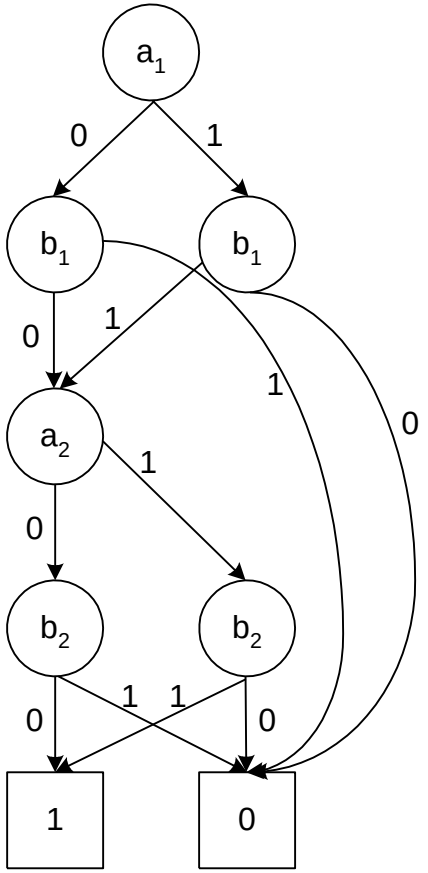
1. BDD. Представление булевых функций

Правила преобразования

1. Удаление повторных вхождений терминалов
   - Остаётся два листа с пометками 0 и 1
     - все дуги, которые ранее вели в удалённые листы, направляются в соответствующий лист
2. Удаление повторных вхождений нетерминалов.
   - Если для нетерминальных вершин u и v верно
     - var(u) = var(v), low(u) = low(v) и high(u) = high(v)
     - одна из них удаляется
     - все ведущие в нее дуги направляются в оставшуюся вершину.
3. Удаление избыточных проверок
   - Если для нетерминальной вершины v верно
     - low(v) = high(v),
     - v удаляется
     - дуги, ведущие в v, направляются в low(v)

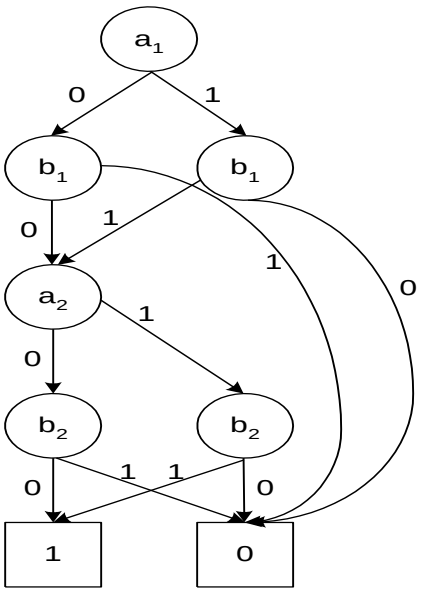
1. BDD. Представление булевых функций

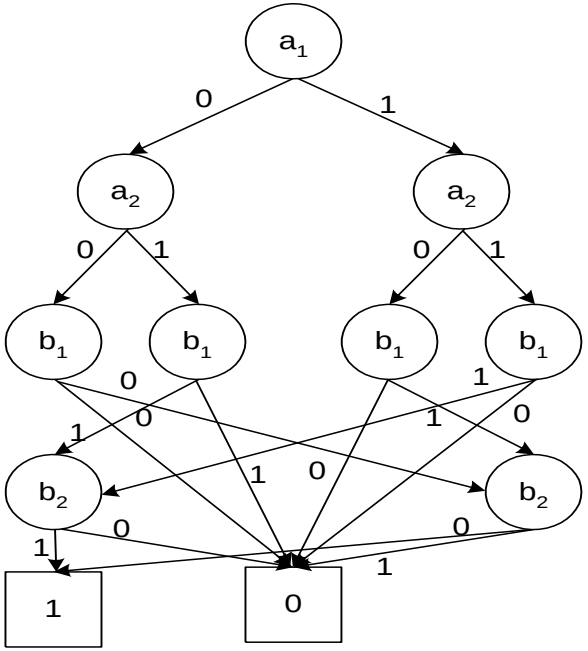
- Каноническая форма получается из любой BDD с заданным порядком переменных
- Правила преобразования применяются до тех пор, пока размер диаграммы не стабилизируется
  - от листьев к корню
  - временная сложность линейна относительно размеров BDD
- Упорядоченная двоичная разрешающая диаграмма (OBDD)
  - граф, полученный методом `Reduce`
- Упорядочение $a_1 < b_1 < a_2 < b_2$ для компаратора



1. BDD. Представление булевых функций

- Пусть OBDD — каноническая форма булевых функций
  - проверка эквивалентности формул
    - проверка изоморфизма между OBDD
  - выполнимость
    - проверки изоморфизма данной OBDD и вырожденной OBDD, состоящей из единственного листа, помеченного 0
- Размер OBDD сильно зависит от упорядочения переменных
- Сравним порядки для компаратора
  - $a_1 < a_2 < b_1 < b_2$
  - $a_1 < b_1 < a_2 < b_2$





1. BDD. Представление булевых функций

- n-битовый компаратор
  - $a_1 < b_1 < \ldots < a_n < b_n$
    - число вершин в OBDD: $3n + 2$
  - $a_1 < \ldots < a_n < b_1 < \ldots < b_n$
    - число вершин в OBDD: $3 \cdot 2^n - 1$
- Найти наилучший порядок расположения переменных — затратная задача
  - проверка оптимальности предложенного порядка
    - NP-полная задача
  - Существуют булевы функции с размером OBDD экспоненциальным относительно числа переменных независимо от порядка их расположения.
    - Функция, вычисляющая n-й бит произведения двоичных целых чисел

1. BDD. Представление булевых функций

Эвристики поиска хороших упорядочений переменных

- Если булева функция представлена логической схемой, то подходят эвристики, основанные на обходе графа схемы в глубину с возвратом
  - Интуитивные соображения
    - OBDD часто уменьшаются, если связанные между собой переменные становятся ближе друг к другу в порядке
      - Переменные схемы связаны, если они совместно определяют выходы схемы
    - Эти переменные должны быть тесно сгруппированы при упорядочении
      - Располагать переменные в том порядке, в котором они учитываются при осуществлении обхода графа схемы в глубину с возвратом

1. BDD. Представление булевых функций

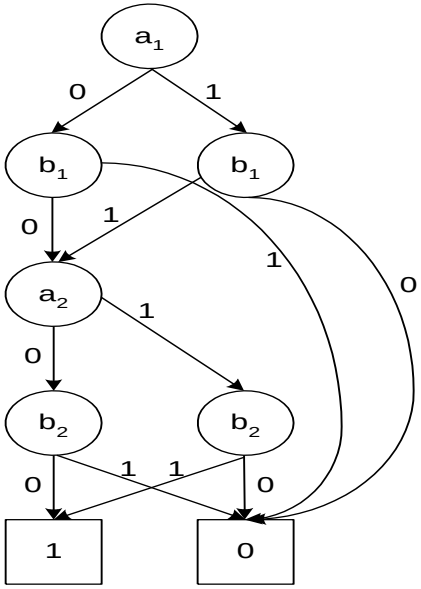
Эвристики поиска хороших упорядочений переменных

- Динамическое переупорядочение
  - никакая очевидная эвристика упорядочения не применима
  - Периодическое переупорядочением переменных
    - с целью сократить число используемых вершин.
  - Скорее экономится время, чем находится оптимальный порядок

1. BDD. Представление булевых функций

Операции для булевых функций, представленных OBDD

- Ограничение значения аргумента x_i булевой функции f заданной константой b
  - $f|_{x_i \leftarrow b}(x_1, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$.
  - OBDD для $f|_{x_i \leftarrow b}$ получено из OBDD для f
  - Для всякой вершины v, указывающей на такую w, что $var(w) = x_i$,
    - направить указатель
      - в low(w), если b = 0
      - в high(w), если b = 1
  - Граф может оказаться неканоническим
    - применить функцию `Reduce`
  - Линейная сложность



1. BDD. Представление булевых функций

Операции для булевых функций, представленных OBDD

- Разложение Шеннона $f = (\neg x \wedge f|_{x_i \leftarrow 0}) \vee (x \wedge f|_{x_i \leftarrow 1})$
- Алгоритм `Apply` для вычисления булевых операций
- Обозначения
  - * — двухместная логическая операция
  - f и f' — булевы функции
  - v и v' — корни OBDD для f и f';
  - x = var(v) и x' = var(v')
- Булево отрицание можно реализовать, применяя `Apply`
  - Проще поменять местами значения в листьях OBDD

1. BDD. Представление булевых функций

$$f * f' = (\neg x \wedge f|_{x_i \leftarrow 0} * f'|_{x_i \leftarrow 0}) \vee (x \wedge f|_{x_i \leftarrow 1} * f'|_{x_i \leftarrow 1})$$

`Apply`

- v и v' — терминальные
  - f * f' = value(v)*value(v')
- x = x‘
  - корень OBDD для f * f' — новая вершина w и var(w) = x
    - low(w) — корень OBDD для $f|_{x_i \leftarrow 0} * f'|_{x_i \leftarrow 0}$
    - high(w) — корень OBDD для $f|_{x_i \leftarrow 1} * f'|_{x_i \leftarrow 1}$
- x < x'
  - $f'|_{x_i \leftarrow 0} = f'|_{x_i \leftarrow 1} = f'$ (f' не зависит от x).
  - корень OBDD для f * f' — новая вершина w и var(w) = x
    - low(w) — корень OBDD для $f|_{x_i \leftarrow 0} * f'$
    - high(w) — корень OBDD для $f|_{x_i \leftarrow 1} * f'$
- x' < x
  - корень OBDD для f * f' — новая вершина w и var(w) = x'
    - low(w) — корень OBDD для $f * f'|_{x`_i \leftarrow 0}$
    - high(w) — корень OBDD для $f * f'|_{x`_i \leftarrow 1}$

1. BDD. Представление булевых функций

- Каждая задача
  - Две подзадачи
    - экспоненциальное разрастание вычислений
      - Динамическое программирование
        - полиномиальность алгоритма
  - Подзадача соответствует паре OBDD
    - подграфы исходных OBDD для f и f'.
      - определяется корнем
        - количество подграфов OBDD для f и f' ограничено размером OBDD
  - Количество подзадач ограничено произведением размеров OBDD для f и f'
- Хэш-таблица используется для хранения уже вычисленных подзадач
- Перед рекурсивным обращением проверить в кэше решённость подзадач
  - Если подзадача решена, использовать результат
  - Иначе выполнить рекурсивное обращение.
    - Канонизировать OBDD
    - Положить в кэш

1. BDD. Представление булевых функций

Обобщения OBDD

- Один многокорневой граф может представлять семейство булевых функций, имеющих общие подграфы
  - Для всех функций этого семейства порядок на переменных одинаков
  - В таком графе нет изоморфных подграфов и избыточных вершин
  - Здесь две функции семейства равны $\Leftrightarrow$

    у них один и тот же корень
    - Время проверки равенства двух функций — константа.
- Добавление пометок дугам графа для обозначения булевых отрицаний
  - Можно использовать один граф для представления формулы и ее отрицания
- Пакеты OBDD могут эффективно работать с графами, имеющими миллионы вершин

1. BDD. Представление булевых функций

OBDD как детерминированный конечный автомат

- Всякая n-местная булева функция характеризуется множеством наборов из $\{0,1\}^n$, на которых она обращается в 1.
  - Этот язык является конечным
    - регулярный
    - есть конечный автомат, допускающий такое множество
  - Этот автомат — каноническое представление булевой функции
- Логические операции над булевыми функциями выражаются посредством теоретико-множественных операций над языками, допускаемыми конечными автоматами.
  - Конъюнкция — пересечение множеств (языков, автоматов)
- Для вычисления теоретико-множественных операций над языками используются стандартные конструкции теории автоматов
- Стандартные операции над OBDD — аналоги таких конструкций

2. BDD. Представление моделей Крипке

- Всякое n-местное отношение Q на множестве {0,1} представляется как OBDD для его характеристической функции:
  - $f_Q(x_1, \ldots, x_n) = 1 \Leftrightarrow Q(x_1, \ldots, x_n)$.
- Пусть Q — n-местное отношение на конечной области D
  - пусть D содержит 2^m элементов для некоторого $m > 1$
- Биекция ϕ: $\{0, 1\}^m \rightarrow D$
  - кодирование элементов D
  - отображает всякий булев вектор длины m в элемент D
- Пусть $\bar{Q}$ — булево отношение местности $m \times n$
  - $\bar{Q}(\ddot{x}_1, \ldots, \ddot{x}_n) = Q(\phi(x_1), \ldots, \phi(x_n))$
    - $\ddot{x}_i$ — вектор, состоящий из m булевых переменных, код переменной x_i, принимающей значения из D
- Отношение Q $\Leftrightarrow$ OBDD для функции $f_{\bar{Q}}$ отношения $\bar{Q}$.
- Можно обобщить на отношения над разными областями $D_1, \ldots, D_n$
- Множества — одноместные отношения
  - метод можно использовать для представления множеств

2. BDD. Представление моделей Крипке

- Дана модель Крипке M = (S,R,L). Надо описать
  - множество S, отношение R и отображение L.
- Кодирование множества состояний S
  - пусть $|S| = 2^m$
  - есть функция ϕ: $\{0, 1\}^m \rightarrow S$
    - каждый набор значений булевых переменных кодирует некоторое состояние из S
  - характеристическая функция для S — константа 1
- Кодирование отношения переходов R
  - два множества булевых переменных
    - для представления исходного состояния
    - для представления результирующего состояния
  - Пусть отношение переходов R закодировано булевой функцией $\hat{R}(\ddot{x}, \ddot{x}`)$
    - R задается OBDD для характеристической функции $f_{\hat{R}}$

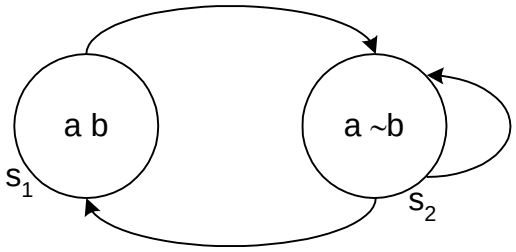
2. BDD. Представление моделей Крипке

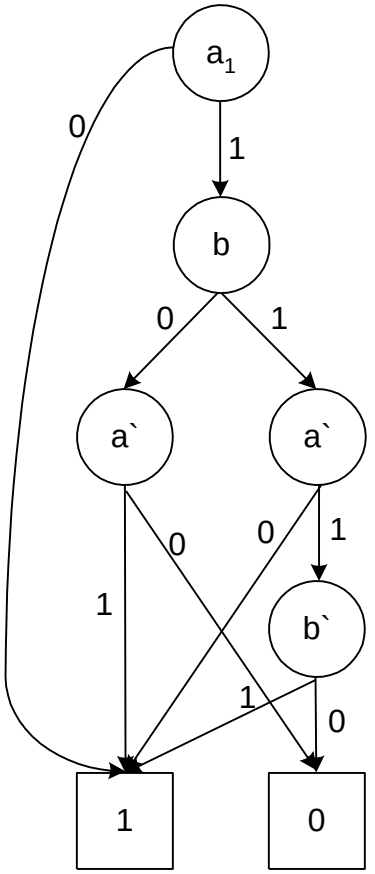
- Кодирование отображения L
  - $L: S \rightarrow 2^{AP}$
    - $L^{-1}: AP \rightarrow 2^{S}$
    - $L^{-1}(p) = \{s \mid p \in L(s)\} = L_p$
  - L_p можно представить в виде OBDD, используя ϕ
    - оценка каждого атомарного высказывания

2. BDD. Представление моделей Крипке

Пример OBDD для представления моделей Крипке

- Две переменных состояния a и b
  - две дополнительные переменные для кодирования состояний-последователей
- Переход из состояния s_1 в состояние s_2
  - $a \wedge b \wedge a` \wedge \neg b`$
- Формула для всего отношения переходов
  - $(a \wedge b \wedge a` \wedge \neg b`) \vee (a \wedge \neg b \wedge a` \wedge \neg b`) \vee (a \wedge \neg b \wedge a` \wedge b`)$
  - формула представляется как OBDD





2. BDD. Представление моделей Крипке

- Множество начальных состояний представляется, как и всякое другое множество.
- Для ограничений справедливости $F = \{P_1, \ldots, P_n\}$ каждое P_i представляется по отдельности.
- Использовать одни обозначения для множеств и отношений и их закодированных вариантов
  - $R \equiv \hat{R}$
- Обычно строить явное представление модели Крипке M, а затем кодировать его, нецелесообразно
  - модель может оказаться слишком большой, хоть и с компактным символьным представлением
- Как правило, OBDD строится непосредственно на основе описания системы на языке высокого уровня
  - Есть процедура трансляции, преобразующая системы в формулы
  - С использованием кодирования, аналогичная процедура подходит для построения OBBD из описания системы на языке высокого уровня.

3. Арифметика Пресбургера. Модели.

- T.Bultan, R.Gerber, W.Pugh (1998)
- Параллельная программа C = (V, I, E)
  - V — конечное множество переменных (данные и контроль)
  - I — начальное условие
  - E — конечное множество атомарных событий
- Состояние программы определяется значениями переменных
  - Область определения переменных— счётное множество
- События
  - пусковое условие
    - множество состояний, в которых возможно действие
  - действие
    - преобразование значений переменных программы

Program: Unbounded Buffer

Data Variables: p, c, q_1, q_2: positive integer

Control Variables: $pc : \{Idle, Send\}$

Initial Condition: $p = c = q_1 = q_2 = 0$

Events:

e_{IS}	**enabled:**	$pc = Idle$
	action:	$pc' = Send$
e_S	**enabled:**	$pc = Send$
	action:	$p' = p + 1 \wedge (q_1' = q_1 + 1 \vee q_2' = q_2 + 1)$
e_{SI}	**enabled:**	$pc = Send$
	action:	$pc' = Idle$
e_R	**enabled:**	$q_1 > 0 \vee q_2 > 0$
	action:	$(q_1 > 0 \wedge q_1' = q_1 - 1 \wedge c' = c + 1) \vee$
		$(q_2 > 0 \wedge q_2' = q_2 - 1 \wedge c' = c + 1)$

3. Арифметика Пресбургера. Модели.

- Модель программы C = (V; I; E) — бесконечная система переходов M = (S; I; X; L)
  - S — множество состояний
    - декартово произведение областей определения переменных $s \equiv \wedge v_i = x_i$
  - I — множество начальных состояний
  - $X \subseteq S \times S$ — отношение переходов (получено из E)
    - $(s,s`) \in X_e \Leftrightarrow$ s — состояние до e, s` — состояние после e
    - $X = \vee_{e \in E} X_e$ — интерливинговая модель
  - L: $S \times SF \rightarrow$ {True; False} — означивание формул состояния над программой SF

3. Арифметика Пресбургера. Формулы.

Формулы Пресбургера

- SF — формулы состояния программы C

  f ::= t ≤ t | (f) | f ∧ f | ¬f | ∃ `intvar` (f)

  t ::= (t) | t + t | `intvar` | `intcons`

  - `intcons`, `intvar` — целые константы и переменные
  - Можно выразить <, = ,∨, ∀, умножение на константу
- Задача проверки истинности формул арифметики Пресбургера разрешима
  - Для f∈SF и s∈S можно проверить s ⊨ f подстановкой значений переменных из s в свободные переменные f
  - В общем случае сложность определения истинности $2^{2^{2^{O(n)}}}$

3. Арифметика Пресбургера. Символическое представление.

- Компактность за счёт возможности описания больших множеств состояний с помощью арифметических неравенств.
  - $p - c \leq q_1 + q_2 \leq p \wedge p, c, q_1, q_2 \geq 0$
    - 4-х-мерный неограниченный многогранник
- События описаны в терминах целочисленных ограничений
  - Для каждого события *e* отношение переходов X_e можно представить формулой Пресбургера
  - Отношение переходов $X = \vee_{e \in E} X_e$
- Основная выгода представления — возможность сохранять небольшой размер формул в процессе вычисления.
  - При осуществлении дизъюнкции и вычислении предусловий число дизъюнктов-областей в пространстве состояний быстро растёт
    - Часто выполнять слияние областей очень вычислительно дорого
  - Используется эвристика разбиения пространства так, что состояния с похожими свойствами попадают в один класс

Символьная проверка моделей

Введение

- Эффективный алгоритм, который использует представление моделей Крипке в символьном виде (OBDD) для проверки выполнимости формулы на модели.
- Символьный алгоритм верификации моделей
  - основан на манипуляциях с символьными представлениями данных (булевыми формулами).
- OBDD представляют множества состояний и переходов
  - операции с множествами
    - не с отдельными состояниями и переходами
- Определение операторов темпоральной логики в терминах неподвижной точки.
- Множество $S' \subseteq S$ — неподвижная точка функции τ: $P(S) \rightarrow P(S)$
  - если $\tau(S') = S'$.

Введение

1. Описание семантики CTL-формулы при помощи наименьшей или наибольшей неподвижной точки подходящей функции.
   - Для вычисления неподвижных точек применяется итеративная процедура, основанная на теоретико-множественных операциях.
2. Алгоритм символьной верификации моделей для CTL.
3. Ограничения справедливости и контрпримеры.
4. Алгоритм символьной верификации моделей для LTL.

1. Представления неподвижной точки

- Задана модель Крипке M = (S, R, L)
- Множество P(S) всех подмножеств S
  - решетка по отношению порядка включения $\subseteq$. P(S)
  - Элемент S' $\in$ P(S)
    - предикат на S, принимающий значение true в точности на всех состояниях из S'
  - Наименьший элемент решетки
    - пустое множество, False
  - Наибольший элемент решетки
    - множество S, True
- τ: P(S) $\rightarrow$ P(S) — преобразователь предикатов

1. Функция τ монотонная
   - из включения P $\subseteq$ Q следует τ(P) $\subseteq$ τ(Q);
2. Функция τ $\cup$-непрерывная
   - из цепочки включений $P_1 \subseteq P_2 \subseteq \ldots$ следует $\tau(\cup_i P_i) = \cup_i \tau(P_i)$;
3. Функция τ $\cap$-непрерывная
   - из цепочки включений $P_1 \supseteq P_2 \supseteq \ldots$ следует $\tau(\cap_i P_i) = \cap_i \tau(P_i)$.

1. Представления неподвижной точки

- $\tau^i(Z)$ — i-кратное применение τ к Z
  - $\tau^0(Z) = Z$ и $\tau^{i+1}(Z) = \tau(\tau^i(Z))$
- У монотонного преобразователя τ на P(S) всегда есть
  - наименьшая неподвижная точка $\mu Z.\tau(Z)$
    - $\mu Z.\tau(Z) = \cap \{ Z \mid \tau(Z) \subseteq Z\}$ для монотонного τ
      - $\mu Z.\tau(Z) = \cup_i \tau(\text{False})$, если τ является $\cup$-непрерывным
  - наибольшая неподвижная точка $\nu Z.\tau(Z)$
    - $\nu Z.\tau(Z) = \cup\{Z \mid \tau(Z) \supseteq Z\}$ для монотонного τ
      - $\nu Z.\tau(Z) = \cap_i \tau(\text{True})$, если τ является $\cap$-непрерывным

1. Представления неподвижной точки

- **Лемма 5**. Если S — конечное множество, а τ — монотонный преобразователь, то
  - τ также $\cup$-непрерывен и $\cap$-непрерывен.

- **Лемма 6**. Если τ — монотонный преобразователь, то
  - для любого $i \geq 0$ верно $\tau^i(\text{False}) \subseteq \tau^{i+1}(\text{False})$ и $\tau^i(\text{True}) \supseteq \tau^{i+1}(\text{True})$

- **Лемма 7**. Если τ — монотонный преобразователь, а S — конечное множество, то
  - существуют такие натуральные числа i_0 и j_0, что
    - для любого $i \geq i_0$ верно $\tau^i(\text{False}) = \tau^{i_0}(\text{False})$, и
    - для любого $j \geq j_0$ верно $\tau^j(\text{True}) = \tau^{j_0}(\text{True})$.

- **Лемма 8**. Если τ — монотонный преобразователь, а S — конечное множество, то
  - существуют такие натуральные числа i_0 и j_0, что
    - $\mu Z.\tau(Z) = \tau^{i_0}(\text{False})$ и $\nu Z.\tau(Z) = \tau^{j_0}(\text{True})$.

1. Представления неподвижной точки

- Вычисление наименьшей неподвижной точки монотонного преобразователя τ

```
function Lfp(Tau: Predicate Transformer): Predicate
    Q := False;
    Q' := Tau(Q);
    while (Q ≠ Q') do
      Q := Q';
      Q' := Tau(Q');
    end while;
    return(Q);
end function
```

1. Представления неподвижной точки

- Вычисление наибольшей неподвижной точки монотонного преобразователя τ

```
function Gfp(Tau: Predicate Transformer): Predicate
    Q := True;
    Q' := Tau(Q);
    while (Q ≠ Q') do
      Q := Q';
      Q' := Tau(Q');
    end while;
    return(Q);
end function
```

1. Представления неподвижной точки

- Инвариант цикла **`while`** в теле процедуры
  - задается отношением $(Q' = \tau(Q)) \wedge (Q' \subseteq \mu Z.\tau(Z))$.
- В начале i-й итерации цикла $Q \subseteq \tau^{i-1}(False)$ и $Q' \subseteq \tau^{i}(False)$.
- Верно $False \subseteq \tau(False) \subseteq \tau^2(False) \subseteq \ldots$ (л.6)
  - Максимальное число итераций оператора цикла ограничено количеством элементов в множестве S.
- На выходе из цикла $Q = \tau(Q)$ и $Q \subseteq \mu Z.\tau(Z)$.
  - Q — неподвижная точка $\Rightarrow \mu Z.\tau(Z) \subseteq Q \Rightarrow Q = \mu Z.\tau(Z)$.
- Возвращаемое значение — действительно наименьшая неподвижная точка.
- Наибольшая неподвижная точка
  - вычисляется аналогично
  - корректность доказывается аналогично

```
function Lfp(Tau: Predicate Transformer)
    Q := False;
    Q' := Tau(Q);
    while (Q ≠ Q') do
        Q := Q';
        Q' := Tau(Q');
    end while;
    return(Q);
end function
```

1. Представления неподвижной точки

- Сопоставим каждой формуле φ предикат $\{s \mid M, s \models \varphi\}$ на $P(S)$
  - Каждый базовый оператор CTL можно выразить в терминах наименьшей или наибольшей неподвижной точки подходящего преобразователя предикатов

- $\mathbf{AF}\varphi = \mu Z.\ \varphi \vee \mathbf{AX}\ Z = (\varphi \vee \mathbf{AX}\mathit{false}) \vee (\varphi \vee \mathbf{AX}\varphi) \vee (\varphi \vee \mathbf{AX}(\varphi \vee \mathbf{AX}\varphi)) \ldots$
- $\mathbf{EF}\varphi = \mu Z.\ \varphi \vee \mathbf{EX}\ Z$
- $\mathbf{AG}\varphi = \nu Z.\ \varphi \wedge \mathbf{AX}\ Z = (\varphi \wedge \mathbf{AX}\mathit{false}) \wedge (\varphi \wedge \mathbf{AX}\varphi) \wedge (\varphi \wedge \mathbf{AX}(\varphi \wedge \mathbf{AX}\varphi)) \ldots$
- $\mathbf{EG}\varphi = \nu Z.\ \varphi \wedge \mathbf{EX}\ Z$
- $\mathbf{A}[\varphi_1\ \mathbf{U}\ \varphi_2] = \mu Z.\ \varphi_2 \vee (\varphi_1 \wedge \mathbf{AX}\ Z)$
- $\mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2] = \mu Z.\ \varphi_2 \vee (\varphi_1 \wedge \mathbf{EX}\ Z)$
- $\mathbf{A}[\varphi_1\ \mathbf{R}\ \varphi_2] = \nu Z.\ \varphi_2 \wedge (\varphi_1 \vee \mathbf{AX}\ Z)$
- $\mathbf{E}[\varphi_1\ \mathbf{R}\ \varphi_2] = \nu Z.\ \varphi_2 \wedge (\varphi_1 \vee \mathbf{EX}\ Z)$

- Наименьшие неподвижные точки
  - свойства, которые должны выполняться когда-нибудь
- наибольшие неподвижные точки
  - свойства, которые должны выполняться всегда

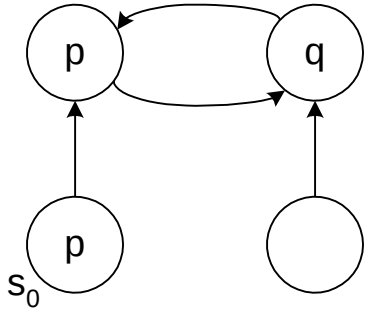
1. Представления неподвижной точки

- Обоснуем равенства для **EG** и **EU**
- Леммы 9-12 $\Rightarrow$ **EG**$\varphi = \nu Z.\ \varphi \wedge$ **EX** Z
- **Лемма 9**. Преобразователь $\tau(Z) = \varphi \wedge$ **EX** Z монотонный.
- **Лемма 10**. Пусть $\tau(Z) = \varphi \wedge$ **EX** Z, $\tau^{j_0}(\text{True})$ — предел последовательности True $\supseteq \tau(\text{True}) \supseteq$ ... Тогда
  - для любого $s \in \tau^{j_0}(\text{True})$ верно $s \models \varphi$ и
  - найдется такое s', что $(s, s') \in R$ и $s' \in \tau^{j_0}(\text{True})$.
- **Лемма 11**. Предикат **EG**φ является неподвижной точкой преобразователя $\tau(Z) = \varphi \wedge$ **EX** Z.
- **Лемма 12**. Предикат **EG**φ является наибольшей неподвижной точкой преобразователя $\tau(Z) = \varphi \wedge$ **EX** Z.

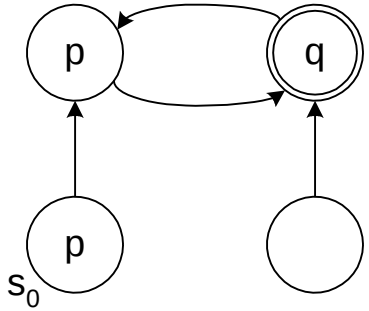
- **Лемма 13**. Предикат **E**$[\varphi_1$ **U** $\varphi_2]$ является наименьшей неподвижной точкой преобразователя $\tau(Z) = \varphi_2 \vee (\varphi_1 \wedge$ **EX** $Z)$

1. Представления неподвижной точки

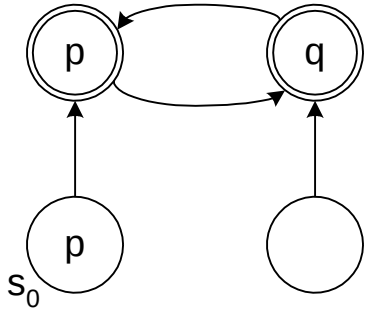
- Sem(**E**[p **U** q]) вычисляется при помощи процедуры *Lfp*.
  - $\tau(Z) = q \vee (p \wedge \mathbf{EX}\ Z)$
  - последовательность τ^i(False) сходится к **E**[p **U** q].
  - Состояния-аппроксимации в двойном кружке
- $\tau^3(\text{False}) = \tau^4(\text{False})$
  - **E**[p **U** q] = τ^3(False)
- $s_0 \in \tau^3(\text{False}) \Rightarrow M, s_0 \models \mathbf{E}[p\ \mathbf{U}\ q]$.



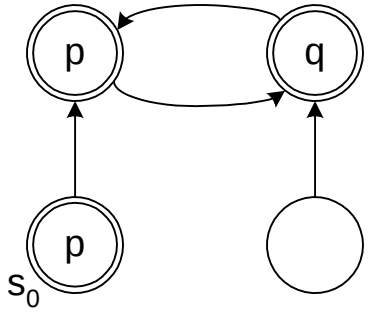
Модель Крипке



τ^1(False)



τ^2(False)



τ^3(False)

2. Символьная верификация моделей для CTL

- Алгоритм верификации моделей для CTL
  - $O(|\varphi|(|S| + |R|))$
  - Параллельная система из многих процессов и компонент
    - резкое увеличение размеров модели
- Символьный алгоритм верификации моделей для CTL
  - Модели Крипке представлены в символьном виде при помощи OBDD

2.1. Квантифицированные булевы формулы

- Сокращенная система обозначений для сложных операций на булевых формулах
  - Логика квантифицированных булевых формул (QBF)
- Пропозициональные переменные $V = \{v_0, \ldots, v_{n-1}\}$.
- QBF(V) — наименьшее множество формул таких, что:

1. каждая переменная из V — формула
2. если φ и ψ — формулы, то $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$ — формулы
3. если φ — формула и $v \in V$, то $\exists v.\varphi$ и $\forall v.\varphi$ — формулы

- Оценка значений истинности для множества QBF(V)
  - функция $\sigma: V \to \{0,1\}$.
  - $\sigma\langle v \leftarrow a\rangle$ — истинностное означивание ($a \in \{0,1\}$)
    - $\sigma\langle v \leftarrow a\rangle(w) = \{ a, \text{если } v=w\} \wedge \{\sigma(w), \text{если } v \neq w\}$

2.1. Квантифицированные булевы формулы

- Запись $\sigma \models \varphi$, где $\varphi \in$ QBF, а σ — истинностное означивание
  - φ истинна при означивании σ
- Отношение $\models$ определяется индуктивно:
- $\sigma \models v \Leftrightarrow \sigma(v) = 1$
- $\sigma \models \neg\varphi \Leftrightarrow \sigma \not\models \varphi$
- $\sigma \models \varphi_1 \wedge \varphi_2 \Leftrightarrow \sigma \models \varphi_1$ и $\sigma \models \varphi_2$
- $\sigma \models \varphi_1 \vee \varphi_2 \Leftrightarrow \sigma \models \varphi_1$ или $\sigma \models \varphi_2$
- $\sigma \models \exists v.\varphi \Leftrightarrow \sigma\langle v \leftarrow 0\rangle \models \varphi$ или $\sigma\langle v \leftarrow 1\rangle \models \varphi$
- $\sigma \models \forall v.\varphi \Leftrightarrow \sigma\langle v \leftarrow 0\rangle \models \varphi$ и $\sigma\langle v \leftarrow 1\rangle \models \varphi$
- Выразительные возможности QBF такие же, как и у обычных пропозициональных формул.

2.1. Квантифицированные булевы формулы

- Каждая QBF-формула задает n-местное булево отношение на множестве V
  - все оценки переменных из V, обращающие формулу в истину
  - Одинаковые обозначения для QBF-формул и их отношений.
- Можно определить OBDD для формул пропозициональной логики.
- Операции наложения кванторов
  - Операция ограничения и операция `Apply`
    - $\exists v.\varphi = \varphi|_{v \leftarrow 0} \vee \varphi|_{v \leftarrow 1}$
    - $\forall v.\varphi = \varphi|_{v \leftarrow 0} \wedge \varphi|_{v \leftarrow 1}$
- Кванторы в операциях реляционного произведения
  - $\exists v.(f(w,v) \wedge g(v,x))$

2.2. Символьный алгоритм верификации моделей

- Символьный алгоритм верификации моделей для CTL
  - Процедура `Check`
    - аргумент — CTL-формула
    - результат — OBDD, представляющую состояния системы, в которых формула выполняется
    - неявный аргумент — OBDD для отношения переходов
- `Check` индуктивна по структуре CTL-формул
  - φ — это атомарное высказывание p
    - `Check`(φ) — OBDD для $L^{-1}(p)$
      - стандартный алгоритм для символьного представления
  - $\varphi = \varphi_1 \wedge \varphi_2$ или $\varphi = \neg\varphi_1$
    - `Check`(φ) — OBDD = `Apply`(`Check`(φ_1), `Check`(φ_2))
      - стандартный алгоритм для символьного представления
  - **EX**φ, **E**[φ**U**ψ] и **EG**φ:
    - `Check`(**EX**φ) = `CheckEX`(`Check`(φ)),
    - `Check`(**E**[φ**U**ψ]) = `CheckEU`(`Check`(φ),`Check`(ψ)),
    - `Check`(**EG**φ) = `CheckEG`(`Check`(φ)).
    - аргументы — OBDD, остальные CTL-операторы выразимы

2.2. Символьный алгоритм верификации моделей

- Процедура `CheckEX`
  - определение темпорального оператора **EX**:
    - **EX**φ истинна в состоянии, если есть последователь, в котором истинна формула φ.
  - `CheckEX`($\varphi(v)$) = $\exists v'.(\varphi(v') \wedge R(v,v'))$
    - R(v, v') — OBDD, представляющая отношение переходов
- Процедура `CheckEG`
  - формула наибольшей неподвижной точки для **EG**
    - **EG**$\varphi = \nu Z.\ \varphi \wedge$ **EX** Z
  - OBDD для φ
  - *Gfp* вычисляет OBDD для **EG**φ

2.2. Символьный алгоритм верификации моделей

- Процедура `CheckEU`
  - формула наименьшей неподвижной точки для **EU**
    - $\mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2] = \mu Z.\ \varphi_2 \vee (\varphi_1 \wedge \mathbf{EX}\ Z)$
  - *Lfp* для вычисления последовательности $Q_0, Q_1, Q_2, \ldots, Q_i, \ldots$ сходящейся к $\mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2]$ за конечное число шагов
  - OBDD для φ_1, φ_2. текущего приближения Q_i
    - можно вычислить Q_{i+1}
  - сходимость легко проверить
    - OBDD — каноническое представление булевых функций
  - Конец вычислений *Lfp*
    - Выполнение равенства $Q_i = Q_{i+1}$
  - OBDD для Q_i
    - Множество состояний для $\mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2]$

3. Справедливость при символьной верификации моделей

- Обобщим символьный алгоритм верификации моделей для CTL на ограничения справедливости
- Ограничения справедливости
  - CTL-формулы $F = \{P_1,...,P_n\}$
- Процедура `CheckFair`
  - для проверки выполнимости CTL-формул с учетом ограничений справедливости F
  - `CheckFairEX`, `CheckFairEU` и `CheckFairEG`
    - аналоги вспомогательных процедур в `Check`

3. Справедливость при символьной верификации моделей

- **EG**φ с ограничением справедливости F
  - существует путь из текущего состояния s, на котором
    - φ выполняется всюду
    - каждая формула из F выполняется бесконечно часто
- Множество Z всех таких состояний s
  1. все состояния из Z удовлетворяют φ
  2. для любого P_k из F и состояния s из Z существует
     - последовательность состояний длины не менее 1,
       - ведущая из s в одно из состояний множества Z,
         - удовлетворяющее P_k, причем
       - в каждом состоянии последовательности выполняется φ.

3. Справедливость при символьной верификации моделей

- Множество Z всех таких состояний s
  1. все состояния из Z удовлетворяют φ
  2. для любого P_k из F и состояния s из Z существует
     - последовательность состояний длины не менее 1,
       - ведущая из s в одно из состояний множества Z,
         - удовлетворяющее P_k, причем
       - в каждом состоянии последовательности выполняется φ.

- Удобно для символьной верификации моделей
  - $\mathbf{EG}\varphi = \nu Z.\ \varphi \wedge \wedge^{n}_{k=1}\mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$
  - данную формулу нельзя выразить в CTL

3. Справедливость при символьной верификации моделей

- Корректность уравнения $\mathbf{EG}\varphi = \nu Z.\ \varphi \wedge \wedge^{n}_{k=1}\mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$
- **Лемма 14**. Справедливая разновидность оператора $\mathbf{EG}\varphi$ является неподвижной точкой уравнения
  - $Z = \varphi \wedge \wedge^{n}_{k=1}\mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$ (6.2)
  - Решение включается в наибольшую неподвижную точку.
- **Лемма 15**. Наибольшая неподвижная точка формулы из уравнения (6.2) включается в $\mathbf{EG}\varphi$.
- $\Rightarrow \mathbf{EG}\varphi$ является наибольшей неподвижной точкой (6.2)

- `CheckFairEG`$(\varphi(v))$
  - $\nu Z(v).\ \varphi(v) \wedge \wedge^{n}_{k=1}\mathbf{EXE}[\ \varphi(v)\ \mathbf{U}\ (Z(v) \wedge P_k)\ ]$
  - вложенные вычисления неподвижных точек (`CheckEU`)

3. Справедливость при символьной верификации моделей

- Проверка **EX**φ и **E**[φ_1**U**φ_2] при ограничениях справедливости
  - похоже на случай явного представления состояний
- Состояния, которыми начинаются справедливые вычисления
  - fair(v) = `CheckFair` (**EG** True)
- **EX**φ истинна при ограничениях справедливости
  - в s $\Leftrightarrow$ есть последователь s', удовлетворяющий формуле φ, из которого исходит хотя бы один справедливый путь.
  - **EX**($\varphi \wedge$fair) (без ограничений справедливости)
  - `CheckFairEX`(φ(v)) = `CheckEX`(φ(v) $\wedge$ fair(v)).
- **E**[φ_1**U**φ_2] (при ограничениях справедливости)
  - **E**[φ_1**U**($\varphi_2 \wedge$ fair)] (без ограничений справедливости)
  - `CheckFairEU`(φ_1(v), φ_2(v)) = `CheckEU`(φ_1(v), φ_2(v) $\wedge$ fair(v))

4. Контрпримеры и свидетельства

- Контрпримеры
  - Верификатор моделей с контрпримерами
    - формула с универсальным квантором пути ложна
      - находит вычислительный путь, где выполняется отрицание заданной формулы
        - **AG**φ неверна
        - путь в состояние, в котором выполняется $\neg\varphi$

- Свидетельства
  - Верификатор моделей со свидетельствами
    - формула с экзистенциальным квантором пути верна
      - находит вычислительный путь, где выполняется заданная формула
        - **EF**φ верна
        - путь в состояние, удовлетворяющее φ.

4. Контрпримеры и свидетельства

- Контрпример для формулы с квантором всеобщности — это свидетельство для двойственной формулы с квантором существования.
- Достаточно уметь строить свидетельства для **EX**, **EG** и **EU**.

- Рассмотрим сильно связные компоненты графа переходов, который определяется моделью Крипке.
- Сформируем новый граф
  - вершины — сильно связные компоненты
  - дуги: две сильно связные компоненты соединены ⇔
    - есть дуга из состояния одной в состояние другой
  - не содержит никаких циклов, отличных от петель
    - всякий цикл в графе не выходит за пределы одной сильно связной компоненты.
  - каждый бесконечный путь имеет суффикс, целиком содержится внутри одной сильно связной компоненты.
    - конечные модели Крипке

4. Контрпримеры и свидетельства

- Поиск свидетельства для формулы **EG**φ
  - при ограничениях справедливости $F = \{P_1,\ldots,P_n\}$.
  - ограничение P_i — множество состояний, на котором оно истинно
- Состояния для **EG**φ с ограничениями справедливости F
  - $\nu Z.\ \varphi \wedge \wedge^{n}_{k=1}\mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$
  - **EG**φ — обозначение для этих состояний
- Для $s \in$ **EG**φ найти путь π из s
  - проходящий только через состояния удовлетворяющие φ
  - проходящий через каждое множество $P_i \in F$ бесконечно часто

4. Контрпримеры и свидетельства

- Этот путь состоит из
  - конечного префикса, за которым есть повторяющийся цикл
- Построение пути
  - последовательно наращивать префикс
  - пока не найдётся цикл
  - можно делать шаге построения, если
    - построенный префикс можно продолжить до справедливого пути, на котором везде φ
      - новое состояние для текущего префикса
        - если оно удовлетворяет формуле **EG**φ

4. Контрпримеры и свидетельства

- Сначала вычислим $\nu Z.\ \varphi \wedge \wedge^{n}_{k=1} \mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$
  - На каждом шаге вычисления внешней неподвижной точки
    - вычисляется множество наименьших неподвижных точек
      - $\mathbf{E}[\ \varphi\ \mathbf{U}\ (Z \wedge P)\ ]$ для каждого ограничения $P \in F$.
  - Для каждого P получается возрастающая последовательность
    - $Q^P_0 \subseteq Q^P_1 \subseteq Q^P_2 \subseteq \ldots$
      - Q^P_i — множество состояний, из которых
        - состояние из $Z \wedge P$ может быть достигнуто
          не более чем за i шагов
          выполняется формула φ.
- На последней итерации вычисления внешней неподвижной точки
  - когда $Z = \mathbf{EG}\varphi$
  - сохранить последовательность Q^P_i для каждого P из F

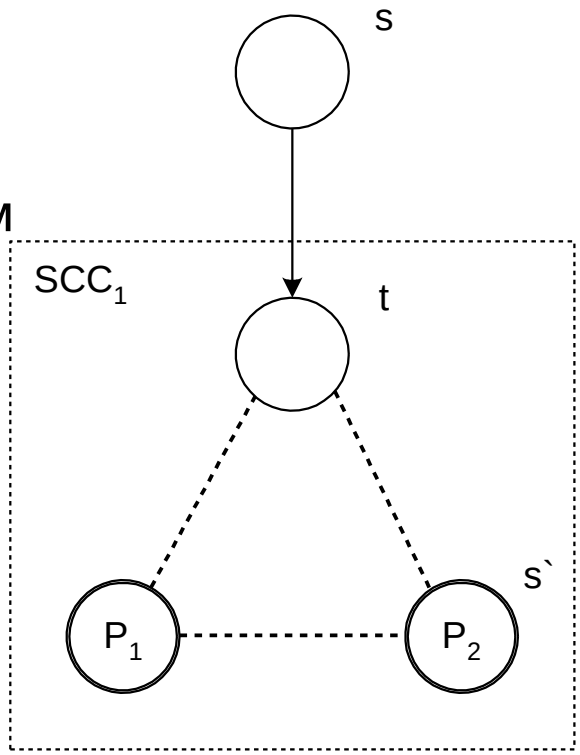
4. Контрпримеры и свидетельства

$$\nu Z.\ \varphi \wedge \wedge^{n}_{k=1} \mathbf{EXE}[\ \varphi\ \mathbf{U}\ (Z \wedge P_k)\ ]$$

- Дано начальное состояние s, удовлетворяющее **EG**φ.
  - Тогда $s \in \mathbf{EG}\varphi \Rightarrow$
    - есть последователь в **E**[φ **U** $(Z \wedge P)$] для каждого $P \in F$
- Выбираем ближайшее ограничение справедливости, достижимое из s
  - чтобы минимизировать длину пути-свидетельства
  - Найти последователя t состояния s в Q^P_0 для каждого $P \in F$
    - Если такое состояние t не обнаружено, поискать в Q^P_1 для всех $P \in F$.
    - Если не нашли t, то поискать в Q^P_2 и т. д.
  - Рано или поздно найдётся в Q^P_i
    - s принадлежит **EG**φ
- Из t выходит путь длины i в состояние из (**EG**φ) $\wedge$ P
  - поэтому t принадлежит **EG**φ

4. Контрпримеры и свидетельства

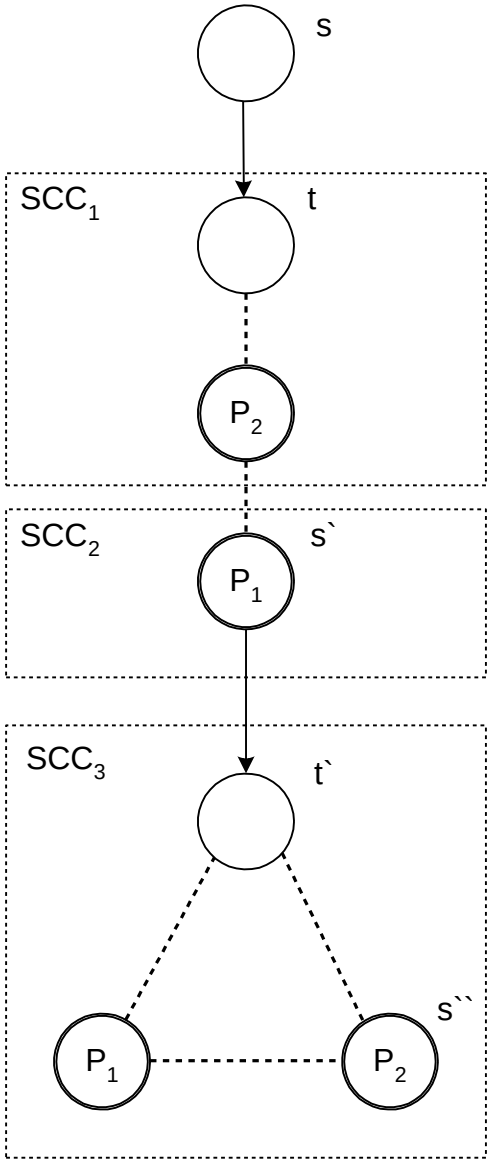
- Если i > 0 для t найдётся последователь в Q^P_{i-1}
  - найти последователей t, пересечь с Q^P_{i-1}, выбрать произвольного
- Выбирать состояния пока не будет i = 0
  - нашли путь из s в некоторое состояние u из (**EG**φ) $\wedge$ P.
- Исключаем из рассмотрения P
- Повторить процедуру для состояния u (вместо s)
  - пока не будут учтены все ограничения справедливости
- Пусть s' — последнее состояние этого пути
- Надо попасть из s' в t невырожденным φ-путем
  - чтобы замкнуть цикл
- То есть требуется свидетельство для формулы {s'} $\wedge$ **EXE**[φ **U** {t}]
  - Если эта формула верна, то нашли путь-свидетельство для s



4. Контрпримеры и свидетельства

- Если же формула {s'} $\wedge$ **EXE**[φ **U** {t}] ложна

1. Повторить процедуру для s‘ и всего списка F
   - Поскольку {s'} $\wedge$ **EXE**[φ **U** {t}] ложна
     - s' не лежит в сильно связной компоненте φ, содержащей t
     - s' $\in$ **EG**φ.
   - Спускаемся по графу сильно связных компонент
     - либо найдётся цикл π,
     - либо достигнем крайней сильно связной компоненты φ
       - цикл найдётся, поскольку покинуть эту крайнюю сильно связную компоненту не удастся
2. Предварительно вычислить **E**[φ **U** {t}]
   - Как только в первый раз покидаем это множество
     - цикл замкнуть не удастся
       - перезапустить процедуру, начиная с этого состояния



4. Контрпримеры и свидетельства

- Эти подходы обнаруживают кратчайшие контрпримеры
  - число сильно связных компонент, как правило, невелико)
  - нет никаких попыток отыскать самый короткий цикл
- Свидетельства для **E**[φ **U** ψ] и **EX**φ при ограничениях справедливости
  - fair — состояния, удовлетворяющих **EG**True при ограничениях F.
  - Вычислить **E**[φ **U** ψ] при условии F
    - Стандартный алгоритм для **E**[φ **U** ($\psi \wedge$fair)].
  - Вычислить **EX**φ при условии F
    - Стандартный алгоритм для **EX**($\varphi \wedge$fair).
- Расширить свидетельства для формул **E**[φ **U** ψ] и **EX**φ до бесконечных путей
  - процедура отыскания свидетельств для **EG**True при ограничениях справедливости F.