Верификация программ на моделях

Лекция № 11

Символьные алгоритмы, хранение множеств состояний

Игорь Коннов

План лекции

- Простой алгоритм поиска достижимых состояний (идея символьных алгоритмов).
- Символьный алгоритм верификации модели.
- Двоичные разрешающие диаграммы (BDD).
- Хранение множеств состояний в Spin:
  - BDD,
  - MDFA.

Символьные алгоритмы

Поиск достижимых состояний

- Можно осуществлять снова поиском в глубину.
- А можно операциями над множествами!
- M = (S, Act, $\rightarrow_a$, s_0, AP, L).
- Для X ⊆ S обозначим Succ(X) = { t | найдутся такие s ∈ S и a ∈ Act , что (s, a, t) ∈ → }.

Символьный алгоритм поиска достижимых состояний

```
Reach := {s0}; T := ∅;
while Reach != T {
   T := Reach;
   Reach:=Reach ∪ Succ(Reach);
}
```

Символьные алгоритмы

- Вместо перебора состояний по отдельности используются операции над множествами.
- На этой идее основан целый класс *символьных алгоритмов* верификации моделей (для CTL, LTL и других логик).
- Почему (интуитивно понятная) идея поиска достижимых состояний работает?
- Как эффективно реализовать операции над множествами?

Почему это работает?

- TS = $(S, Act, \rightarrow_a, s_0, AP, L)$.
- Множество S конечно.
- На множестве 2^S задано отношение порядка $\subseteq$ (т.е. $(2^S, \subseteq)$ – решётка):
  - наименьший элемент – $\varnothing$,
  - наибольший элемент – S.
- Оператор $T: 2^S \rightarrow 2^S$ называется *преобразователем предикатов*.

Свойства преобразователя

- Оператор T – монотонный, если из $P \subseteq Q$ следует $T(P) \subseteq T(Q)$.
- Оператор T – $\cup$-непрерывный, если из цепочки $P_1 \subseteq P_2 \subseteq \ldots$ следует $T(\cup_i P_i) = \cup_i T(P_i)$.
- Оператор T – $\cap$-непрерывный, если из цепочки $P_1 \supseteq P_2 \supseteq \ldots$ следует $T(\cap_i P_i) = \cap_i T(P_i)$.
- Пусть задано множество $Z \subseteq 2^S$:
  - $T^0(Z) = Z$,
  - $T^{i+1}(Z) = T(T^i(Z))$.

Следствия теоремы Тарского

- Множество $X \in 2^S$ – неподвижная точка, если $T(X) = X$.
- У монотонного преобразователя T на 2^S есть наибольшая неподвижная точка $\mu Z.T(Z) = \cap\{Z \mid T(Z) \subseteq Z\}$ и наименьшая неподвижная точка $\nu Z.T(Z) = \cup\{Z \mid T(Z) \supseteq Z\}$.
- Для ∩-непрерывного преобразователя верно $\mu Z.T(Z) = \cup_i T^i(\emptyset)$.
- Для ∪-непрерывного преобразователя верно $\nu Z.T(Z) = \cap_i T^i(S)$.

Леммы для систем переходов

Лемма 1. Если S – конечное множество, а T – монотонный преобразователь на конечной системе переходов, то T также $\cap$-непрерывен и $\cup$-непрерывен.

Лемма 2. Если T – монотонный преобразователь, то для любого натурального числа i имеют место включения: $\mathsf{T}^i(\varnothing) \subseteq \mathsf{T}^{i+1}(\varnothing)$ и $\mathsf{T}^i(\mathsf{S}) \supseteq \mathsf{T}^{i+1}(\mathsf{S})$.

Леммы для систем переходов

Лемма 3. Если T – монотонный преобразователь, а S – конечное множество, то существуют такие натуральные числа m и n, что для любых $i \geq m$ и $j \geq n$ верно: $T^i(\varnothing) = T^m(\varnothing)$ и $T^j(S) = T^n(S)$.

Лемма 4. Если T – монотонный преобразователь, а S – конечное множество, то существуют такие натуральные числа m и n, что $\mu Z.T(Z) = T^m(\varnothing)$ и $\nu Z.T(Z) = T^n(S)$.

Вычисление неподвижной точки

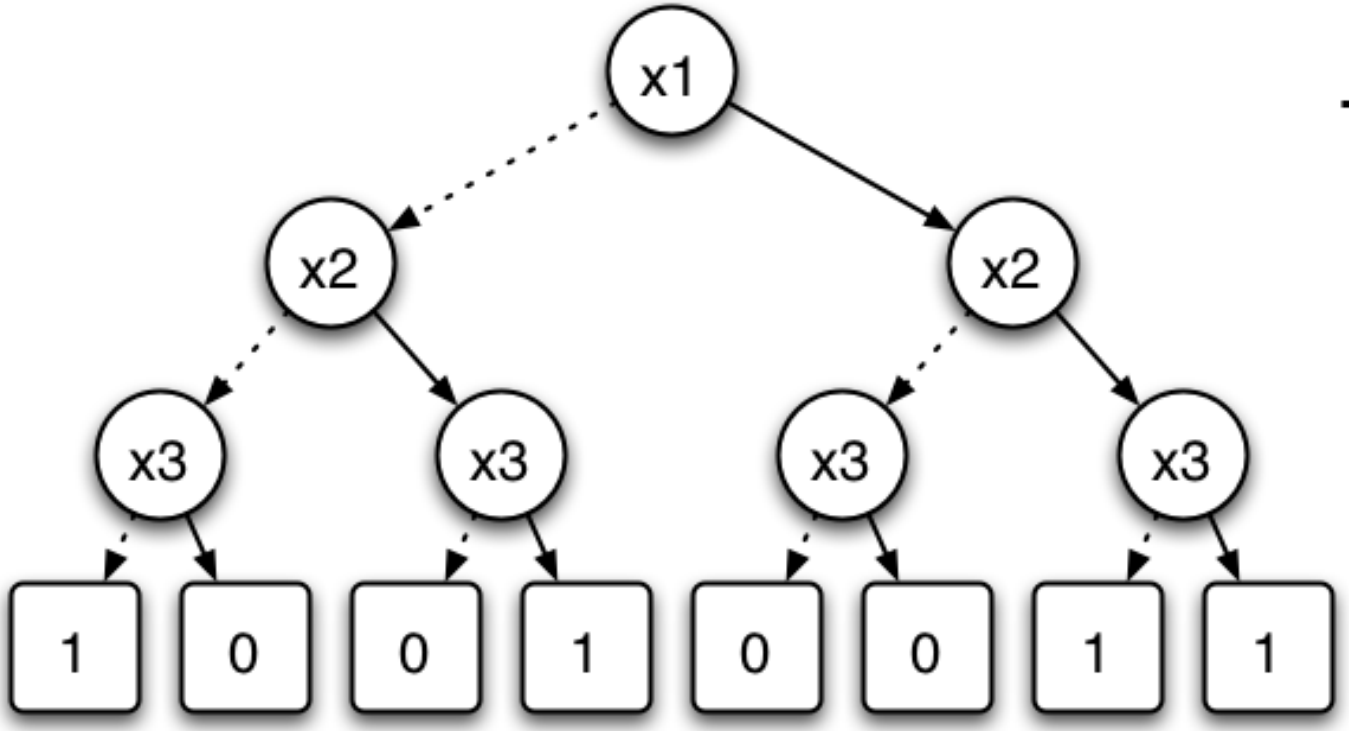
```
Q := ∅; Q' := T(Q);
while Q != Q' {
   Q := Q';
   Q':=T(Q');
}
//post: Q – неподвижная точка
```

Двоичные разрешающие диаграммы (BDD)

Двоичные разрешающие диаграммы (BDD)

- С помощью BDD задают булевы функции от *n* переменных $f(x_1, \ldots, x_n)$.
- Ориентированный ациклический граф:
  - выделены две завершающие вершины: 0 и 1.
  - Из каждой вершины u (кроме 0 и 1) выходят две дуги low(u) и high(u).
  - Каждая вершина u помечена переменной var(u).

Пример BDD



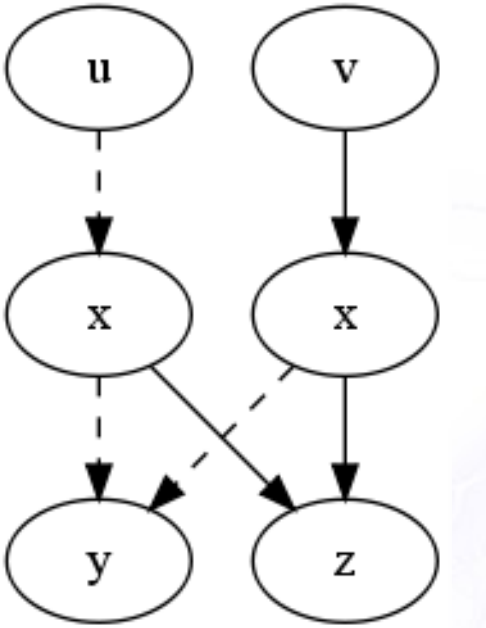
x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

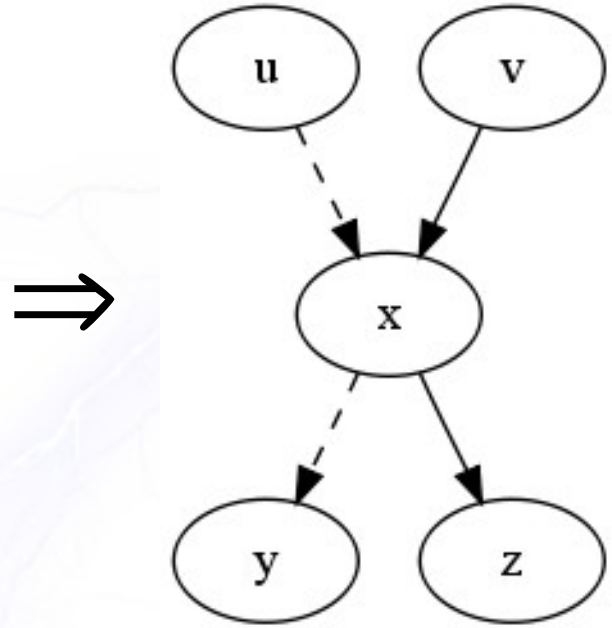
http://en.wikipedia.org/wiki/Binary_decision_diagram

ROBDD

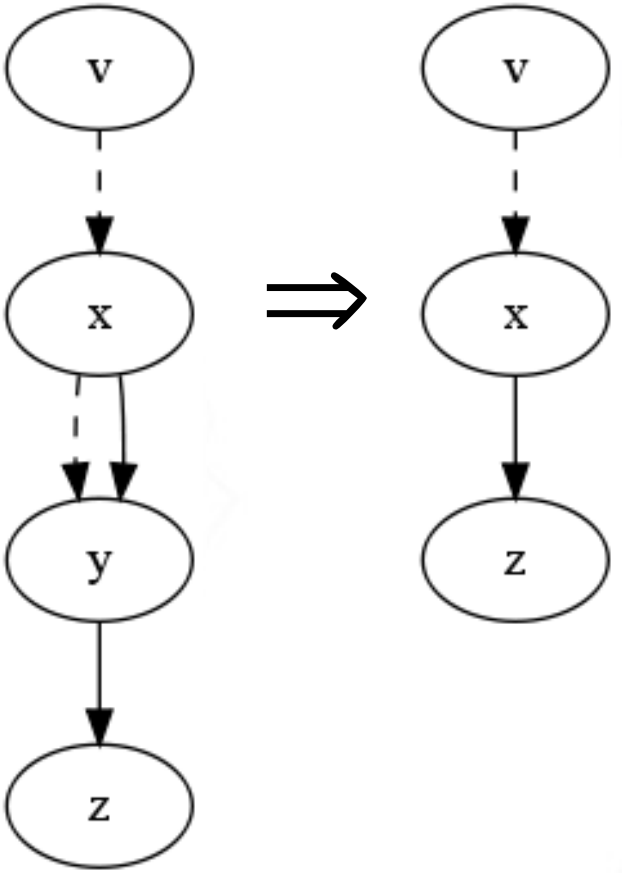
- На каждом пути порядок следования переменных фиксирован: $x_1 \prec x_2 \prec \ldots \prec x_n$
- Если var(u) = var(v), low(u) = low(v), high(u) = high(v), то u =v.
- Для каждой вершины u верно low(u) != high(u).
- Каноническое представление, т.е.
  - Если $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$, то BDD для f и g совпадают.

Иллюстрации правил редукции





Правило 1



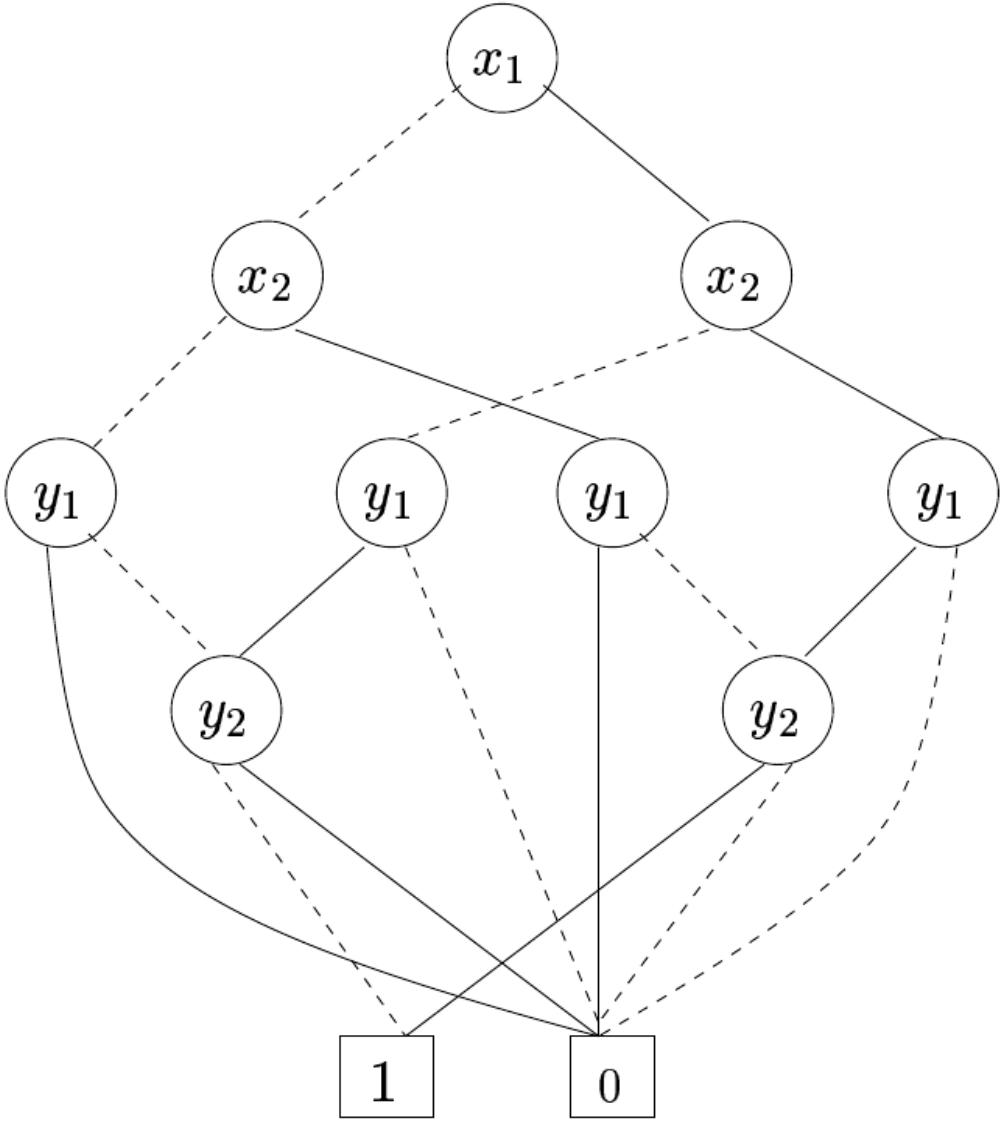
Правило 2

Пример

Булева функция для

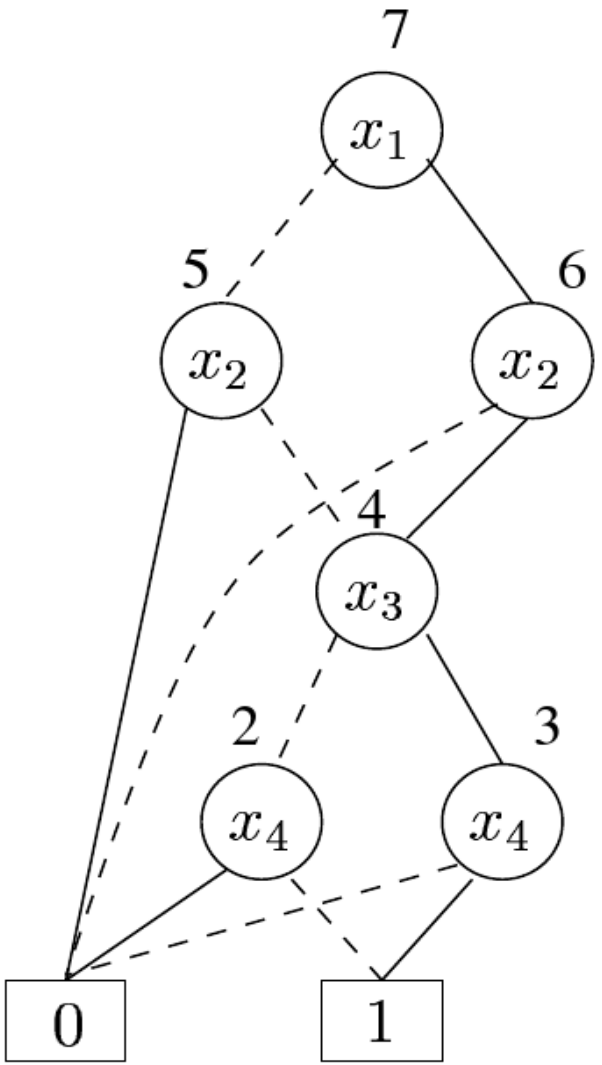
$(x_1 \leftrightarrow y_1) \wedge (x_2 \leftrightarrow y_2)$

Порядок: $x_1 \prec x_2 \prec y_1 \prec y_2$



Henrik Reif Andersen. An Introduction to Binary Decision Diagrams

Ещё пример



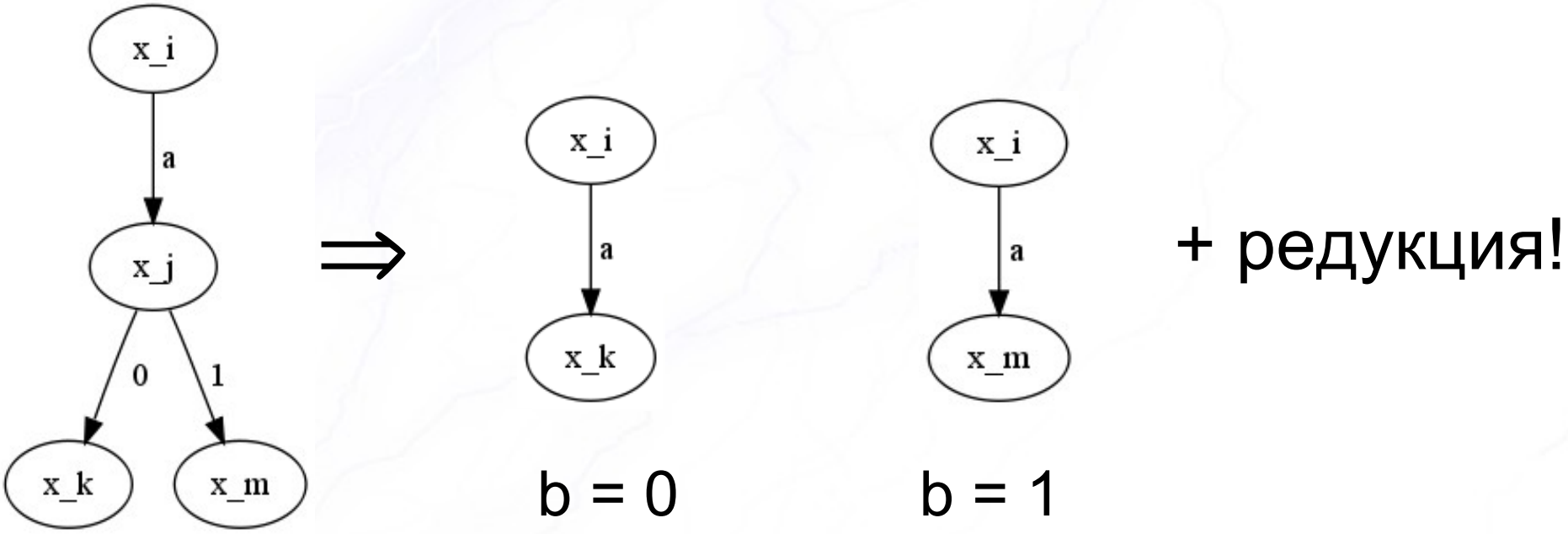
$T : u \mapsto (i, l, h)$

u	var	low	$high$
0	5		
1	5		
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6

Henrik Reif Andersen. An Introduction to Binary Decision Diagrams

Операция ограничения

- Пусть задана функция $f(x_1, \ldots, x_n)$ и её ROBDD B_1.
- Как построить ROBDD $f[x_j \leftarrow b]$, т.е. $f(x_1, \ldots, x_{j-1}, b, x_{j+1}, \ldots, x_n)$?



Реализация булевых функций

- Разложение Шеннона:
  - $f = (\neg x \wedge f[x \leftarrow 0]) \vee (x \wedge f[x \leftarrow 1])$.
- Пусть $*$ обозначает $\vee$, $\wedge$ или $\neg$.
- Посмотрим, как построить $f_1 * f_2$.
- Пусть v и v' – корневые вершины ROBDD для f_1 и f_2.
- $x = var(v)$ и $x' = var(v')$.

Варианты 1, 2

1. Если v и v' – терминальные вершины (0 или 1), то $f_1 * f_2 = value(v) * value(v')$,

2. Если x = x', то

$$f_1 * f_2 = (\neg x \wedge (f_1[x \leftarrow 0] * f_2[x \leftarrow 0])) \vee (x \wedge (f_1[x \leftarrow 1] * f_2[x \leftarrow 1])).$$

Поэтому построим ROBDD $u_1 \rightarrow (f_1[x \leftarrow 0] * f_2[x \leftarrow 0])$ и $u_2 \rightarrow (f_1[x \leftarrow 1] * f_2[x \leftarrow 1])$ и создадим вершину w: $var(w) = x$, $low(w) = u_1$, $high(w) = u_2$.

Варианты 3, 4

3. Если $x < x'$, то $f_2[x \leftarrow 0] = f_2[x \leftarrow 1] = f_2$ и

   $f_1 * f_2 = (\neg x \wedge (f_1[x \leftarrow 0] * f_2))$

   $\vee (x \wedge (f_1[x \leftarrow 1] * f_2)).$

   Далее, аналогично варианту 2.
4. Если $x' < x$, то аналогично варианту 3.

Операция apply

- Мы описали операцию apply:
  - На каждом шаге рекурсивно строится BDD для функций в разложении Шеннона
- Для предотвращения экспоненциального роста используется кэш (g_1 , g_2 , $*$) для функций, вычисляемых в процессе построения.
- Для удаления неиспользуемых вершин используется сборщик мусора.

Что же хорошего в BDD?

- На BDD определяются операции ∧, ∨ и ¬.
- Их сложность равна $O(m*n)$, где m и n – число вершин в BDD.
- Есть операторы проекции и квантор существования:
  - f(x, 0, 1, z) {сложность $O(m)$}
  - $\exists\, y\,(f(x, y, z))$ {сложность $O(m*m)$}

Как представить наши множества в виде BDD?

- Множество S, |S| = n. Выберем k: $n \leq 2^k$
  - Состояние $s \in S$ кодируется с помощью переменных $x_1, \ldots, x_k$.
  - Множество кодируется с помощью булевой функции $f(x_1, \ldots, x_k)$.
- Отношение переходов кодируется с помощью булевой функции $T(x_1, \ldots, x_k, x'_1, \ldots, x'_k)$.

Применение символьных алгоритмов и BDD

- Для логики ветвящегося времени CTL символьные алгоритмы (с BDD) применяются для проверки формулы табличным методом:
  - Верификатор моделей SMV (МакМиллан).
- Для LTL также применимы символьные алгоритмы с BDD:
  - Задачу верификации LTL можно свести к задаче проверки CTL с ограничениями справедливости.

- Э. Кларк мл., О. Грамберг, Д. Пелед. Верификация моделей программ: Model Checking. Издательство Московского центра непрерывного математического образования. -- М. 2002

Символьное представление множеств состояний в Spin

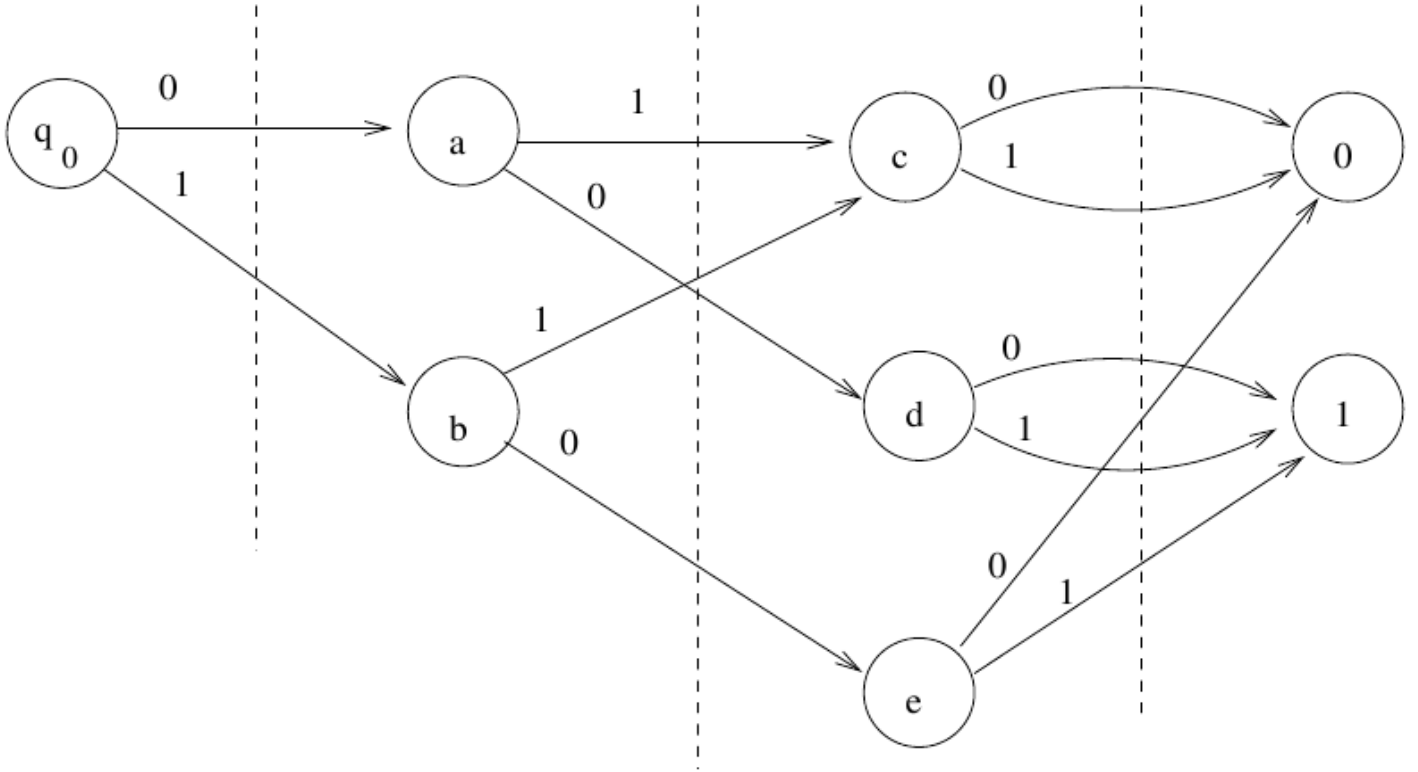
Spin и BDD

- BDD можно применять для хранения visited, nvisited и stack.
- В Spin используется другое представление: k-уровневый минимизированный конечный автомат.
  - Состояние кодируется с помощью k слов над алфавитом $\Sigma = \{0, 1, ..., 255\}$.
- Автомат, соответствующий множеству, попадает в допускающее состояние на слове ⇔ слово соответствует элементу, принадлежащему множеству.

Минимизированый ДКА

- Автомат $A = (\{Q_i\}^k_{i=0}, \delta, \Sigma)$.
- В автомате k + 1 слой.
- $Q_0 = \{q_0\}$, $Q_k = \{0, 1\}$,
- $\delta: Q_i \times \Sigma \rightarrow Q_{i+1}$, $0 \leq i \leq k - 1$.
- $Q_i \cap Q_j = \varnothing$ для $i \neq j$.
- Автомат – минимизированный, т.е.:
  - $L(q_i) = L(q_j) \iff q_i = q_j$.

Пример



G. Holzmann, A. Puri. A Minimized Automaton Representation of Reachable States. International Journal on Software Tools for Technology Transfer (STTT), 1999, vol. 2(3), P. 270-278.

Операции

- Проверка вхождения слова в язык автомата (состояния в множество состояний): O(k).
- Добавление слова в автомат с минимизацией: $O(k \cdot |\Sigma|)$.
- Можно использовать вместо меток на рёбрах метки с диапазонами [i, j] – существенно сокращает размер автомата на практике.

Реализация в Spin

- Режим использования минимизированного ДКА включается опцией `-DMA=n`, где:
  - n – максимальное число байтов (слоёв) в автомате.

Спасибо за внимание!