1 BDDs: diagramas de decisão binários

Verificação por modelos simbólica (*Symbolic model checking*)

- Os estados dum sistema de transições correspondem a possíveis valores de variáveis e o seu número é geralmente exponencial no número de variáveis.
- O algoritmo de etiquetagem usa intensivamente conjuntos de estados em em particular o cálculo dos sucessores e predecessores de um estado.
- Uma das maneiras eficientes de representar os conjuntos de estados é considerar codificações em binário que correspondem a funções Booleanas
- e representá-las usando OBDDs (*ordered binary decision diagrams*).

OBDDs:*ordered binary decision diagrams*

- Os OBDDs servem para representar funções Booleanas.

$$f : \{0,1\}^n \to \{0,1\}$$

- Operações Booleanas:

  conjunção $f \cdot g$ ou $f \wedge g$
  disjunção $f + g$ ou $f \vee g$
  complemento $\bar{f}$ ou $\neg f$
  ...

- Uma função Booleana f de n variáveis pode ser representada por uma tabela com 2^n linhas.
- Também pode ser representada por fórmulas da lógica, mas também é ineficiente.

Funções Booleanas e BDDs

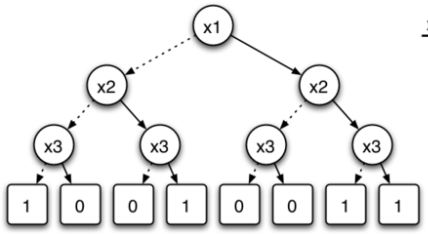
BDD: Binary decision diagrams

Uma função booleana pode ser representada por um digrafo acíclico com raiz que é constituído por:

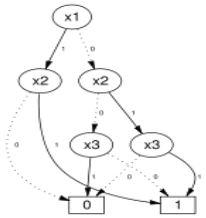
- nós de decisão (internos)
- apenas dois nós terminais: 0 e 1.
- cada nó de decisão é etiquetado por uma variável proposicional e tem dois filhos: cujos arcos (*tracejado* e `sólido`) correspondem às possíveis atribuições de valores às variáveis (0 e 1).

Função boolenana

Árvore de decisão



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

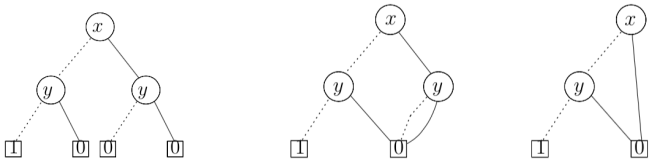


BDD

BDDs reduzidos

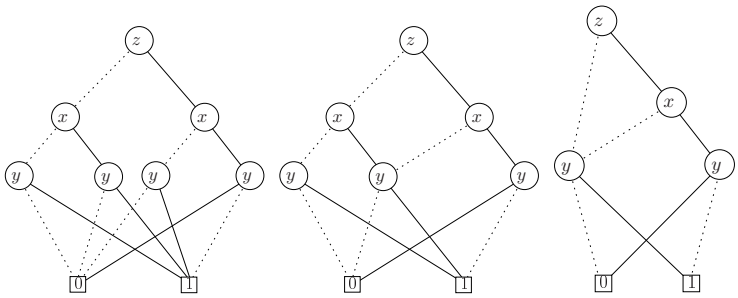
Um BDD é reduzido se os subgrafos isomorfos estão identificados e não tem nós cujos filhos são isomorfos:

R1 Testes redundantes: ambos os arcos de um nó n têm o mesmo nó destino; n pode ser eliminado.



R2 Nós de decisão redundantes: se são raizes de subBDDs estruturalmente idênticos; um deles pode ser eliminado. Isto garante também apenas dois nós folha.

Redução de BDDs

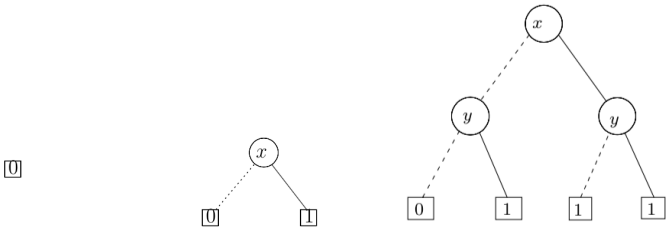


Eficiência da representação com BDDs

- Representação compacta
- Safisfazibilidade: determinar se há um caminho consistente desde a raiz que termine em 1. Um caminho é consistente se ao longo dele duma mesma variável que só saem arcos tracejados ou só arcos sólidos.
- Validade: nenhum nó terminal 0 é atingido por caminhos consistentes.
- Conjunção: dados B_f e B_g representando duas funções booleanas f e g, construir um BDD para $f \cdot g$ tal que cada nó 1 de B_f é substituído por B_g.
- Disjunção: como o anterior, mas substituindo todos os nós 0 de B_f por B_g (BDD para $f + g$)
- Complemento: $B_{\bar{f}}$ é obtido de B_f, substituindo os nós terminais 0 por 1 e vice-versa.

B_0, B_1, B_x e complemento

B_1 é análogo a B_0 mas com um 1



Mostra que o terceiro corresponde a $f_1(x, y) = x \vee y$ e calcula BDDs para $f(x, y) = \neg(x \vee y)$ e $f_2(x, y) = x \wedge y$.

1.1 BDDs ordenados

OBDD: Ordered Binary Decision Diagrams

Um BDD é ordenado

se as várias variáveis aparecem sempre na mesma ordem ao longo qualquer caminho desde a raíz. Isto induz uma ordenação no conjunto das variáveis.

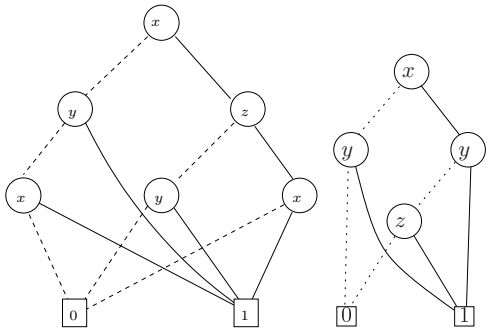
Seja $[x_1, \ldots, x_n]$ uma lista ordenada de variáveis. Um BDD B tem a ordem $[x_1, \ldots, x_n]$ se para qualquer ocorrência de x_i seguida de x_j ao longo de um caminho em B, temos que $i < j$.

Um BDD é ordenado se tem uma ordem para a lista das suas variáveis. Duas ordenações de dois OBDD B_1 e B_2 são compatíveis se não existirem duas variáveis x e y tal que $x < y$ em B_1 mas $y < x$ em B_2.

Teorema 17.1. *O OBDD reduzido que representa uma dada função booleana f é único a menos de isomorfismo. Isto é, dois OBDDs B e B' com ordenações compatíveis, se representarem a mesma função booleanas têm a mesma estrutura.*

OBDDs

Estes dois BDDs são equivalentes:



Mas só o segundo é ordenado: $[x, y, z]$.

Para o primeiro não é possível encontrar uma ordem: para haver uma ordem não podem haver múltiplas ocorrências duma variável ao longo de um caminho.

Operações booleanas com OBDDs

- Para efectuar operações booleanas com dois OBDDs B_1 e B_2 é necessário que tenham ordem compatíveis: não existirem duas variáveis x e y tal que $x < y$ em B_1 mas $y < x$ em B_2.
- Pela unicidade dos OBDDs reduzidos, é imediato determinar se dois OBDDs

são equivalentes: basta reduzí-los e ver se o OBBD resultante é estruturalmente igual (*forma canónica*).

Importância da forma canónica

Não há variáveis redundantes (das quais a função f não dependa)

Equivalência Permite testar se duas funções f e g são equivalentes desde que tenham OBDDs com ordenações compatíveis.

Validade Se a função f for uma tautologia o OBDD reduzido tem apenas um nó com o valor 1 (B_1).

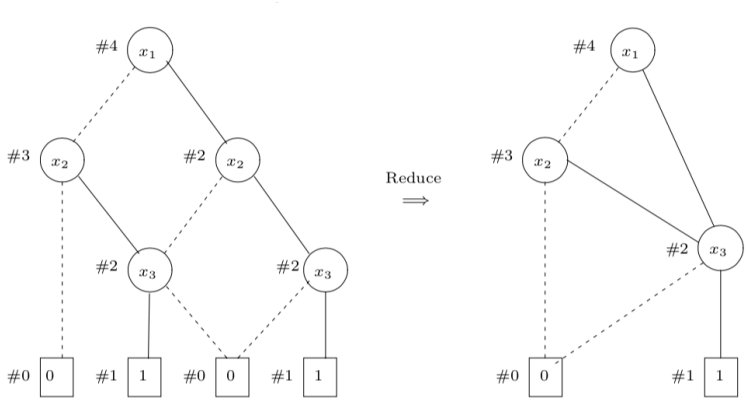
Consequência para saber se g é consequência semântica de f, calcular o OBDD reduzido para $f \cdot \bar{g}$ e determinar se é o OBDD com um só nó 0 (B_0).

Satisfazibilidade Uma função booleana f é satisfazível se e só se o seu OBDD reduzido é diferente de B_0.

Algoritmos para OBDDs:redução

- REDUCE(B_f):
- Se a ordem de B é $[x_1, \ldots, x_l]$ B tem no máximo $l + 1$ níveis
- Começando nos nós terminais (bottom-up), cada nó n recebe uma etiqueta $i(n)$ de forma a que dois nós têm a mesma etiqueta se e só as respectivas sub-OBDDs representarem a mesma função booleana:
  - Atribuir a etiqueta #0 a todas as folhas com o valor 0 e #1 a todas as com o valor 1.
  - Se $i(lo(n)) = i(hi(n))$, então $i(n)$ recebe a mesma etiqueta ($lo(n)$ é o nó abaixo com ligação tracejada e $hi(n)$ o com ligação sólida), i.e. o nó n pode ser eliminado por ser redundante.
  - Se existir outro nó m com a mesma variável x_i, tal que $i(lo(n)) = i(lo(m))$ e $i(hi(n)) = i(hi(m))$, então $i(n) = i(m)$.
  - Se nenhum dos dois casos anteriores se aplicar, atribuí-se a n o próximo inteiro ainda não utilizado.

Algoritmos para OBDDs: `reduce`



Exercício

Para cada uma das funções booleanas indicadas a seguir, determina os OBDD's reduzidos, para as ordens $[x, y, z]$ e $[z, y, x]$ respectivamente. Para isso determina primeiro a árvore de decisão binária correspondente e em seguida aplica o algoritmo REDUCE.

a)

x	y	z	$f(x,y,z)$
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

b) $f(x, y, z) = x \cdot (y + \overline{z})$.

Expansão de Shannon

Definição 17.1. *Seja f uma fórmula Booleana e x uma variável.*

1. *$f[0/x]$ é obtida de f substituindo x por* 0
2. *$f[1/x]$ é obtida de f substituindo x por* 1

Lema 17.1 (Expansão de Shannon)**.**

$$f \equiv \bar{x} \cdot f[0/x] + x \cdot f[1/x]$$

ou, equivalentemente, escreve-se $f = x \to f[1/x], f[0/x]$, o que corresponde a uma instrução IF... THEN... ELSE.

Expansão de Shannon

Para qualquer operador Booleano binário *op*

$$f \ op \ g \quad = \quad \bar{x_i} \cdot (f[0/x_i] \ op \ g[0/x_i]) + x_i \cdot (f[1/x_i] \ op \ g[1/x_i])$$

ou, equivalentemente,

$$x_i \rightarrow f[1/x_i] \ op \ g[1/x_i], f[0/x_i] \ op \ g[0/x_i],$$

Toda a função Booleana pode ser escrita deste modo.

Algoritmos para OBDDs: aplicação

APPLY(op,B_f,B_g): onde OP é uma operação Booleana binária.
Ideia:

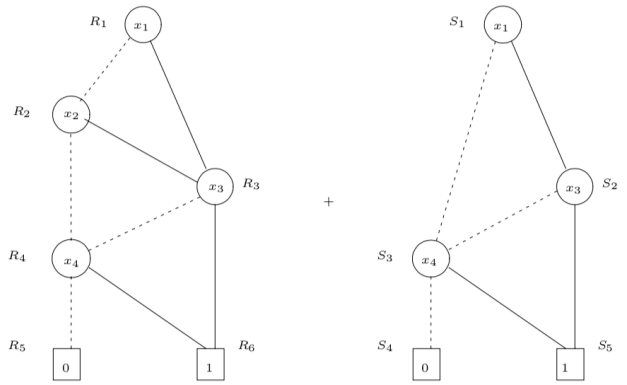
- seja v a variável de maior ordem em B_f ou B_g
- separar o problema em dois supondo v igual a 0 e v igual a 1
- nas folhas aplicar a operação Booleana correspondente

apply

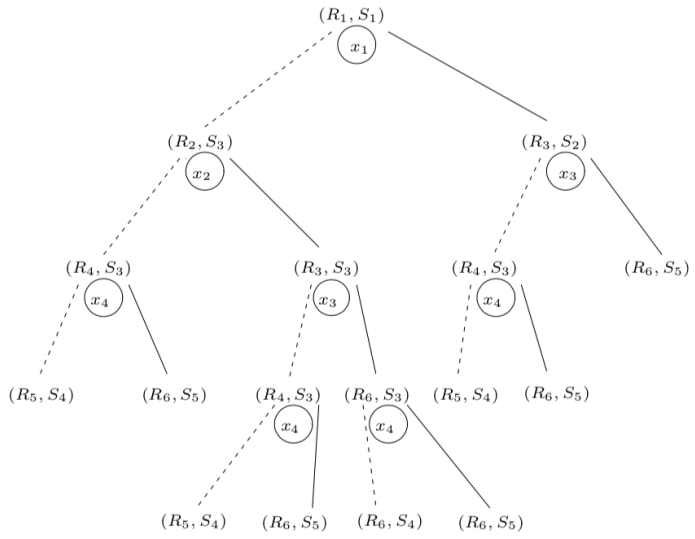
APPLY(op,B_f,B_g):
Sejam r_f e r_g respectivamente as raízes de B_f e B_g. Para calcular o OBDD $B_{f \texttt{op} g}$, que geralmente não está em forma reduzida, aplicar os passos seguintes.

- Se ambos forem folhas com valores l_f e l_g (em $\{0,1\}$), tomar $B_{f \ op \ g} = B_0$ se $l_f \ op \ l_g = 0$ e $B_{f \ op \ g} = B_1$, caso contrário.
- Senão há pelo menos um nó raiz que não é uma folha. Se ambas raízes forem x_i-nós, cria-se um x_i-nó com um arco tracejado para o OBDD APPLY(op,$lo(r_f)$,$lo(r_g)$) e um arco sólido para o OBDD APPLY(op,$hi(r_f)$,$hi(r_g)$). Corresponde à expansão de Shannon.
- Se r_f for um x_i-nó e r_g for uma folha ou um x_j-nó com $j > i$ (numa ordem $[x_1, \ldots, x_n]$), então cria-se um x_i-nó com um arco tracejado para o OBDD APPLY(op,$lo(r_f)$,r_g) e um arco sólido para o OBDD APPLY(op,$hi(r_f)$,r_g). Neste caso g não depende de x_i e $g \equiv g[0/x_i] \equiv g[1/x_i]$.
- O caso simétrico é análogo.

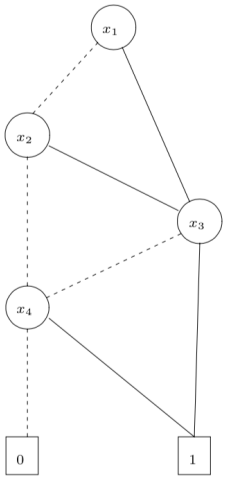
Algoritmos para OBDDs: apply$(+, B_f, B_g)$



Algoritmos para OBDDs: apply



Algoritmos para OBDDs: apply



Algoritmos para OBDDs: restrição $f[val/x]$

RESTRICT(val,x,B_f):

- Para calcular o OBDD $B_{f[0/x]}$ redireccionar os arcos que apontam para um x-nó n para o nó $lo(n)$ e remover o nó n. Reduzir.
- Para calcular o OBDD $B_{f[1/x]}$ redireccionar os arcos que apontam para um x-nó n para o nó $hi(n)$ e remover o nó n. Reduzir.

Algoritmos para OBDDs:exists

$$\begin{aligned} \exists x.f &\equiv f[0/x] + f[1/x] \\ \text{EXISTS}(x, B_f) &= \text{APPLY}(+, B_{f[0/x]}, B_{f[1/x]}) \\ \forall x.f &\equiv f[0/x] \cdot f[1/x] \\ \text{FORALL}(x, B_f) &= \text{APPLY}(\cdot, B_{f[0/x]}, B_{f[1/x]}) \end{aligned}$$

Fórmulas Booleanas e OBDDs

Boolean formula f	OBDD B_f que a representa
0	B_0
1	B_1
x	B_x
$\bar{f}$	trocar os nós 0 e 1 em B_f
$f + g$	APPLY $(+, B_f, B_g)$
$f \cdot g$	APPLY $(\cdot, B_f, B_g)$
$f \oplus g$	APPLY $(\oplus, B_f, B_g)$
$f[1/x]$	RESTRICT $(1, x, B_f)$
$f[0/x]$	RESTRICT $(0, x, B_f)$
$\exists x.f$	APPLY $(+, B_{f[0/x]}, B_{f[1/x]})$
$\forall x.f$	APPLY $(\cdot, B_{f[0/x]}, B_{f[1/x]})$

Complexidade Computacional de OBDDs

Algoritmo	Input OBDD(s)	Output OBDD	Complexidade Temporal
REDUCE	B	reduzido B	$O(\|B\| \cdot \log \|B\|)$
APPLY	B_f, B_g (reduzido)	$B_{f \text{ op } g}$ (reduzido)	$O(\|B_f\| \cdot \|B_g\|)$
RESTRICT	B_f (reduzido)	$B_{f[0/x]}$ or $B_{f[1/x]}$ (reduzido)	$O(\|B_f\| \cdot \log \|B_f\|)$
$\exists$	B_f (reduzido)	$B_{\exists x_1.\exists x_2.\ldots.\exists x_n.f}$ (reduzido)	NP-completo

Exercício

Considera as funções $f(x, y) = x + y$, $g(x, y) = \overline{x} \cdot \overline{y}$ e $h(x, y, z) = x \cdot y + \overline{z} \cdot \overline{x}$.

- Determina os OBDD's reduzidos correspondentes B_f, B_g e B_h para a ordem $[x, y, z]$.
- Determina $B_{\overline{f}}$.
- Determina B_{f+g} aplicando para isso o algoritmo `apply` a B_f e B_g e reduzindo em seguida.
- Determina $B_{\exists yh}$ e $B_{\forall yh}$.

Modelos em OBDDs

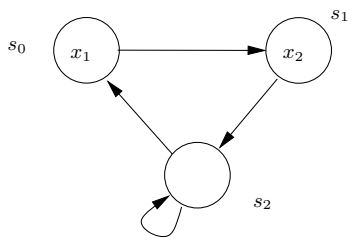
Seja $T = (S, \longrightarrow, I, AP, L)$ um modelo e $|AP| = n$. Cada estado $s \in S$ pode ser representado pelo conjunto de variáveis $L(s)$, i.e. por um tuplo Booleano $(v_1, \ldots, v_n)$ tal que $v_i = 1$ se $x_i \in L(s)$ e $v_i = 0$, caso contrário. Isto obriga que L seja injectiva, o que se pode sempre garantir pela introdução de variáveis proposicionais novas.

Cada estado s pode então ser associado a um OBDD da função Booleana dada por $l_1 \cdot l_2 \cdots l_n$ tal que $l_i = x_i$ se $x_i \in L(s)$ e $\bar{x_i}$, caso contrário (isto é, uma possível valoração dos x_i).

Um conjunto de estados $\{s_1, \ldots, s_m\}$ pode ser representado pelo OBDD da função Booleana:

$$l_{11} \cdot l_{12} \cdot \cdots \cdot l_{1n} + \cdots + l_{m1} \cdot l_{m2} \cdot \cdots \cdot l_{mn}$$

Modelos em OBDDs

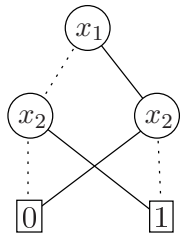


conj. estados	representação por valores booleanos	representação por função booleana
$\emptyset$		0
$\{s_0\}$	$(1,0)$	$x_1 \cdot \bar{x_2}$
$\{s_1\}$	$(0,1)$	$\bar{x_1} \cdot x_2$
$\{s_2\}$	$(0,0)$	$\bar{x_1} \cdot \bar{x_2}$
$\{s_0, s_1\}$	$(1,0)$, $(0,1)$	$x_1 \cdot \bar{x_2} + \bar{x_1} \cdot x_2$
$\{s_0, s_2\}$	$(1,0)$, $(0,0)$	$x_1 \cdot \bar{x_2} + \bar{x_1} \cdot \bar{x_2}$
$\{s_1, s_2\}$	$(0,1)$, $(0,0)$	$\bar{x_1} \cdot x_2 + \bar{x_1} \cdot \bar{x_2}$
S	$(1,0)$, $(0,1)$, $(0,0)$	$x_1 \cdot \bar{x_2} + \bar{x_1} \cdot x_2 + \bar{x_1} \cdot \bar{x_2}$

Representação compacta usando OBDDs

Usando as funções Booleanas dadas os conjuntos de estados podem ser representados compactamente usando OBDDs.

Por exemplo o estado $\{s_0, s_1\}$, pode ser representado por:



E $s \in S'$, se a valoração representada por s satisfaz B_S, ou seja $B_s \cdot B_S$ é satisfazível. No algoritmo de etiquetagem, as operações de conjuntos usadas são

a intersecção, reunião e complemento, todas elas implementáveis como operações de funções Booleanas ($\cdot$, $+$ e $^-$respectivamente) e em OBDDS usando o `apply`.

Modelos em OBDDs

Para a representação da relação de transição $\longrightarrow$, basta ver que é um subconjunto de $S \times S$. Então uma transição $s \longrightarrow s'$ pode ser representada por um par de vectores booleanos $((v_1, \ldots, v_n), (v'_1, \ldots, v'_n))$, ou seja o OBDD da função booleana:

$$(l_1 \cdot l_2 \cdots l_n).(l'_1 \cdot l'_2 \cdots l'_n)$$

tal que l_i é definido como acima e $l'_i = x'_i$ se $x_i \in L(s')$, $l'_i = \bar{x'_i}$, caso contrário. E toda a relação de transição por uma disjunção de destas fórmulas.

$[x_1, x_2, x'_1, x'_2]$

x_1	x_2	x'_1	x'_2	$\rightarrow$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

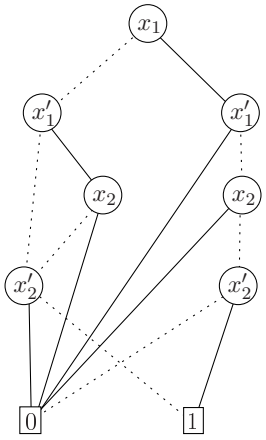
$[x_1, x'_1, x_2, x'_2]$

x_1	x'_1	x_2	x'_2	$\rightarrow$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$$f_{esq}^{\longrightarrow} = \bar{x_1} \cdot \bar{x_2} \cdot \bar{x'_1} \cdot \bar{x'_2} + \bar{x_1} \cdot \bar{x_2} \cdot x'_1 \cdot \bar{x'_2} + x_1 \cdot \bar{x_2} \cdot \bar{x'_1} \cdot x'_2 + \bar{x_1} \cdot x_2 \cdot \bar{x'_1} \cdot \bar{x'_2}$$

Representação das transições em OBDDs

Usando a tabela da direita:



Representação das transições em OBDDs

Os OBDDs para

$$pre_{\exists}(X) = \{s \in S \mid \exists s'\, s \longrightarrow s' \wedge s' \in X\}$$

e

$$pre_{\forall}(X) = \{s \in S \mid \forall s'(s \longrightarrow s') \Rightarrow s' \in X)\}$$

podem ser obtidos a partir de B_X e de $B_{\longrightarrow}$. Como $pre_{\forall}(X) = S \setminus pre_{\exists}(S \setminus X)$, bas considerar o primeiro. Para $pre_{\exists}(X)$:

- considerar $B_{X'}$;
- determinar `exists(`$\vec{x}'$`,apply(`$\cdot$`,`$B_{\longrightarrow}$`,`$B_{X'}$`))`.

Onde $\exists \vec{x} f$ representa $\exists x_1 \ldots \exists x_n f$, para $\vec{x} = (x_1, \ldots, x_n)$. Quer dizer que existe uma valoração para $\vec{x}$ que torna f verdade. Uma valoração corresponde a um estado s' $(l'_1 \cdot l'_2 \cdots l'_n)$ tal que $B_{\longrightarrow} \cdot B_{X'}$ representa o conjunto de estado s $(l_1 \cdot l_2 \cdots l_n)$ tal que $(l_1 \cdot l_2 \cdots l_n) \cdot (l'_1 \cdot l'_2 \cdots l'_n)$ é verdade.

Exercício 17.1. *Considera o modelo* $\mathcal{M} = (S = \{s_0, s_1, s_2, s_3\}, \{s_0 \to s_2, s_0 \to s_1, s_1 \to s_1, s_1 \to s_2, s_1 \to s_3, s_2 \to s_0, s_2 \to s_1, s_2 \to s_2, s_3 \to s_0, s_3 \to s_3\}, L(s_0) = \{x_1, x_2\}, L(s_1) = \{x_1\}, L(s_2) = \{\}, L(s_3) = \{x_2\})$.

a) Utilizando a ordem $[x_1, x_2]$*, determina OBDD's para representar os conjuntos de estados* $\{s_0, s_1\}$ *e* $\{s_0, s_2\}$.

b) Determina a tabela de verdade para a relação de transição utilizando a ordem $[x_1, x'_1, x_2, x'_2]$.

c) Desenha o OBDD para a relação de transição (utilizando a ordem da alínea anterior).

d) Aplica o algoritmo de etiquetagem (adaptado à representação por OBDD's e utilizando a ordem $[x_1, x_2]$) ao modelo $\mathcal{M}$, para determinar os conjuntos de estados onde se verificam respectivamente as fórmulas seguintes.

- $EX\ x_2$;
- $AG\ (x_1 \vee x_2)$;
- $E\ (x_2\ U\ x_1)$.

$\diamond$