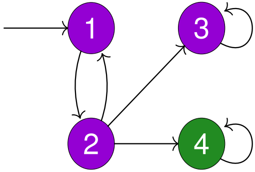
Binary Decision Diagrams

EECS 4315

www.cse.yorku.ca/course/4315/

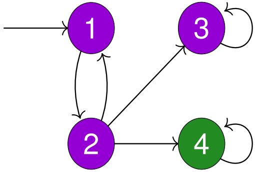
Representation States



Question

How many Boolean variables do we need to encode the states?

Representation States



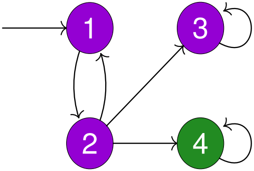
Question

How many Boolean variables do we need to encode the states?

Answer

Two.

Representing Labelling

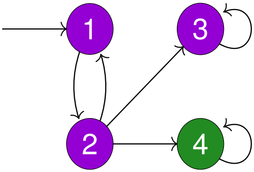


state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

Question

Which Boolean formula represents the CTL formula purple?

Representing Labelling



state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

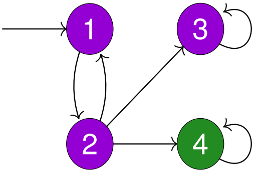
Question

Which Boolean formula represents the CTL formula purple?

Answer

$[\text{purple}] = \neg x_1 \vee \neg x_2$.

Representing Labelling

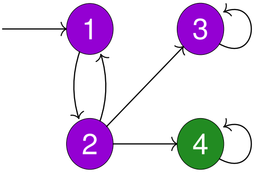


state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

Question

Which Boolean formula represents the CTL formula green?

Representing Labelling



state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

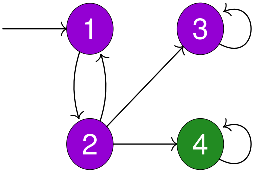
Question

Which Boolean formula represents the CTL formula green?

Answer

$[\text{green}] = x_1 \wedge x_2$.

Representing Transition Relation



state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

Question

Which Boolean formula represents the transition relation? (Represent the source by x_1 and x_2 and represent the target by x_1' and x_2'.)

Representing Transition Relation

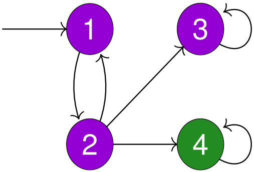
$$
\begin{array}{rl}
(\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \vee & 1 \to 2 \\
(\neg x_1 \wedge x_2 \wedge \neg x_1' \wedge \neg x_2') \vee & 2 \to 1 \\
(\neg x_1 \wedge x_2 \wedge x_1' \wedge \neg x_2') \vee & 2 \to 3 \\
(\neg x_1 \wedge x_2 \wedge x_1' \wedge x_2') \vee & 2 \to 4 \\
(x_1 \wedge \neg x_2 \wedge x_1' \wedge \neg x_2') \vee & 3 \to 3 \\
(x_1 \wedge x_2 \wedge x_1' \wedge x_2') & 4 \to 4
\end{array}
$$

which is equivalent to

$$
\begin{array}{r}
(\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \vee \\
(\neg x_1 \wedge x_2 \wedge \neg x_2') \vee \\
(x_2 \wedge x_1' \wedge x_2') \vee \\
(x_1 \wedge \neg x_2 \wedge x_1' \wedge \neg x_2')
\end{array}
$$

which is denoted by $[\to]$.

Representing Initial States

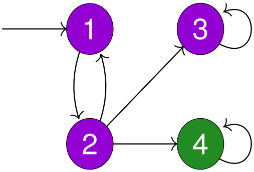


state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

Question

Which Boolean formula represents the set of initial states?

Representing Initial States



state	x_1	x_2
1	0	0
2	0	1
3	1	0
4	1	1

Question

Which Boolean formula represents the set of initial states?

Answer

$[I] = \neg x_1 \wedge \neg x_2$.

Review: Computing $Sat(\exists(\Phi \text{ U } \Psi))$

The function $F : 2^S \to 2^S$ is defined by

$$F(T) = Sat(\Psi) \cup \{ s \in Sat(\Phi) \mid Post(s) \cap T \neq \emptyset \}$$

where

$$Post(s) = \{ s' \in S \mid s \to s' \}.$$

```
T = ∅
while T ≠ F(T)
    T = F(T)
return T
```

Representing $Sat(\exists(\Phi \text{ U } \Psi))$

$$[F](T)(\vec{x}) = [\Psi](\vec{x}) \vee \exists \vec{x}' [\rightarrow](\vec{x}, \vec{x}') \wedge [\Phi](\vec{x}) \wedge T(\vec{x}')$$

$T = 0$
while $T \neq [F](T)$
  $T = [F](T)$
return T

Data structures for BDDs

The nodes are represented as integers 0, 1, 2, ... where 0 and 1 represent the leaves labelled 0 and 1.

Given a variable ordering $x_1 < x_2 < \cdots < x_n$, the variables are represented by their indices 0, 1, ..., n.

Node table

The *node table* can be viewed as a partial function

$$T : \mathbb{N} \to (\mathbb{N}^3 \cup \mathbb{N})$$

which maps the index of a node to the indices of its variable, low- and high-successor.

$$u \mapsto (v, \ell, h)$$

Note that 0 and 1 do not have a low- and high-successor. These external vertices are assigned a variable index which is $n + 1$, where n is the number of variables. (This choice simplifies some of the algorithms to be discussed later.)

Operations on node table

init(T): initializes T to contain only nodes 0 and 1.

u	$var(u)$	$low(u)$	$high(u)$
0	$n+1$		
1	$n+1$		

Operations on node table

$u \leftarrow add(T, i, \ell, h)$: allocate a new node u with attributes (i, ℓ, h).

Question

Given the node table

u	$var(u)$	$low(u)$	$high(u)$
0	$n+1$		
1	$n+1$		

what does the operation $add(T, 4, 1, 0)$ return?

Operations on node table

$u \leftarrow add(T, i, \ell, h)$: allocate a new node u with attributes (i, ℓ, h).

Question

Given the node table

u	$var(u)$	$low(u)$	$high(u)$
0	$n+1$		
1	$n+1$		

what does the operation $add(T, 4, 1, 0)$ return?

Answer

2.

Operations on node table

$u \leftarrow add(T, i, \ell, h)$: allocate a new node u with attributes (i, ℓ, h).

Question

Given the operation $add(T, 4, 1, 0)$ applied to the node table

u	$var(u)$	$low(u)$	$high(u)$
0	5		
1	5		

what is the resulting node table?

Operations on node table

$u \leftarrow add(T, i, \ell, h)$: allocate a new node u with attributes (i, ℓ, h).

Question

Given the operation $add(T, 4, 1, 0)$ applied to the node table

u	$var(u)$	$low(u)$	$high(u)$
0	5		
1	5		

what is the resulting node table?

Answer

u	$var(u)$	$low(u)$	$high(u)$
0	5		
1	5		
2	4	1	0

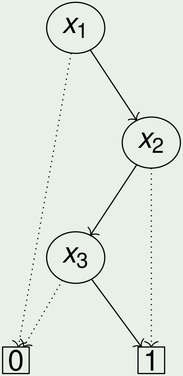
Operations on node table

$var(u)$	:	look up the var attribute of u in T
$low(u)$	:	look up the low attribute of u in T
$high(u)$	:	look up the high attribute of u in T

Example of node table

Question

Give the node table corresponding to the BDD



Example of node table

Answer

u	$var(u)$	$low(u)$	$high(u)$
0	4		
1	4		
2	3	0	1
3	2	1	2
4	1	0	3

Inverse of node table

The *inverse of the node table* can be viewed as a partial function

$$H : \mathbb{N}^3 \rightarrow \mathbb{N}$$

which maps the indices of the attributes of a node to the index of the node.

$$(v, \ell, h) \mapsto u$$

For all $u \geq 2$,

$$T(u) = (i, \ell, h) \text{ iff } H(i, \ell, h) = u.$$

Operations on inverse of node table

$$
\begin{aligned}
\mathit{init}(H) &: \text{initializes } H \text{ to be empty} \\
b \leftarrow \mathit{member}(H, i, \ell, h) &: \text{check if } (i, \ell, h) \text{ is in } H \\
u \leftarrow \mathit{lookup}(H, i, \ell, h) &: \text{find } H(i, \ell, h) \\
\mathit{insert}(H, i, \ell, h, u) &: \text{make } (i, \ell, h) \text{ map to } u \text{ in } H
\end{aligned}
$$

Operations on BDDs

Question

Consider the node table T and its inverse H.

- Let ℓ and h be indices of nodes u_ℓ and u_h.
- Let i be the index of variable x_i.[a]

Return the index of the node of T corresponding to $x_i \to u_h, u_\ell$ and expand T and H if needed.

[a] In the variable ordering, this variable occurs before all variables occurring in the subgraphs rooted at ℓ and h.

Operations on BDDs

```
MK[T, H](i, ℓ, h)
   if ℓ = h then
      return ℓ
   else if member(H, i, ℓ, h) then
           return lookup(H, i, ℓ, h)
   else
      u ← add(T, i, ℓ, h)
      insert(H, i, ℓ, h, u)
      return u
```

Operations on BDDs

Question

Consider the node table T and its inverse H. Let t be a Boolean expression. Return the node of T corresponding to t.

Operations on BDDs

```
BUILD[T, H](t)
   return build(t, 1)

function build(t, i)
   if i > n then
      if t is false then return 0 else return 1
   else
      u0 ← build(t[0/xi], i + 1)
      u1 ← build(t[1/xi], i + 1)
      return MK(i, u0, u1)
```

Proposition

For all binary Boolean operators $\otimes$,

$$(x \to t_1, t_0) \otimes (x \to u_1, u_0) = x \to t_1 \otimes u_1, t_0 \otimes u_0.$$

Question

Consider the node table T and its inverse H.

- Let u_1 and u_2 be indices of nodes.
- Let $\oplus$ be a binary Boolean operator.

Return the index of the node of T corresponding to $u_1 \oplus u_2$ and expand T and H if needed.

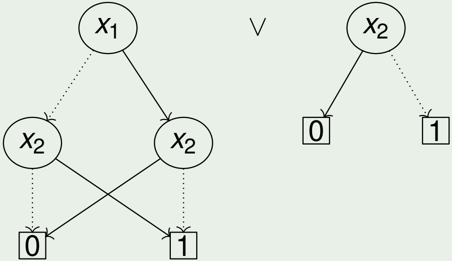
$\text{APPLY}[T, H](\oplus, u_1, u_2)$
  return $app(u_1, u_2)$

function $app(u_1, u_2)$
  if $u_1 \in \{0, 1\}$ **and** $u_2 \in \{0, 1\}$ **then**
    $u \leftarrow u_1 \oplus u_2$
  else if $var(u_1) = var(u_2)$ **then**
    $u \leftarrow \text{MK}(var(u_1), app(low(u_1), low(u_2)), app(high(u_1), high(u_2)))$
  else if $var(u_1) < var(u_2)$ **then**
    $u \leftarrow \text{MK}(var(u_1), app(low(u_1), u_2), app(high(u_1), u_2))$
  else
    $u \leftarrow \text{MK}(var(u_2), app(u_1, low(u_2)), app(u_1, high(u_2)))$
  return u

Example of apply

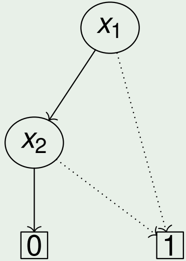
Question

What is the result of



Example of apply

Answer



Question

Consider the node table T and its inverse H.

- Let u be the index of a node.
- Let j be the index of a variable.
- Let $b \in \{0, 1\}$.

Return the index of the node of T corresponding to $u[b/x_j]$ and expand T and H if needed.

```
RESTRICT[T, H](u, j, b)
   return res(u)

function res(u)
   if var(u) > j then
      return u
   else if var(u) < j then
      return MK(var(u), res(low(u)), res(high(u)))
   else if b = 0 then
      return low(u)
   else
      return high(u)
```

Operations on BDDs

Question

Consider the node table T and its inverse H.

- Let u be the index of a node.
- Let j be the index of a variable.

Return the index of the node of T corresponding to $\exists x_j : u$ and expand T and H if needed. You may use operations that we have already defined.

Operations on BDDs

Answer

$u_0 \leftarrow \textsc{Restrict}[T, H](u, j, 0)$
$u_1 \leftarrow \textsc{Restrict}[T, H](u, j, 1)$
return $\textsc{Apply}[T, H](\vee, u_0, u_1)$

Implementing T and H

T : dynamic array
H : hash table