Preliminaries

p holds at the even states and does not hold at the odd states

p & G (p <-> ¬ (X p))

This is syntactically a CTL* path formula, but it’s meaning is actually

A (p & G (p <-> ¬ (X p)))

(A means ”all paths”, not ”always”.)

Model Checking II

How CTL model checking works

CTL

A E   X F G U

Model checking problem

Determine $M, s0 \models f$

Or find all s s.t. $M, s \models f$

Model checking

Last lecture – semantics of CTL*: M, s0 $\models$ f

- Impractical as algorithm (A,E require exploring an infinite set of paths; F,U require searching indefinitely for future time-point)

This lecture – alternative semantics (of CTL):

$H(f) = \{s \mid M, s \models f\}$ ”set of states for which f holds”

- Easy to turn into a practical algorithm!

Explicit state model checking

Option 1

Represent state transition graph explicitly

Walk around marking states

Graph algorithms involving strongly connected components etc.

Not covered in this course (cf. SPIN)

Used particularly in software model checking

Symbolic MC

Option 2 McMillan et al

because of

STATE EXPLOSION problem

- State graph exponential in program/circuit size
- Graph algorithms linear in state graph size

INSTEAD

- Use symbolic representation both of sets of states and of state transtion graph

CTL

Need only the boolean connectives (**¬** , **&**) and

A X F G U

(different choice from yesterday to follow Seger paper more closely)

Define others

e.g.

$EG\ p \Leftrightarrow \neg AF \neg p$

$E(p\ U\ q) \Leftrightarrow \neg (A(\neg q\ U\ (\neg p\ \&\ \neg q)) \vee AG(\neg q))$

Set of states in which a formula holds

CTL formula f	H(f) set of states satisfying f
a (atomic)	{s | a in L(s)} (cf.Lars)

Set of states in which a formula holds

CTL formula f	H(f) set of states satisfying f
a (atomic)	{s | a in L(s)} (cf.Lars)
¬p	S – H(p)

Set of states in which a formula holds

CTL formula f	H(f) set of states satisfying f
a (atomic)	{s | a in L(s)} (cf.Lars)
¬p	S – H(p)
p **&** q	H(p) ∩ H(q)

Set of states in which a formula holds

CTL formula f | H(f) set of states satisfying f

AX f | {s | forall t sRt => t $\in$ H(f)}

Now gets harder

AG p ⇔ p & AX AG p

Recursive

Want to write something like

H(AG p) = H(p) ∩ {s | forall t sRt => t ∈ H(AG p)}

How to solve this equation?

want to find a set U such that

$U = H(p) \cap \{s \mid \text{forall } t \ sRt \Rightarrow t \in U\}$

form is

$U = f(U)$

We need to compute a fixed point (or fixpoint) of function f

Fixed points (Tarski)

(Normally expressed in terms of general lattices; here only considering the special case of sets.)

Let f be a monotonic function on sets ($x \subseteq f(x)$ or $x \supseteq f(x)$)

Then there will be a least fixed point Lfp U. f(U)

or a greatest fixed point Gfp U. f(U)

Lfp for increasing sets and Gfp for decreasing sets

Next question

Do we need a least or a greatest fixed point for

$U = H(p) \cap \{s \mid \text{forall } t\ sRt \Rightarrow t \in U\}$

?

Answer is Gfp

Idea: start with S (entire set of states) as first approx.

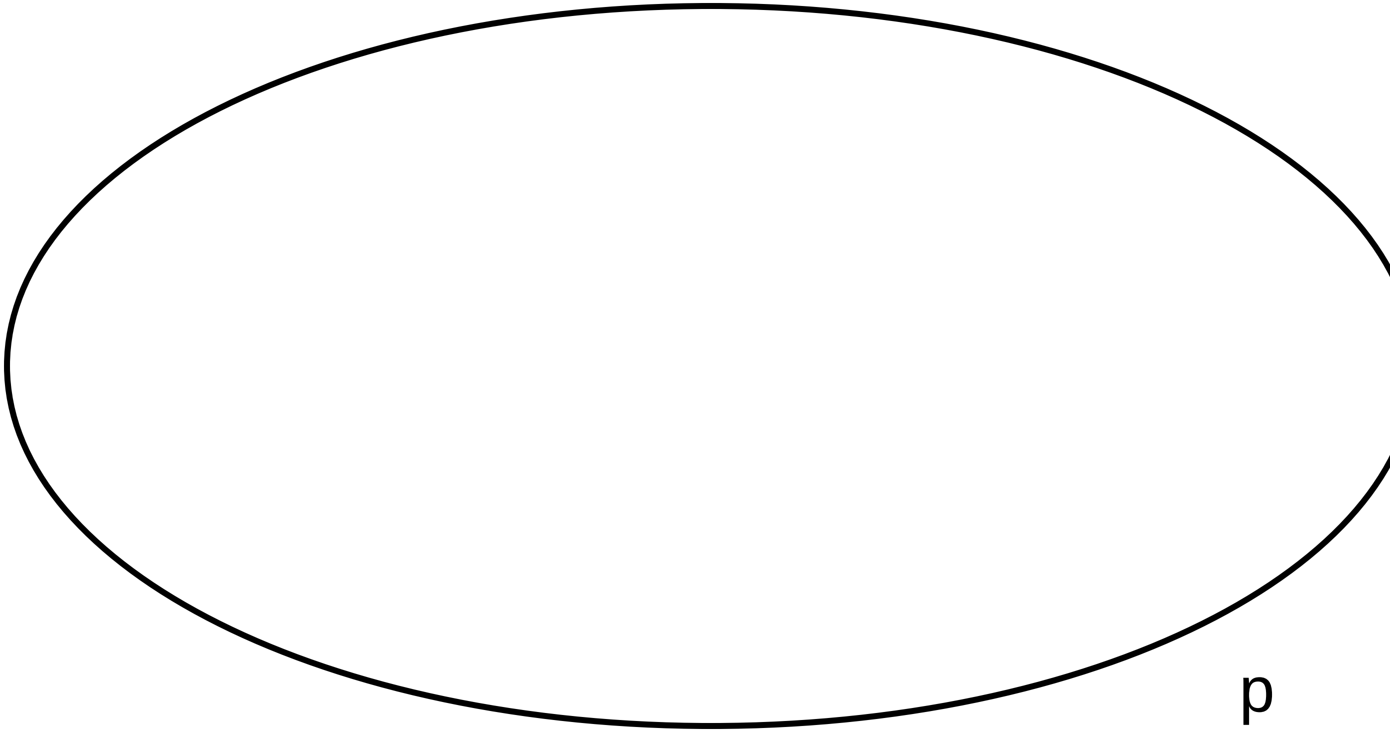
Then compute f(S), f (f (S)) until no change in set

Conclusion

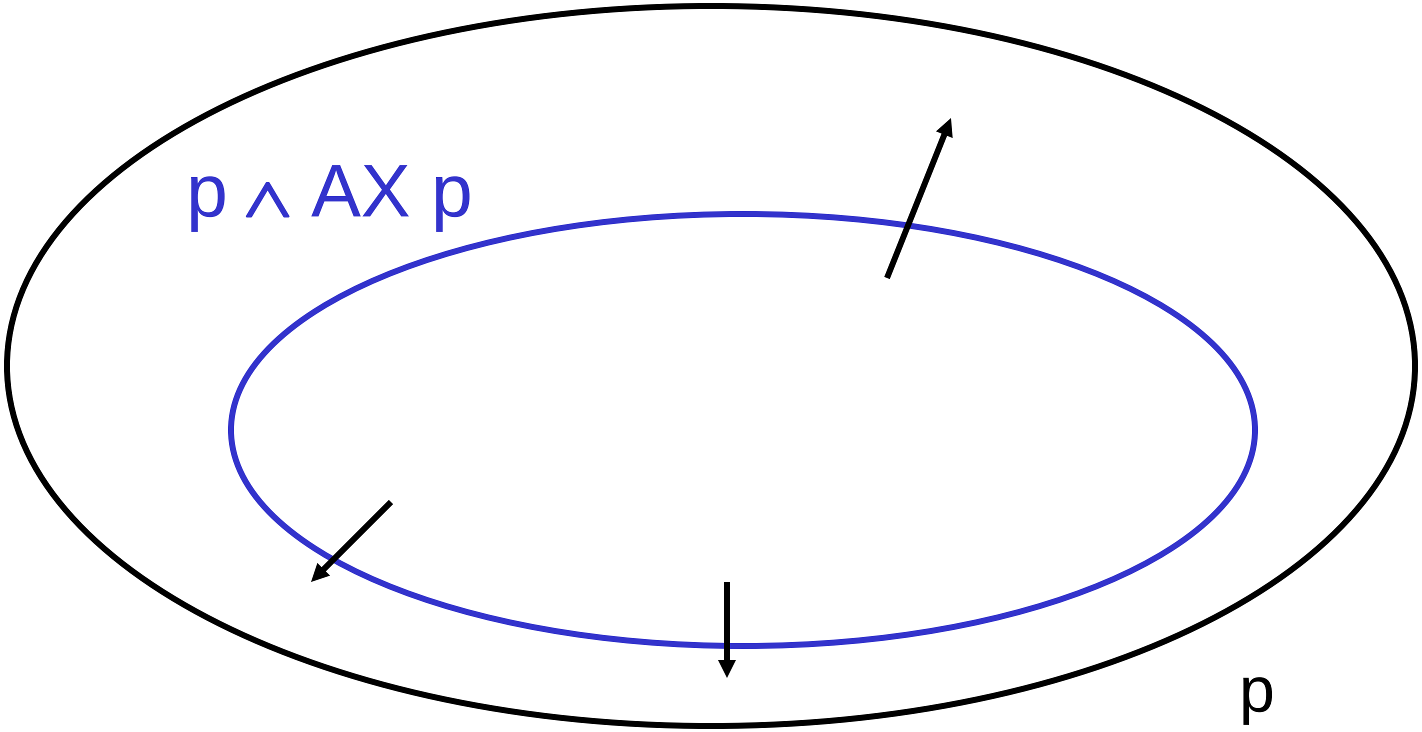
H(AG p)

= Gfp U . H(p) ∩ {s | forall t sRt => t ∈ U}

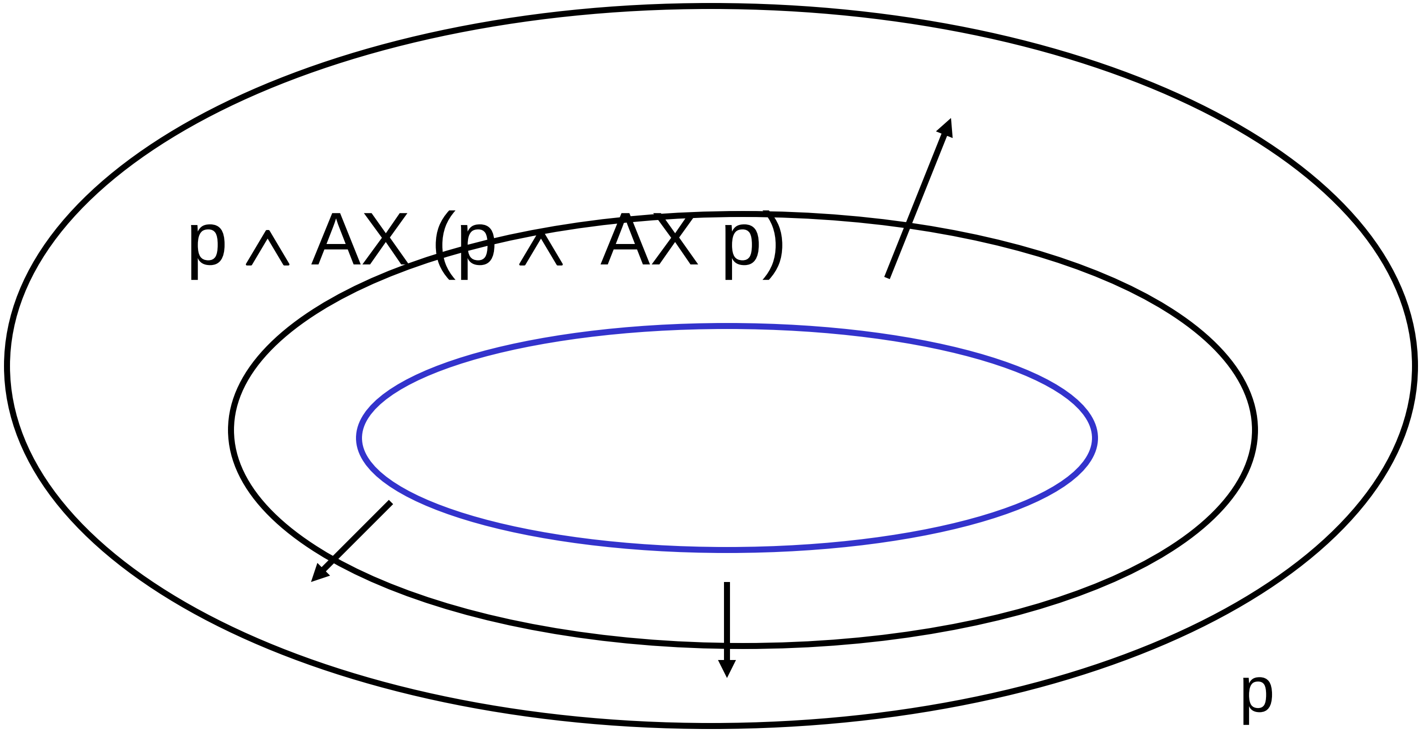
Fixed point iteration



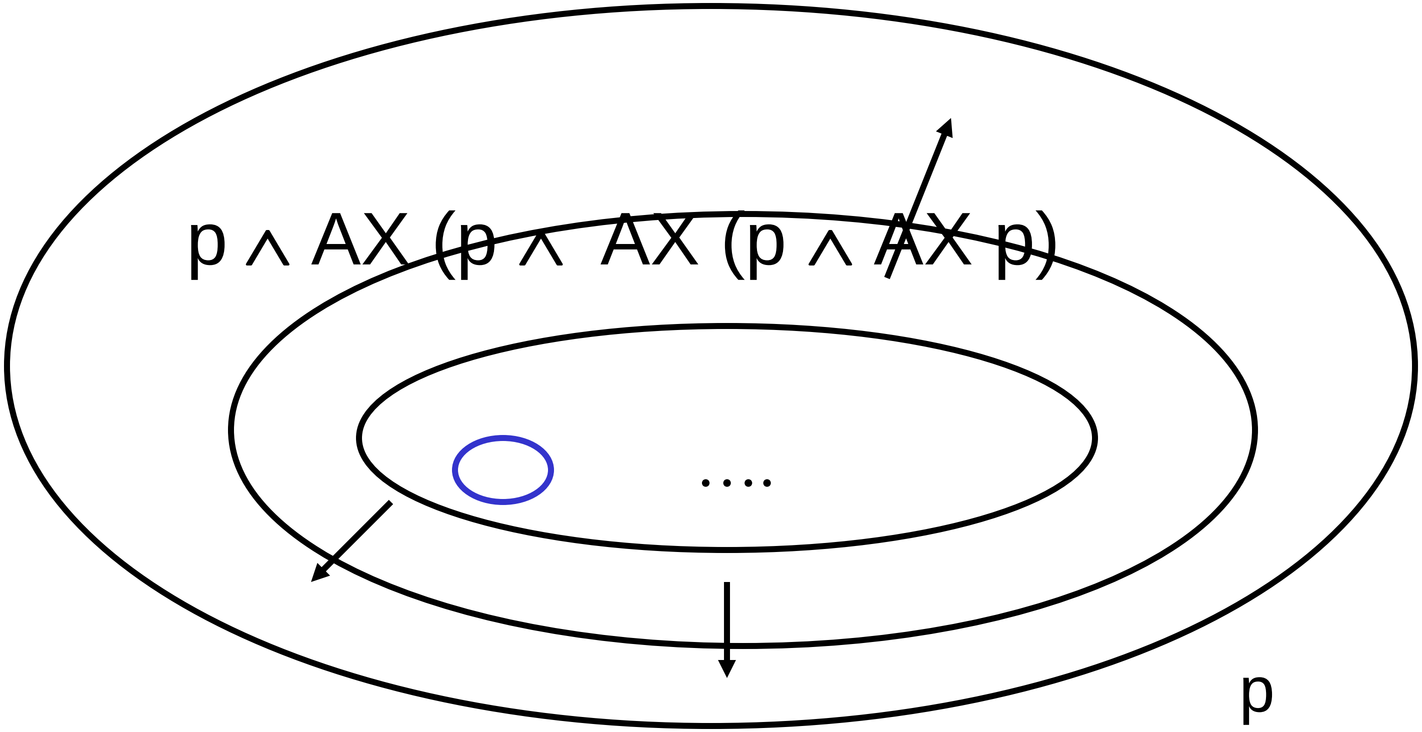
Fixed point iteration



Fixed point interation in the other direction



Fixed point iteration



AF

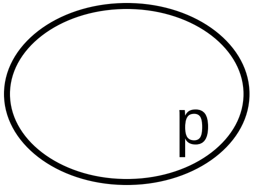
AF p ⇔ p $\vee$ AX AF P

Same kind of pattern but this time need least fixed point (starting with empty set)

H(AF p)

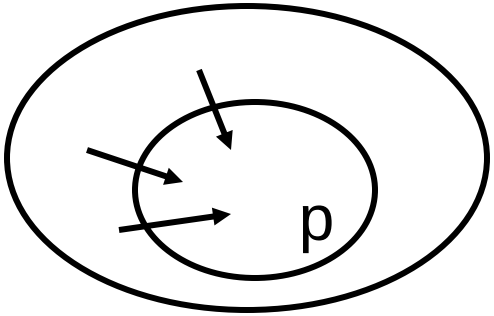
= Lfp U. H(p) $\cup$ {s | forall t sRt => t $\in$ U}

Fixed point iteration



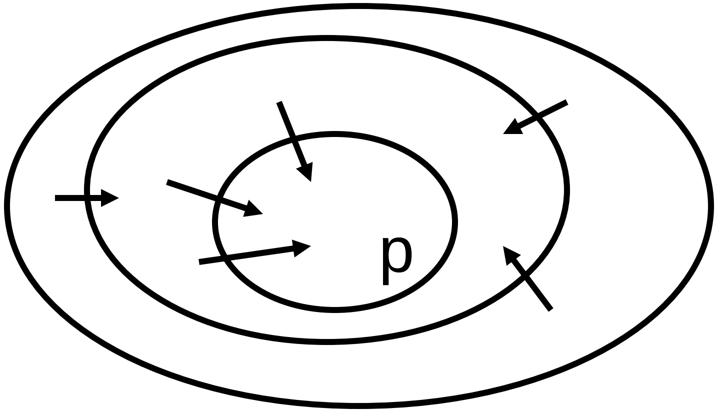
Fixed point iteration

$p \vee AX\, p$



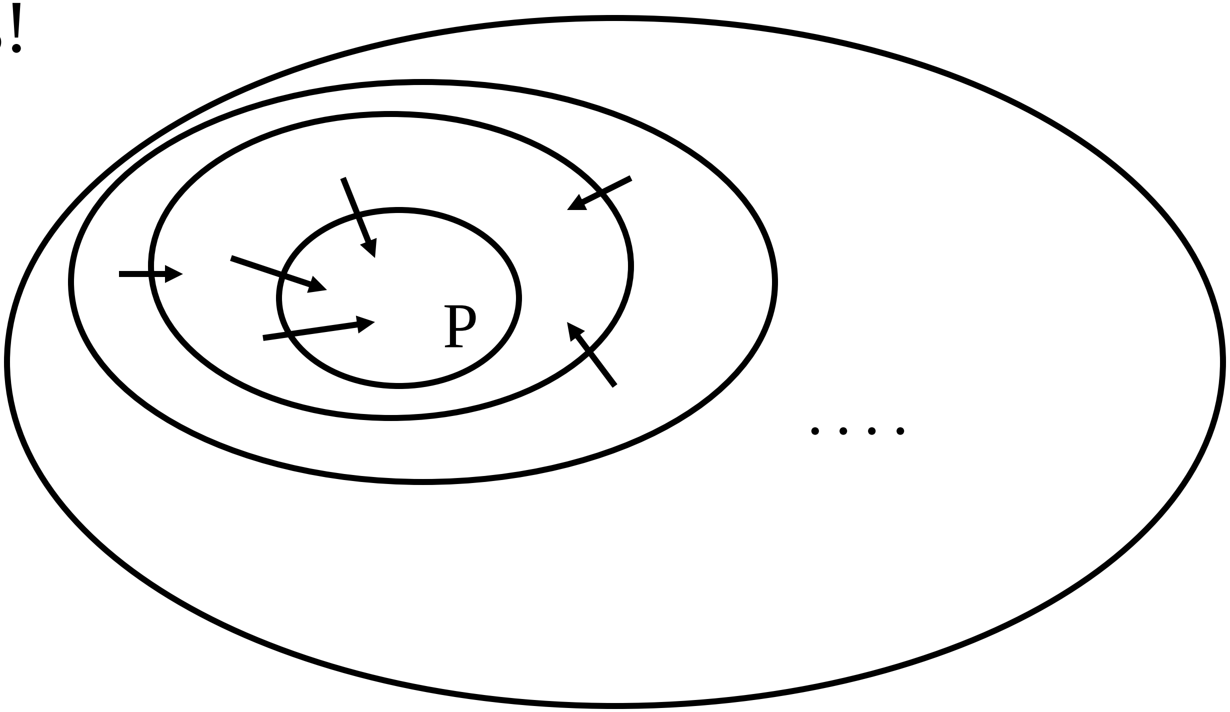
Fixed point iteration

p ∨ AX (p ∨ AX p)



Fixed point iteration

Evetually stops!



Similar story for Until

$A\,(p\ U\ q) \Leftrightarrow q \vee (p \wedge AX\,(A\,(p\ U\ q)))$

$H(A\,(p\ U\ q))$

$= \text{Lfp}\ \boldsymbol{U}.\ H(q) \cup (H(q) \cap \{s \mid \text{forall}\ t\ sRt \Rightarrow t \in \boldsymbol{U}\})$

.

Rest are defined in terms of these

e.g.

EG p $\Leftrightarrow$ $\neg$ AF $\neg$p

E(p U q) $\Leftrightarrow$ $\neg$ (A($\neg$ q U $\neg$ p & $\neg$ q) $\vee$ AG($\neg$ q))

Put H around each side

So far so good

Only talked about sets of states so far

Will come back to concrete calculations with these

What about BDDs to represent them??

BDD based Symbolic MC

Sets of states

relations between states 〉 BDDs

Fixed point characterisations of CTL ops

NO explicit state graph

Boolean formulas

$(x \oplus y) \oplus z$ (⊕ is exclusive or)

$(1 \oplus 0) \oplus 0 \quad = \quad 1$

assignment [x=1,y=0,z=0] gives answer 1

is a model or satisfying assignment

Write as 100

Exercise: Find another model

Boolean formulas

$(x \oplus y) \oplus z$

$(1 \oplus 1) \oplus 0 \quad = \quad 0$

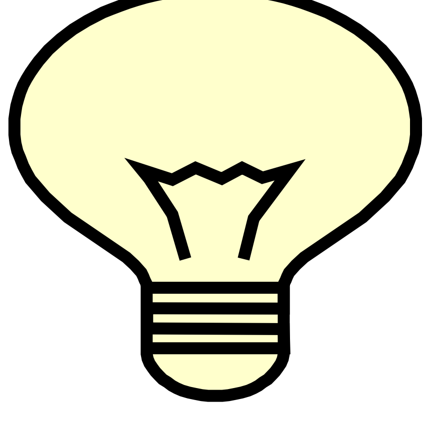
assignment [x=1,y=1,z=0] is not a model

Formula is a tautology if ALL assignments are models and is contradictory if NONE is.

Boolean formulas

For us, interesting formulas are somewhere in between: some assignments are models, some not

IDEA: A formula can represent a set of states (its models)



{}	false
{111}	$x \wedge y \wedge z$
{101}	$x \wedge \neg y \wedge z$
{111,101}	$x \wedge z$
.	
.	
{000,001, … , 111}	true

Example

$(x \oplus y) \oplus z$ represents {100,010,001,111} for states of the form xyz

Exercise: Find formulas (with var. names x,y,z) for the sets

{}

{100}

{110,100,010,000}

What is needed now?

A good data structure for boolean formulas

Have already seen

Binary Decision Diagrams (BDDs)

- Bryant (IEEE Trans. Comp. 86, most cited CS paper!)
  - see also Bryant's document about a Hitachi patent from 93
- McMillan saw application to symbolic MC

A state

Vector of boolean variables

$(v1,v2,v3, \ldots, vn) \in \{0,1\}^n$

Represent a set of states

Just make the BDD for a corresponding formula!

BDD for set P using state variable vector v:

bdd(P,v)

Represent a transition relation R

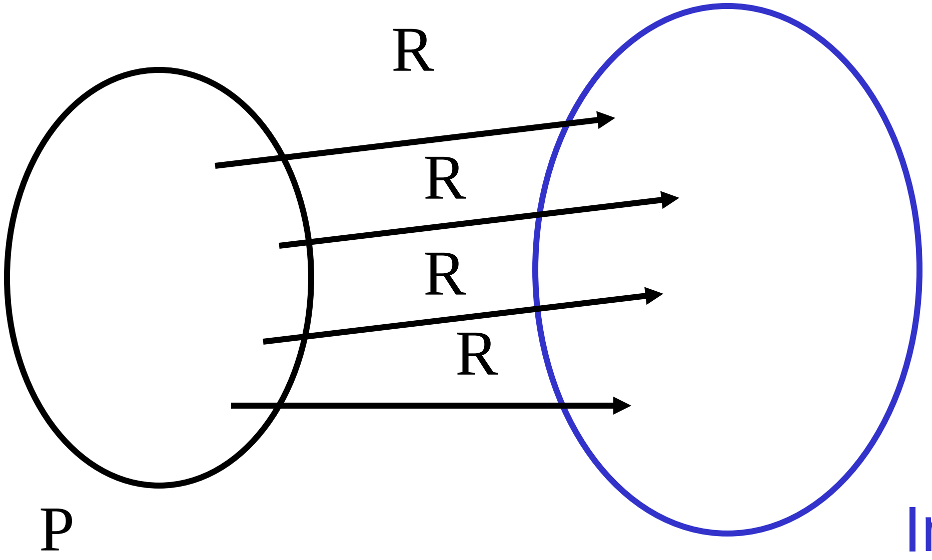
Remember that R is just

a set of pairs of states

Use two variable vectors, v and v' (with the primed variables representing next states)

Make a formula involving both v and v' and from that a BDD   bdd(R,(v,v'))

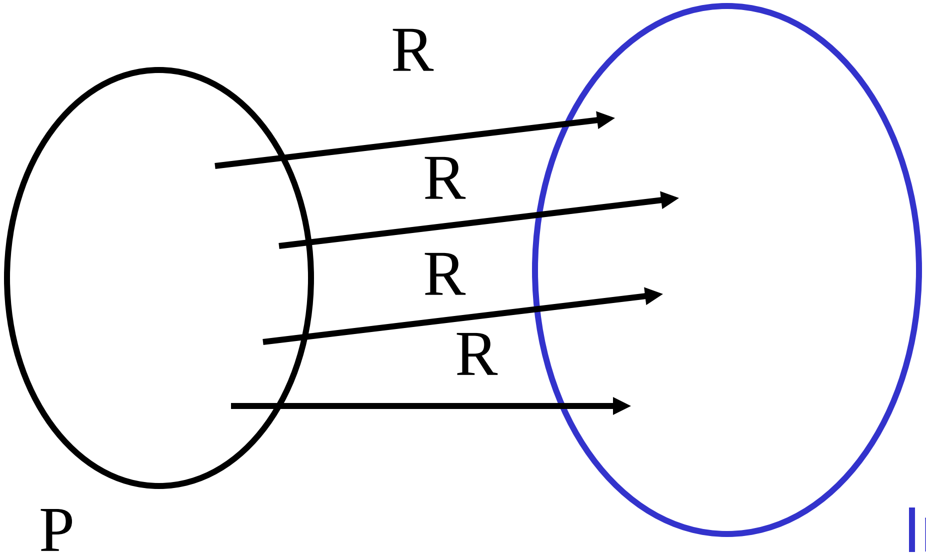
What set of states can we reach from set P in one step?



Image(P,R)

$\{t \mid \exists s \;\; s \in P \wedge s \, R \, t\}$

What set of states can we reach from set P in one step?

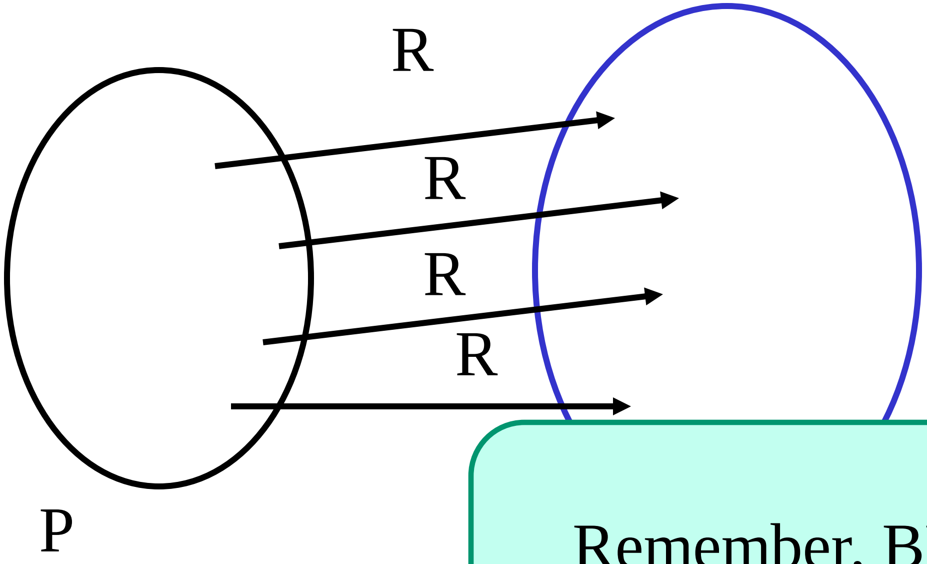


Image(P,R)

$\{t \mid \exists s \;\; s \in P \wedge s \; R \; t\}$

bdd(Image(P,R),v') $=$ $\exists$ v bdd(P,v) $\wedge$ bdd(R,(v,v'))

What set of states can we reach from set P in one step?



Remember, BDDs support quantification

,R)

s ∈ P ∧ s R t}

bdd(Image(P,R),v') = ∃ v bdd(P,v) ∧ bdd(R,(v,v'))

So far

BDDs for

1) sets of states
2) transition relation
3) calculating forward image of a set

Before we go on with MC, note that we can now compute Reachable States (see Hu paper)

Let T be the transition relation

$R_0(v)$ = BDD for reset (or initial) state

$R_1(v)$ = $R_0(v) \vee$ bdd(Image(R_0,T),v)

…

$R_{i+1}(v)$ = $R_i(v) \vee$ bdd(Image(R_i,T),v)

Will eventually converge with $R_{i+1}(v) = R_i(v)$.

Why???

Before we go on with MC, note that we can now compute Reachable States (see Hu paper)

Let T be the transition relation

$R_0(v)$ = BDD for reset (or initial) state

$R_1(v)$ = $R_0(v) \vee$ bdd(Image(R_0,T),v)

BDD or

…

$R_{i+1}(v)$ = $R_i(v) \vee$ bd

Will eventually converge with $R_{i+1}(v) = R_i(v)$.

Why???

Before we go on with MC, note that we can now compute Reachable States (see Hu paper)

Let T be the transition relation

$R_0(v)$ = BDD for reset (or initial) state

$R_1(v)$ = $R_0(v) \vee$ bdd(Image(R_0,T),v)

Easy to check. Why?

…

$R_{i+1}(v)$ = $R_i(v) \vee$ bdd(Image(R_i,T),v)

Will eventually converge with $R_{i+1}(v) = R_i(v)$.

Back to MC

CTL formula f	H(f) set of states satisfying f
a (atomic)	{s | a in L(s)} (cf.Lars)
¬p	S – H(p)
p **&** q	H(p) ∩ H(q)

CTL formula f | H(f) set of states satisfying f

AX f | {s | forall t sRt => t ∈ H(f)}

All of the above operations easy to do with BDDs

BDDs also fine in fixed point iterations

H(AF p)

= Lfp U. H(p) $\cup$ {s | forall t sRt => t $\in$ U}

becomes

U_0 = empty set

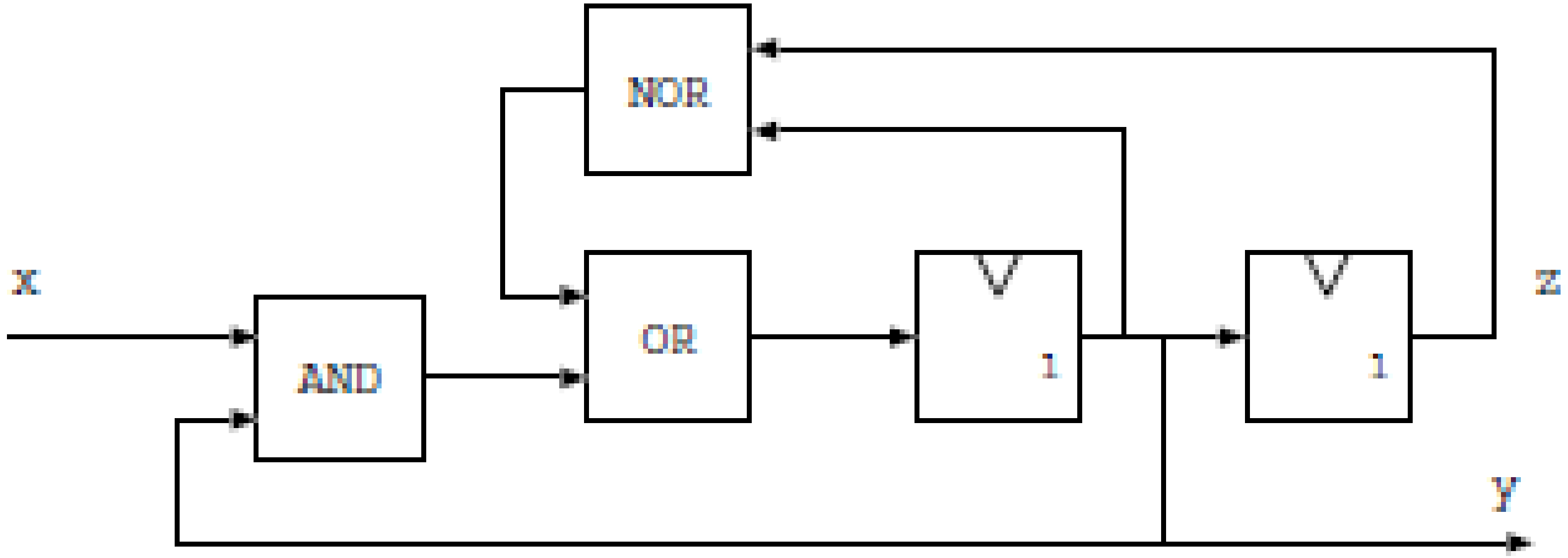
U_1 = H(p) $\cup$ {s | forall t sRt => t $\in$ U_0}

U_2 = H(p) $\cup$ {s | forall t sRt => t $\in$ U_1}

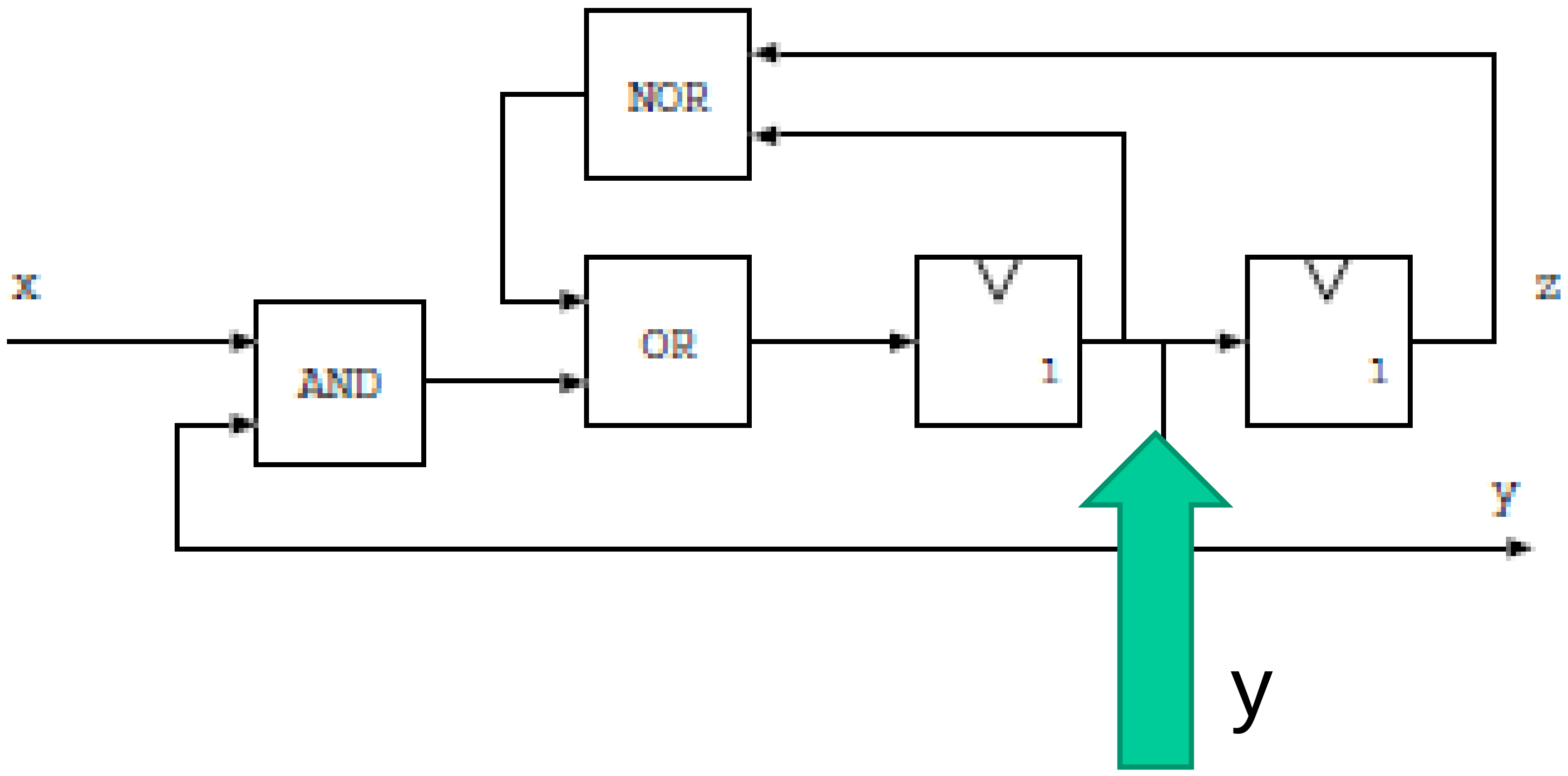
…

All done with BDDS (and recursion and fixed point iteration)

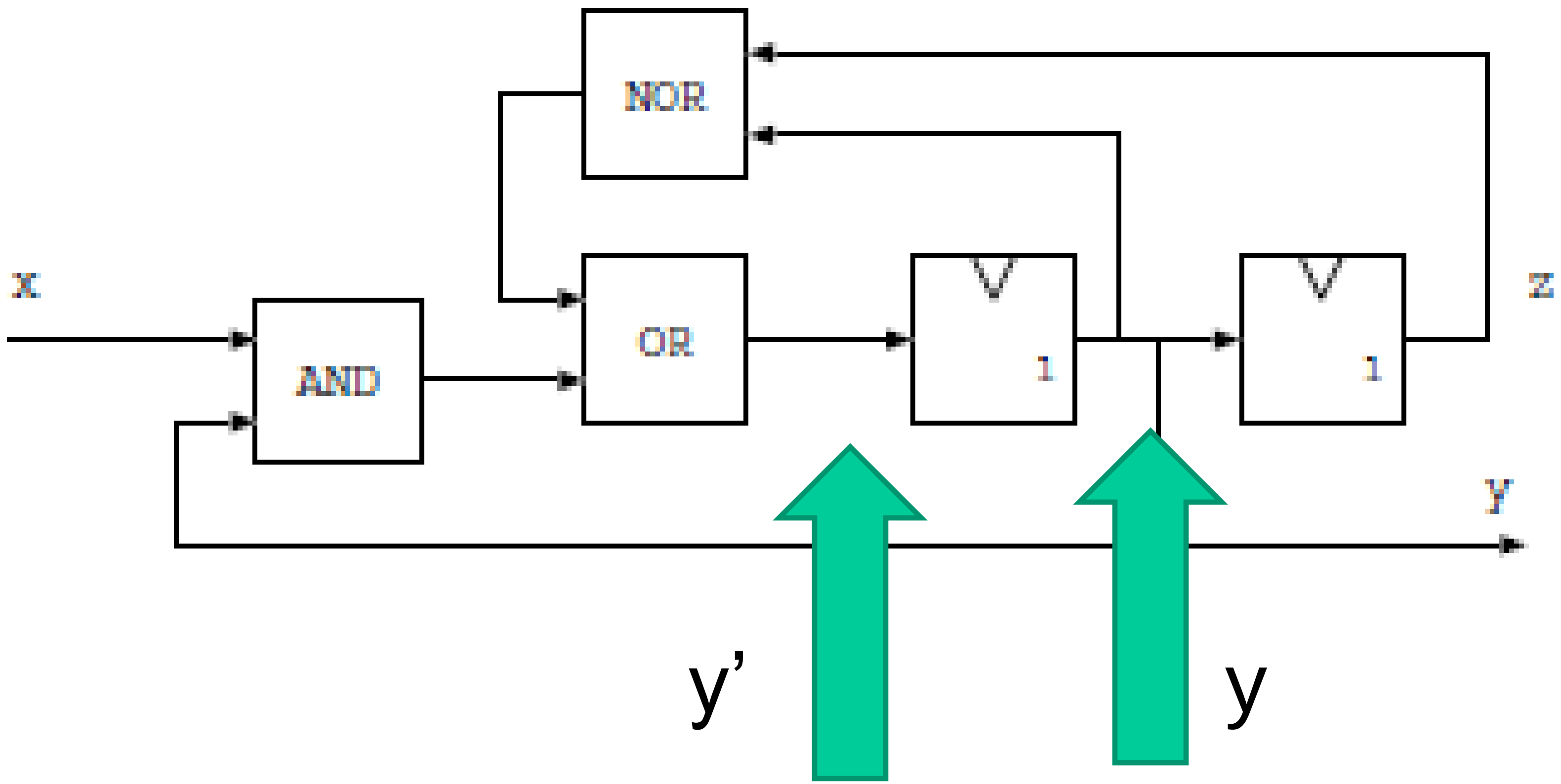
Example of manual calculation (from exam 2009)



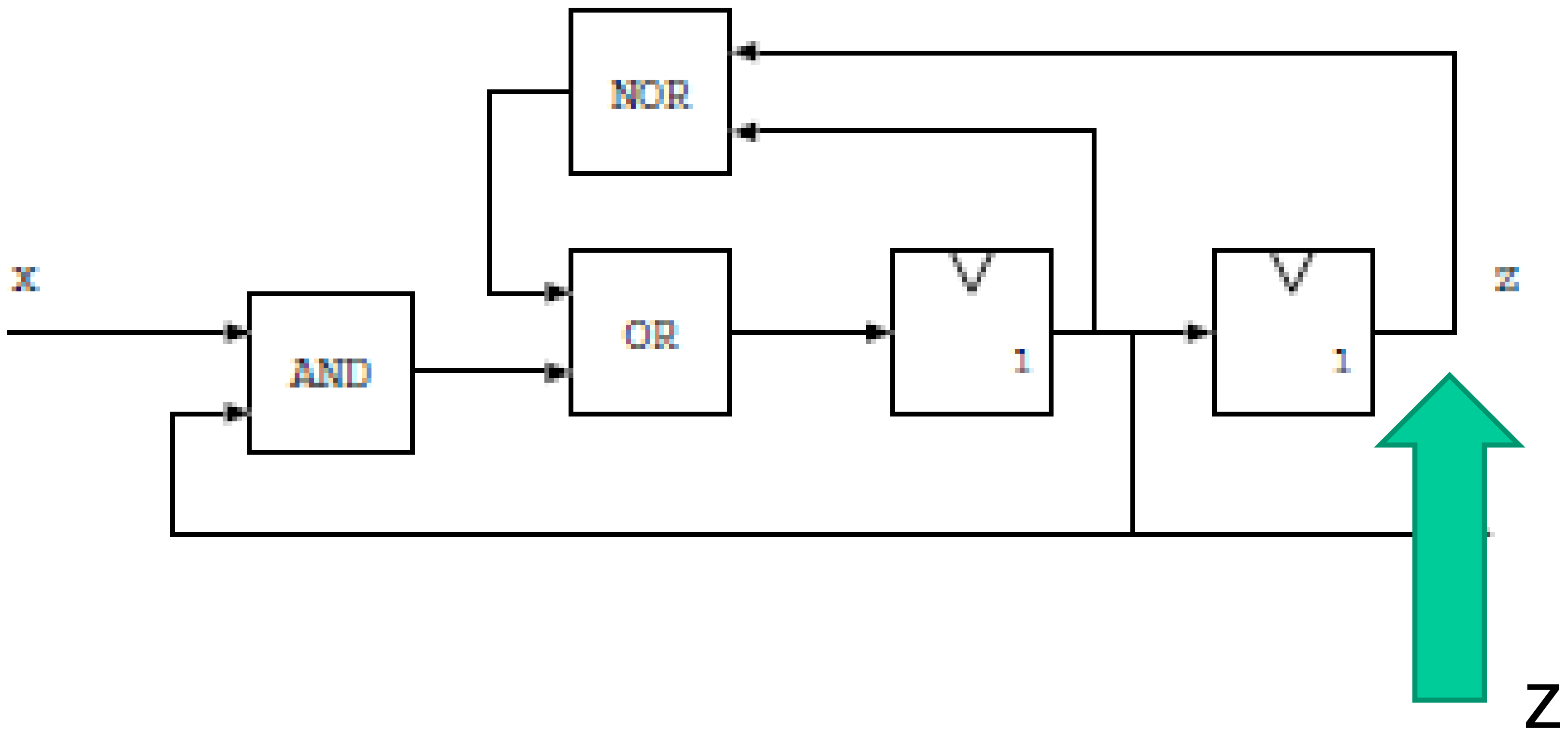
Example of manual calculation (from exam 2009)



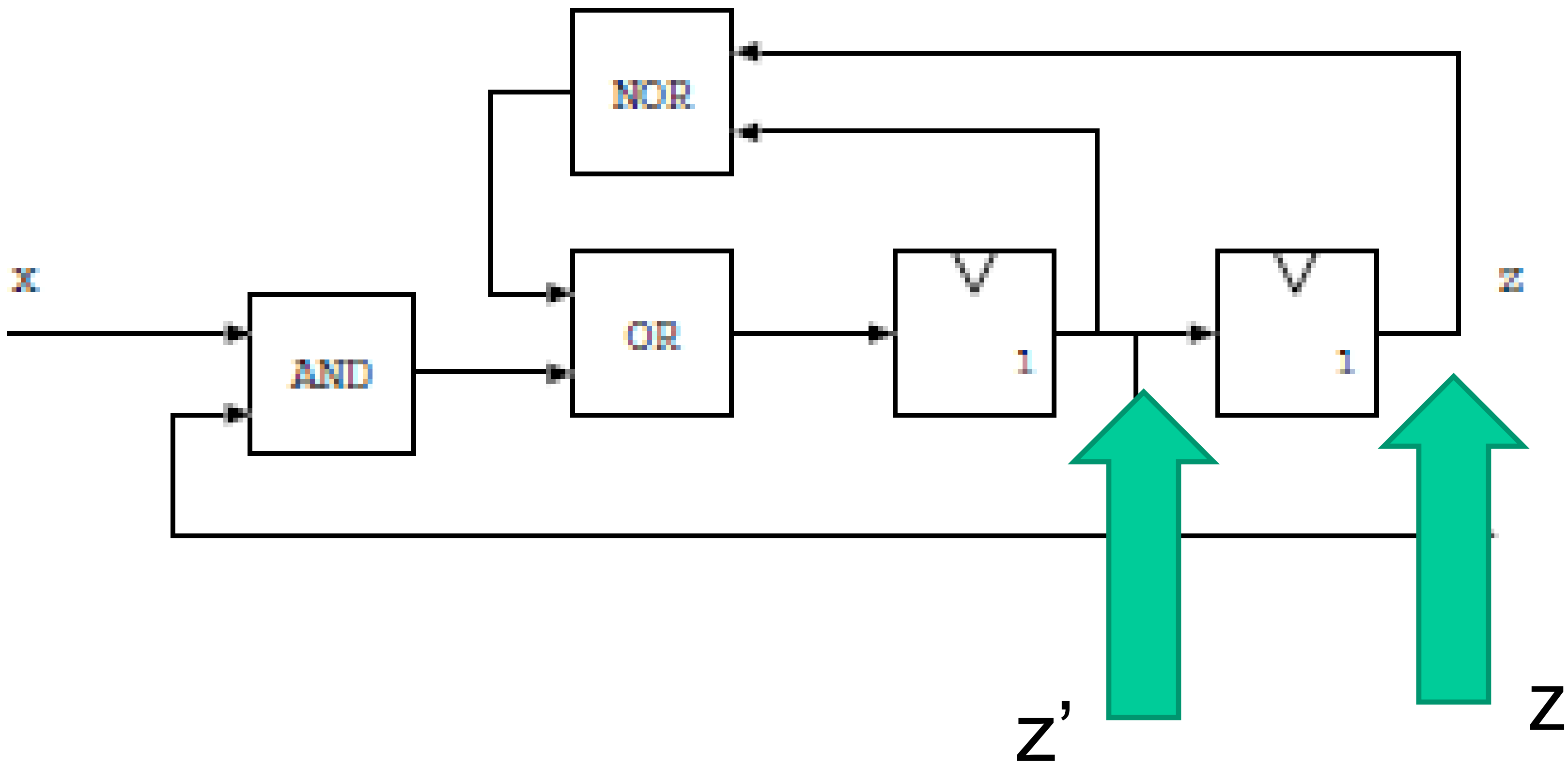
Example of manual calculation (from exam 2009)



Example of manual calculation (from exam 2009)



Example of manual calculation (from exam 2009)



transitions

$(x, y, z) \rightarrow (x', y', z')$

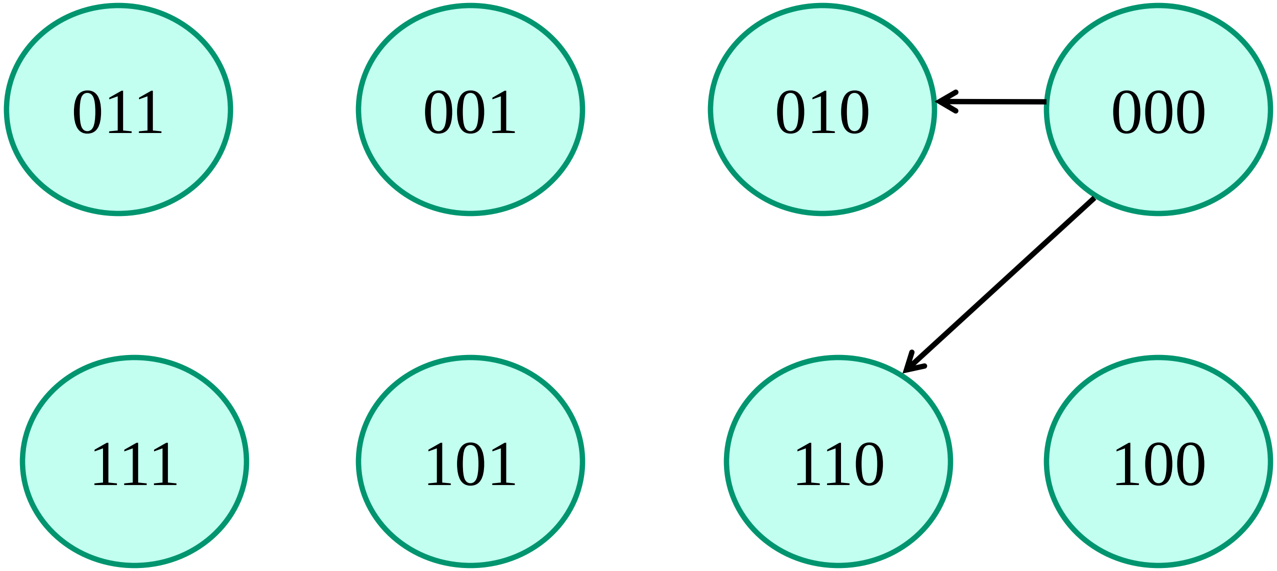
$y' = (x \wedge y) \vee \neg(y \vee z)$

$z' = y$

Show state transition diagram

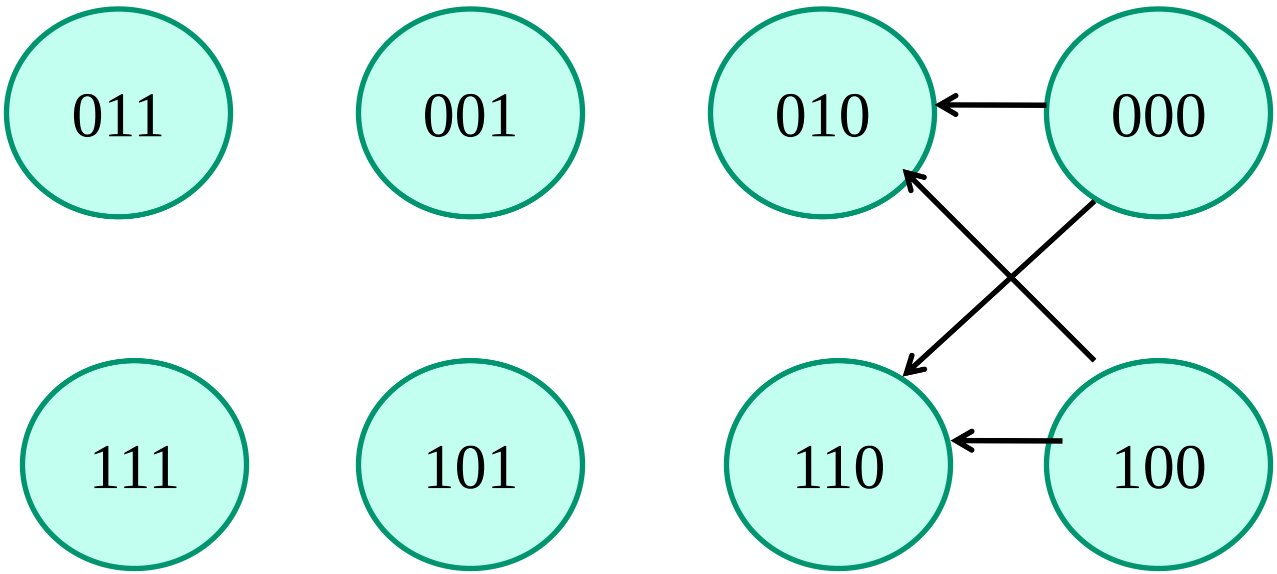
Calculate states in which EG y holds

state transition graph



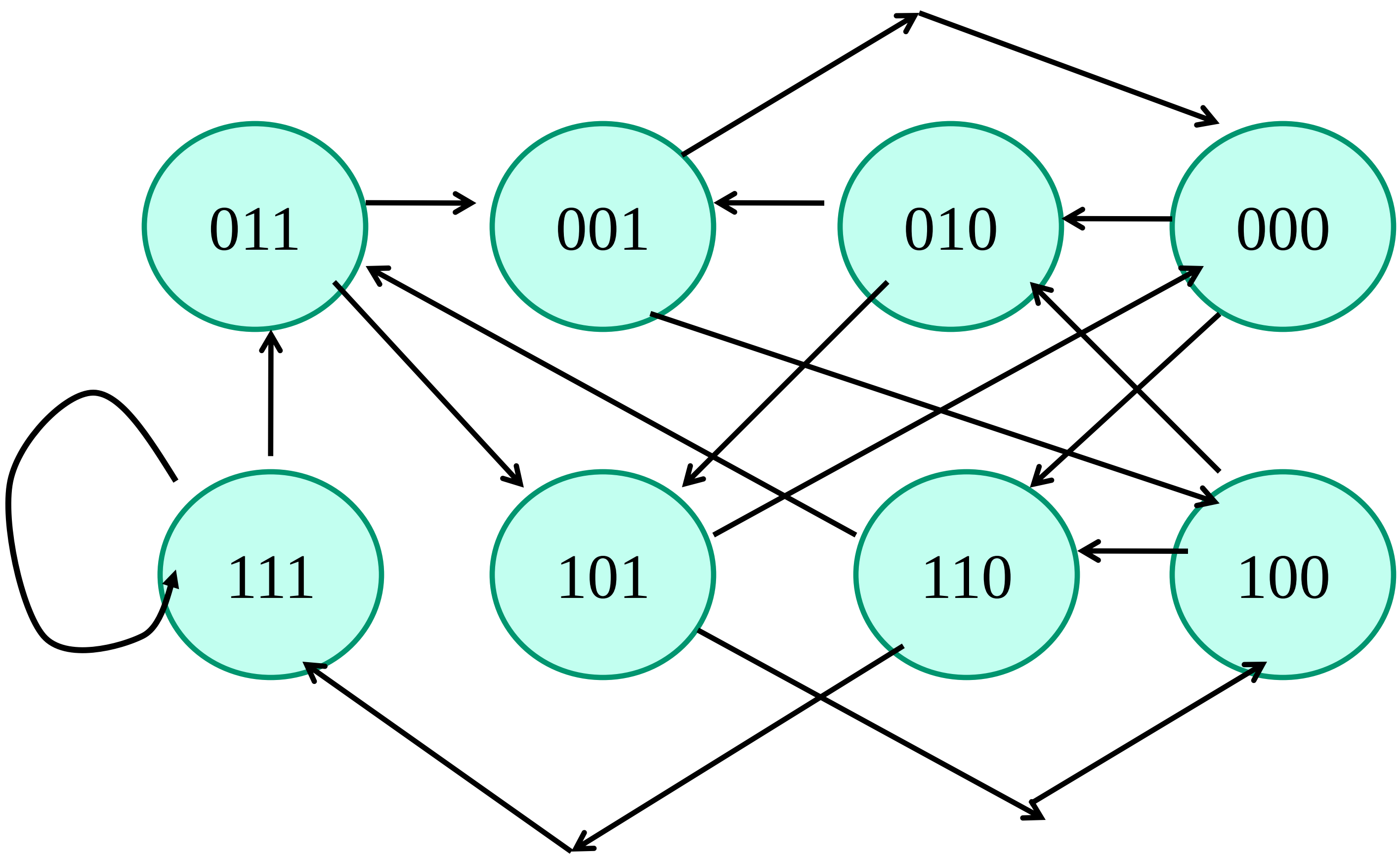
000 -> 010 110

state transition graph



100 -> 010 110

state transition graph



$H(EG\ y) = H(\neg AF \neg y)$

$= S - H(AF \neg y)$

$H(AF \neg y) =$

$\quad Lfp\ U.\ H(\neg y) \cup \{s \mid \text{forall } t\ sRt \Rightarrow t \text{ in } U\}$

$H(\neg y) = \{000, 001, 100, 101\}$

Fixed point iteration

U0 = empty set

U1 = H(¬y) ∪ {s | forall t sRt => t in U0}

= H(¬y) = {000,001,100,101}

U2 = H(¬y) ∪ {s | forall t sRt => t in U1}

= H(¬y) ∪ {011,010}

U3 = H(¬y) ∪ {s | forall t sRt => t in U2}

= H(¬y) ∪ {011,010}

$H(AF\ \neg y) = \{000,001,100,101,011,010\}$

Therefore,

$H(EG\ y) = S - H(AF\ \neg y)$

$= \{110,111\}$

Happy easter!