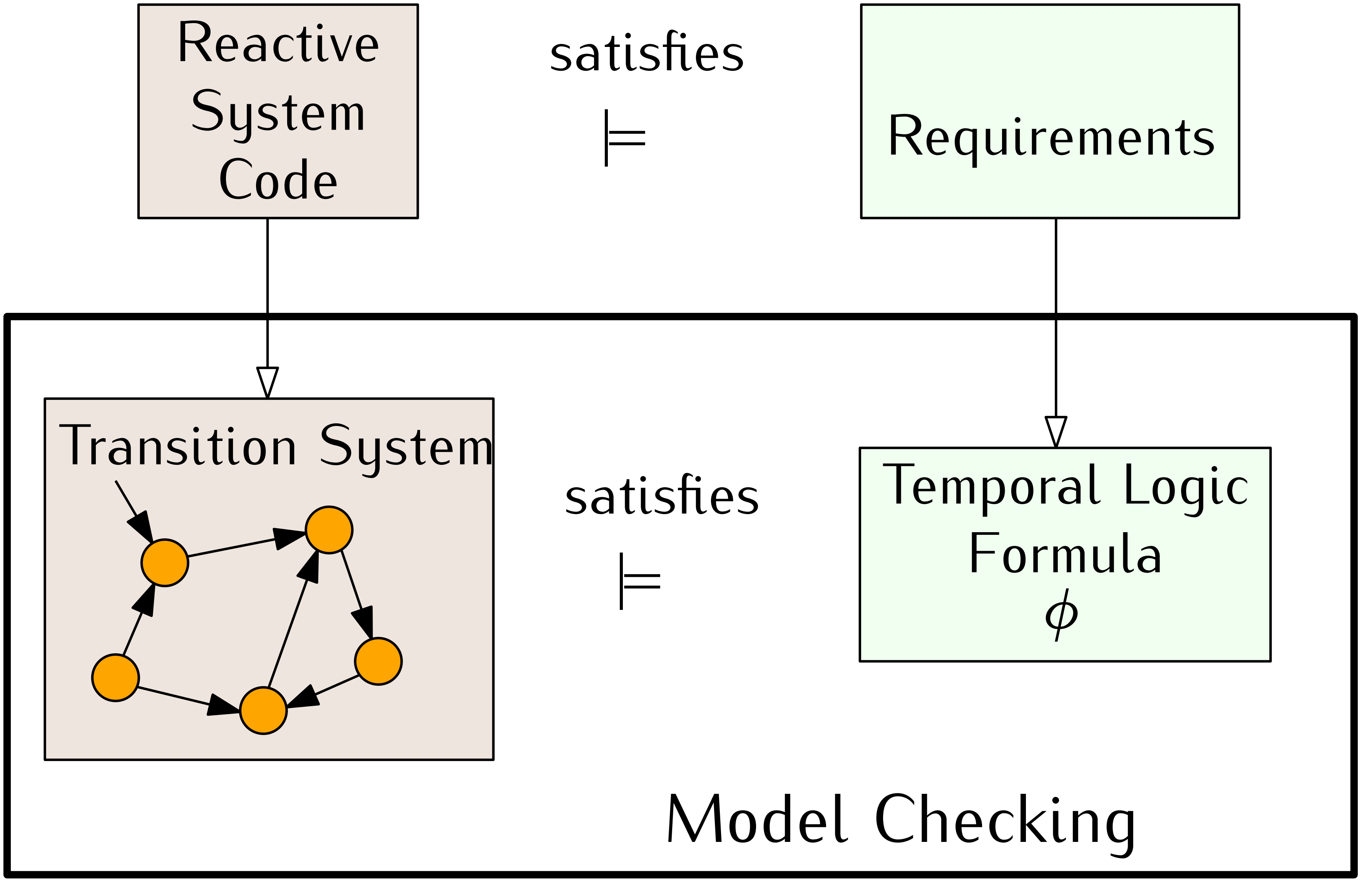
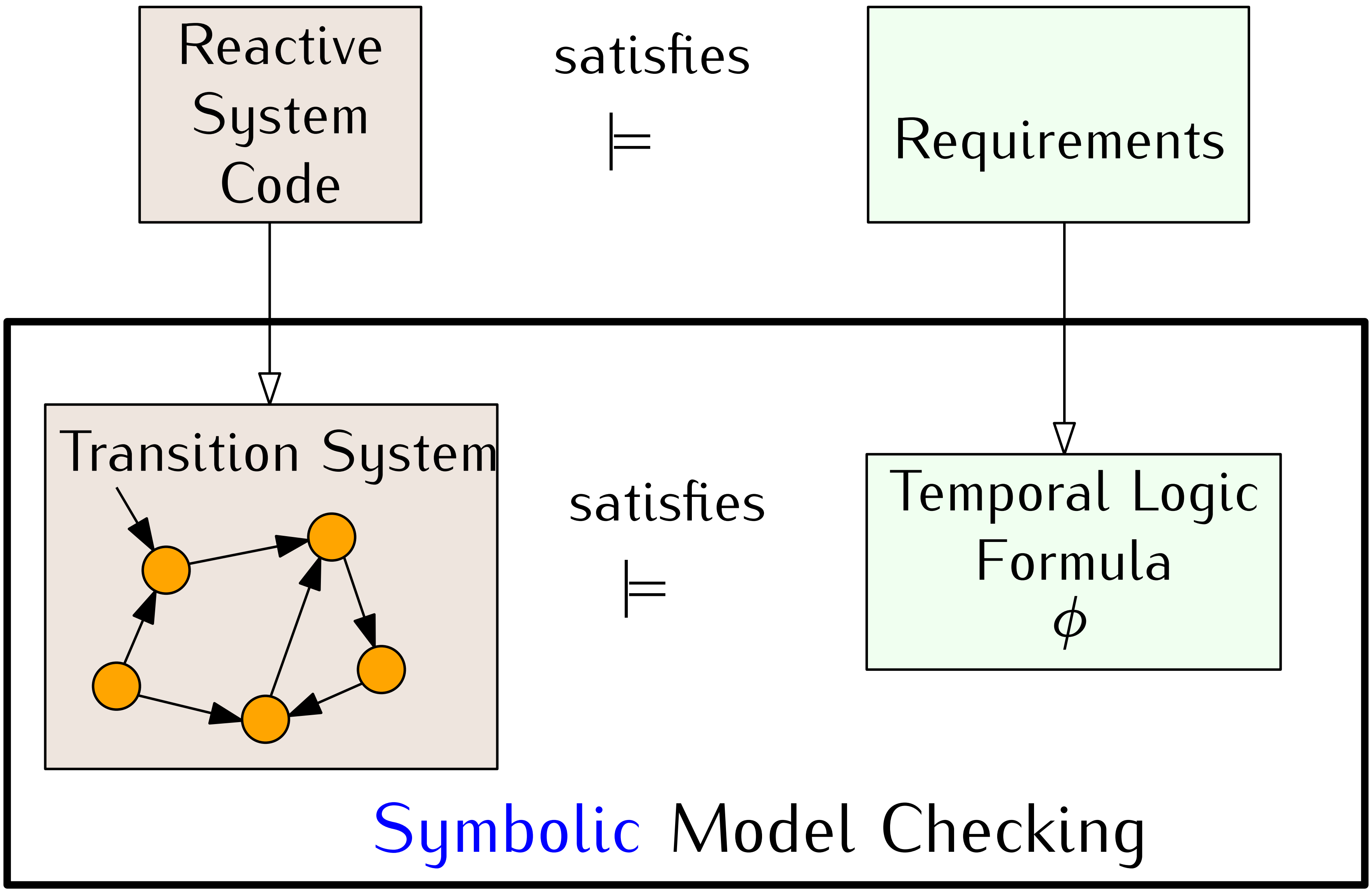
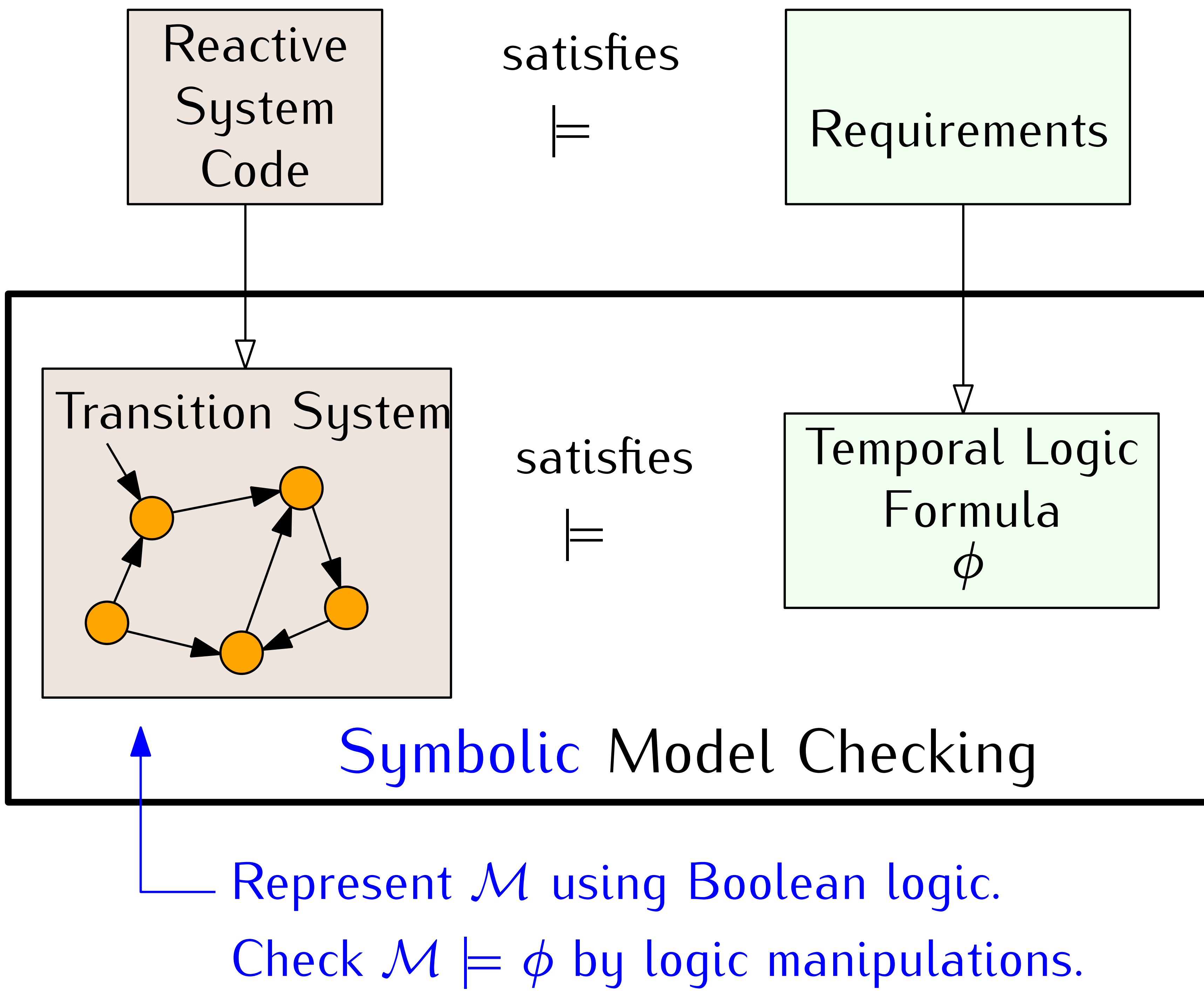
CS 181u Applied Logic

Lecture 15

Symbolic Model Checking Using Py-Z3


Reactive System Code
satisfies
⊨
Requirements
Transition System
satisfies
⊨
Temporal Logic Formula
ϕ
Model Checking


Reactive System Code
satisfies
$\models$
Requirements
Transition System
satisfies
$\models$
Temporal Logic Formula
ϕ
Symbolic Model Checking


Reactive System Code
satisfies
$\models$
Requirements
Transition System
satisfies
$\models$
Temporal Logic Formula ϕ
Symbolic Model Checking
Represent $\mathcal{M}$ using Boolean logic.
Check $\mathcal{M} \models \phi$ by logic manipulations.

Variable Replacement

We often need to replace variables with other expressions. For a formula f, variable v, and expression e, we write $f[e/v]$ to indicate a new formula that is the same as f but with all occurences of v replaced by e.

Example: $f = \neg x \wedge \neg y$

$$f[z/x] = \neg z \wedge \neg y$$

$$f[T/x] = \neg T \wedge \neg y \equiv F \wedge \neg y \equiv F$$

$$f[F/y] = \neg x \wedge \neg F \equiv \neg x \wedge T \equiv \neg x$$

We can do several variables at once:

$$f[(\neg w, F)/(x, y)] = \neg\neg w \wedge \neg F = w$$

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \; : \; f$
2. plugging in all possible values of v into f and taking a disjunction.

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \ : \ f$
2. plugging in all possible values of v into f and taking a disjunction.

For Boolean formulas:

$$\exists v \ : \ f \ \equiv f[T/v] \ \vee \ f[F/v]$$

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \; : \; f$
2. plugging in all possible values of v into f and taking a disjunction.

For Boolean formulas:

$$\exists v \; : \; f \;\equiv f[T/v] \;\vee\; f[F/v]$$

Example: $f = \neg x \wedge \neg y$

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \;:\; f$
2. plugging in all possible values of v into f and taking a disjunction.

For Boolean formulas:

$$\exists v \;:\; f \;\equiv f[T/v] \;\vee\; f[F/v]$$

Example: $\quad f = \neg x \wedge \neg y$

$\exists y \;:\; f \;\equiv:\; f[T/y] \;\vee\; f[F/y]$

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \;:\; f$
2. plugging in all possible values of v into f and taking a disjunction.

For Boolean formulas:

$$\exists v \;:\; f \;\equiv f[T/v] \;\vee\; f[F/v]$$

Example: $\quad f = \neg x \wedge \neg y$

$\exists y \;:\; f \;\equiv :\; f[T/y] \;\vee\; f[F/y]$

$\equiv \; (\neg x \wedge \neg T) \;\vee\; (\neg x \wedge \neg F)$

Existential Quantifier Elimination

For a formula f, we can "get rid" of a variable v by

1. writing $\exists v \ : \ f$
2. plugging in all possible values of v into f and taking a disjunction.

For Boolean formulas:

$$\exists v \ : \ f \ \equiv f[T/v] \ \vee \ f[F/v]$$

Example: $\quad f = \neg x \wedge \neg y$

$\exists y \ : \ f \ \equiv : \ f[T/y] \ \vee \ f[F/y]$

$\equiv \ (\neg x \wedge \neg T) \ \vee \ (\neg x \wedge \neg F)$

$\equiv \ F \ \vee \ \neg x \equiv \neg x$

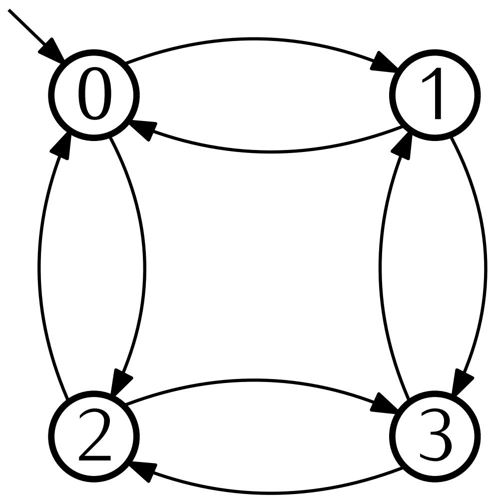
No more y

Explicit Model Representation

The transition system $\mathcal{M}$ is specified
by literally listing out all of the pieces.

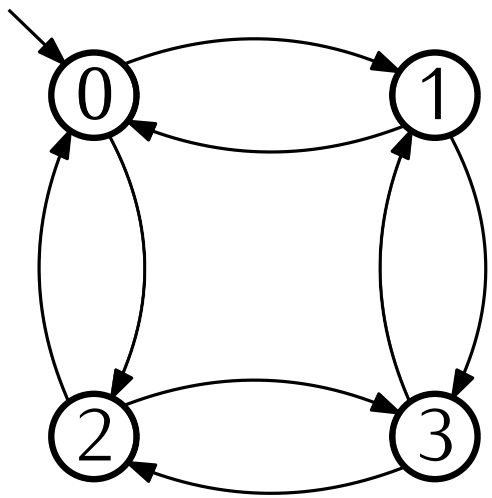
Explicit Model Representation

The transition system $\mathcal{M}$ is specified
by literally listing out all of the pieces.



Explicit Model Representation

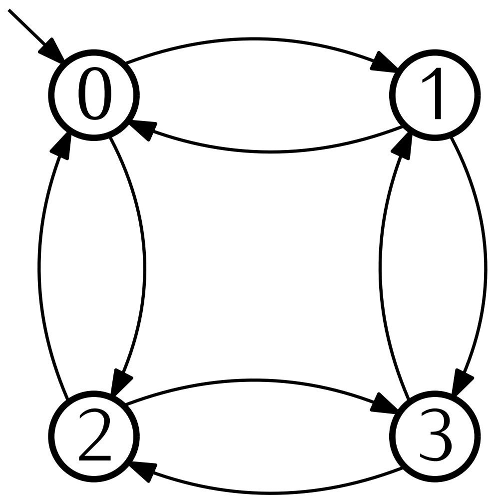
The transition system $\mathcal{M}$ is specified by literally listing out all of the pieces.



States: $S = \{0, 1, 2, 3\}$

Explicit Model Representation

The transition system $\mathcal{M}$ is specified
by literally listing out all of the pieces.

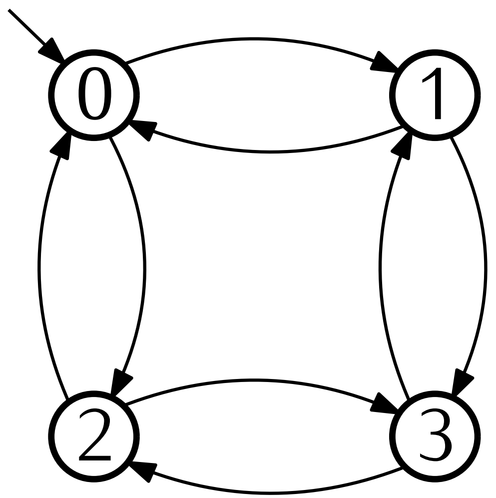


States: $S = \{0, 1, 2, 3\}$

Initial States: $I = \{0\}$

Explicit Model Representation

The transition system $\mathcal{M}$ is specified
by literally listing out all of the pieces.



States: $S = \{0, 1, 2, 3\}$

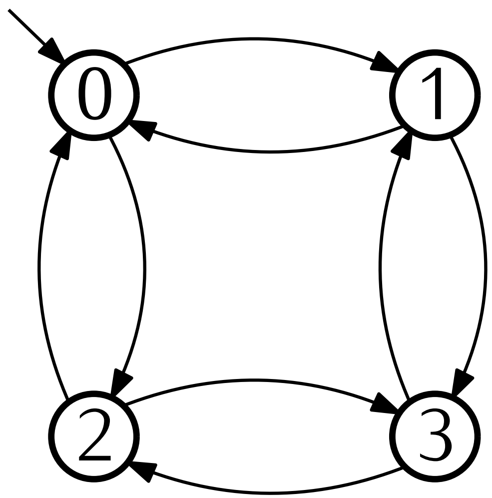
Initial States: $I = \{0\}$

Transitions:

$$R = \left\{ \begin{matrix} (0,1) & (0,2) & (1,3) & (2,3) \\ (1,0) & (2,0) & (3,1) & (3,2) \end{matrix} \right\}$$

Explicit Model Representation

The transition system $\mathcal{M}$ is specified
by literally listing out all of the pieces.



States: $S = \{0, 1, 2, 3\}$

Initial States: $I = \{0\}$

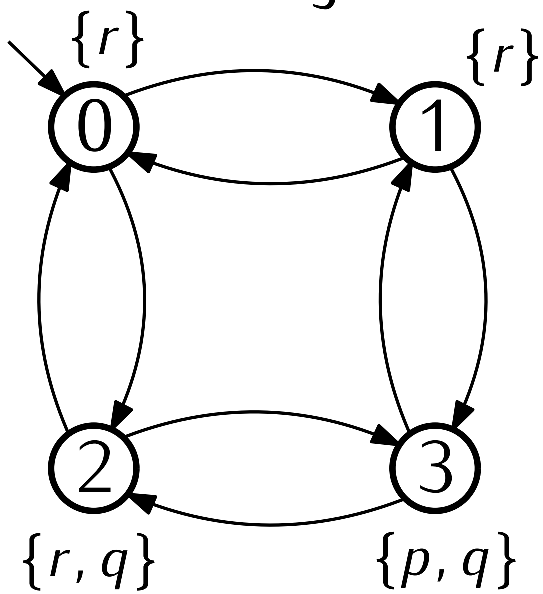
Transitions:

$$R = \left\{ \begin{array}{cccc} (0,1) & (0,2) & (1,3) & (2,3) \\ (1,0) & (2,0) & (3,1) & (3,2) \end{array} \right\}$$

Atomic Propositions: $AP = \{p, q, r\}$

Explicit Model Representation

The transition system $\mathcal{M}$ is specified by literally listing out all of the pieces.



States: $S = \{0, 1, 2, 3\}$

Initial States: $I = \{0\}$

Transitions:

$$R = \left\{ \begin{array}{cccc} (0,1) & (0,2) & (1,3) & (2,3) \\ (1,0) & (2,0) & (3,1) & (3,2) \end{array} \right\}$$

Atomic Propositions: $AP = \{p, q, r\}$

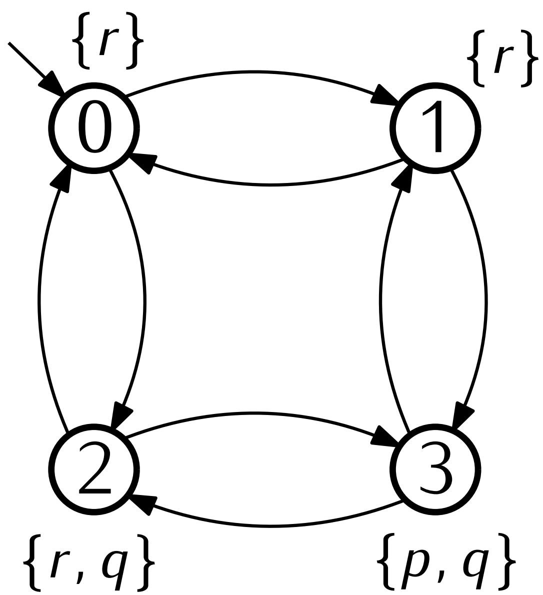
Labelling Function $\mathcal{L} : S \to \mathcal{P}(AP)$

$$\mathcal{L}(0) = \{r\} \qquad \mathcal{L}(2) = \{r, q\}$$

$$\mathcal{L}(1) = \{r\} \qquad \mathcal{L}(1) = \{p, q\}$$

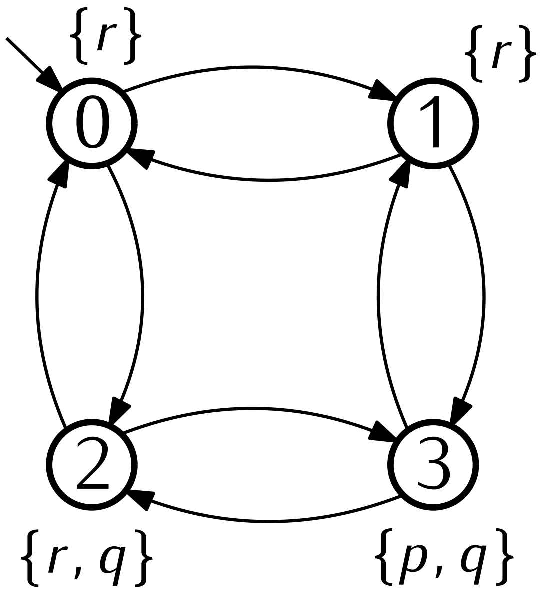
Symbolic Model Representation

Represent $\mathcal{M}$ using Booelan logic.



Symbolic Model Representation

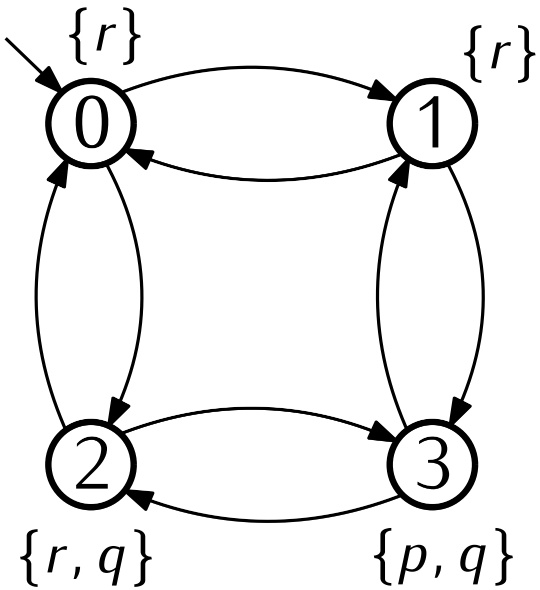
Represent $\mathcal{M}$ using Booelan logic.



States		
0		
1		
2		
3		

Symbolic Model Representation

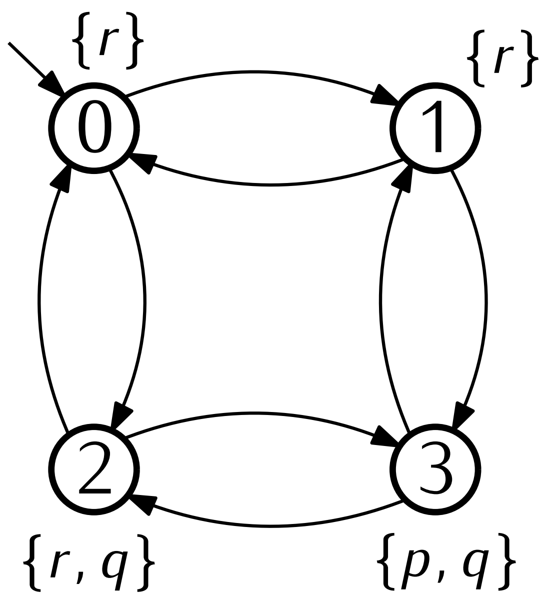
Represent $\mathcal{M}$ using Booelan logic.



States	binary	
	x	y
0	0	0
1	0	1
2	1	0
3	1	1

Symbolic Model Representation

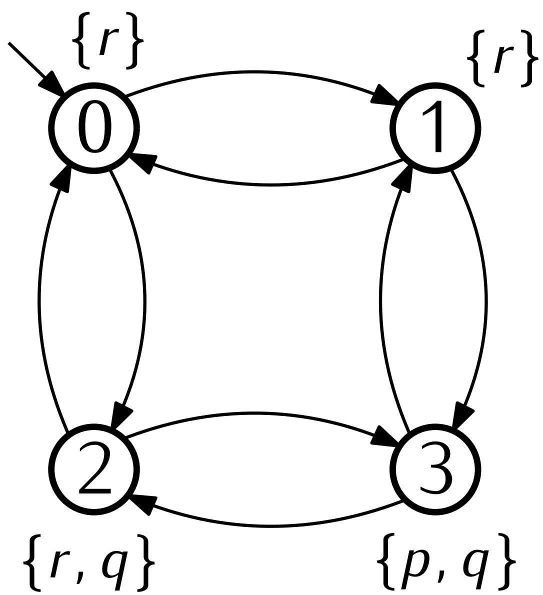
Represent $\mathcal{M}$ using Booelan logic.



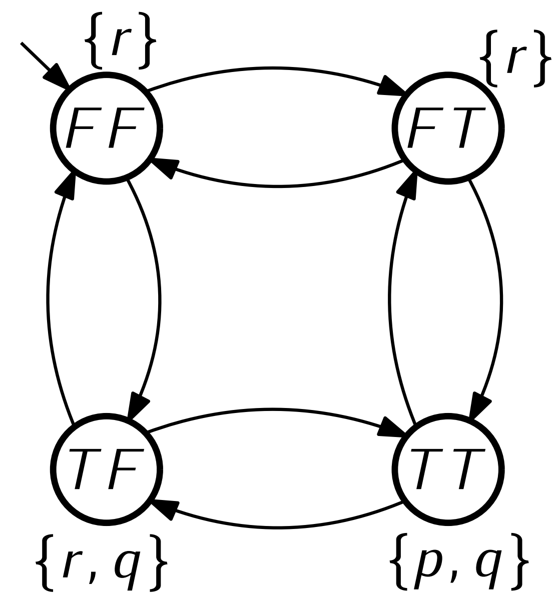
States	binary		truth values	
	x	y	x	y
0	0	0	F	F
1	0	1	F	T
2	1	0	T	F
3	1	1	T	T

Symbolic Model Representation

Represent $\mathcal{M}$ using Booelan logic.



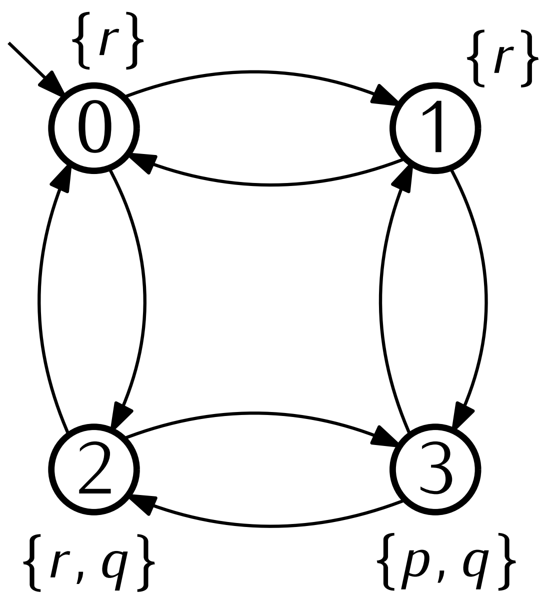
Boolean state variables
$V = \{x, y\}$



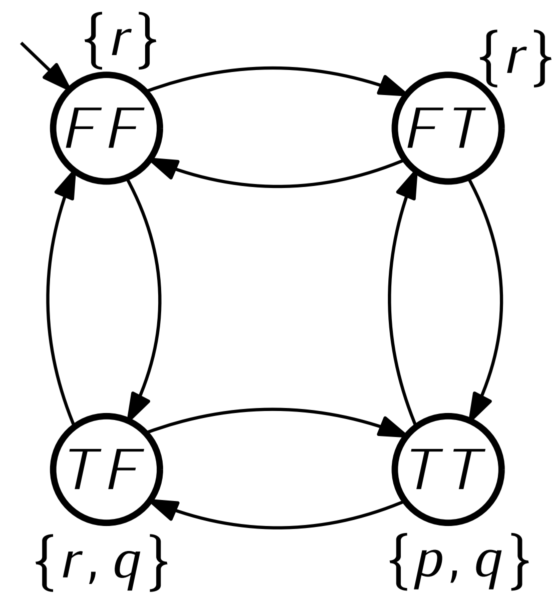
States	binary		truth values	
	x	y	x	y
0	0	0	F	F
1	0	1	F	T
2	1	0	T	F
3	1	1	T	T

Symbolic Model Representation

Represent $\mathcal{M}$ using Booelan logic.



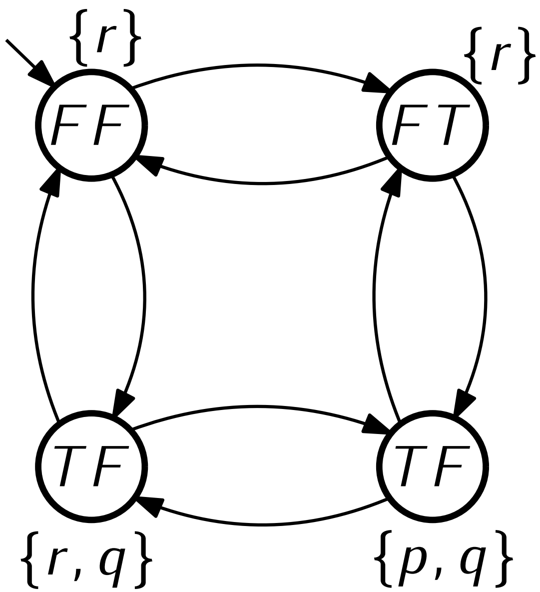
Boolean state variables
$V = \{x, y\}$



States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

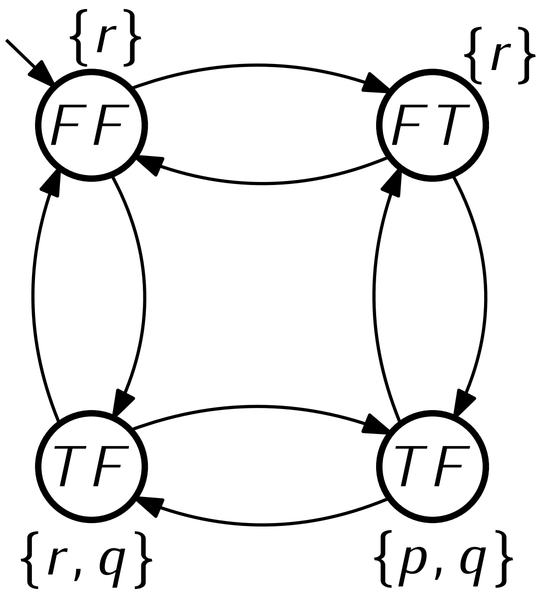
Represent $\mathcal{M}$ using Boolean logic.



States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.

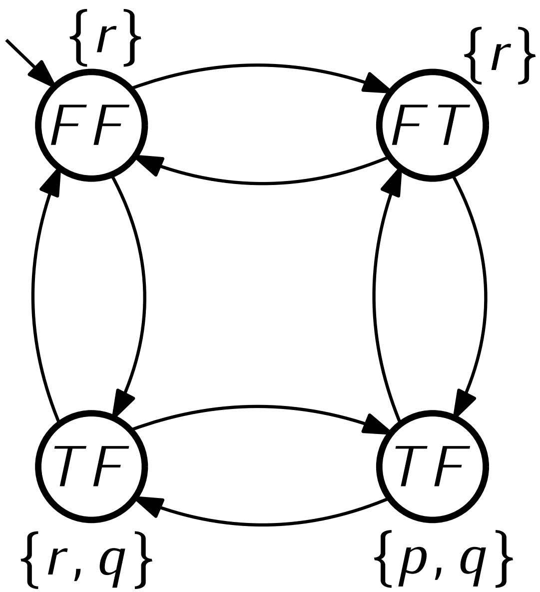


Initial State: $\neg x \wedge \neg y$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



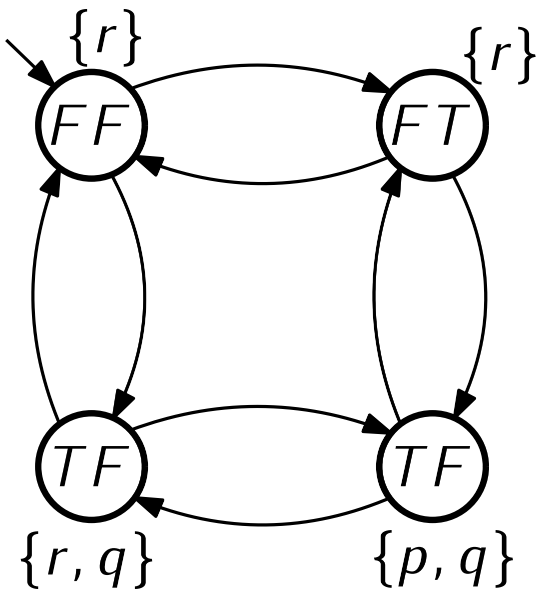
Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Initial State: $\neg x \wedge \neg y$

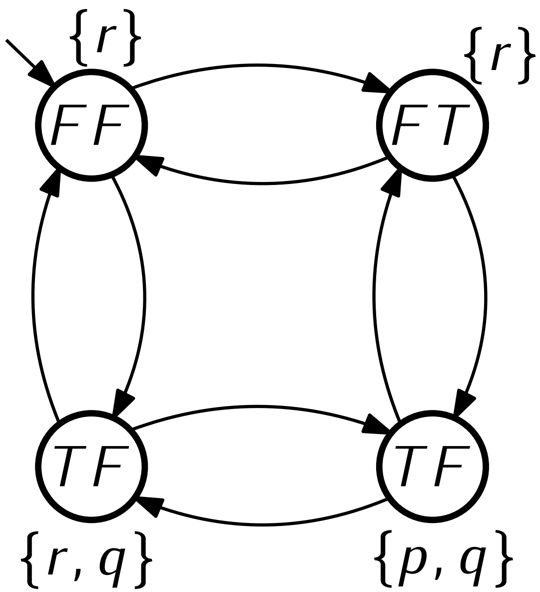
Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : S \to \mathcal{P}(AP)$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Initial State: $\neg x \wedge \neg y$

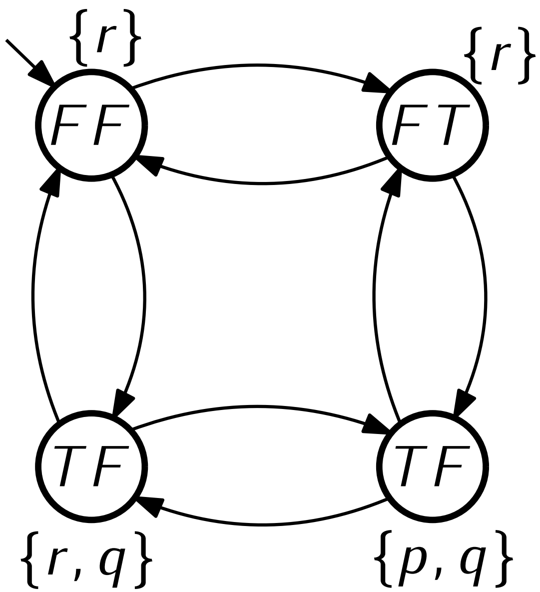
Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\cancel{\mathcal{L} : S \to \mathcal{P}(AP)}$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

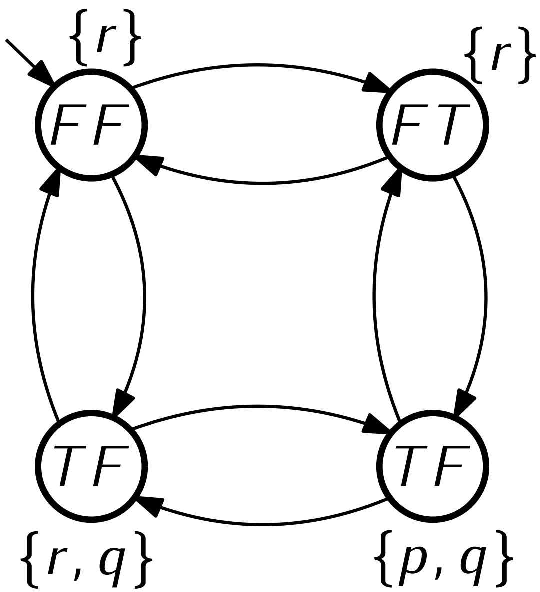
Labelling Function $\cancel{\mathcal{L} : S \to \mathcal{P}(AP)}$

$\mathcal{L} : AP \to \mathcal{F}(x, y)$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

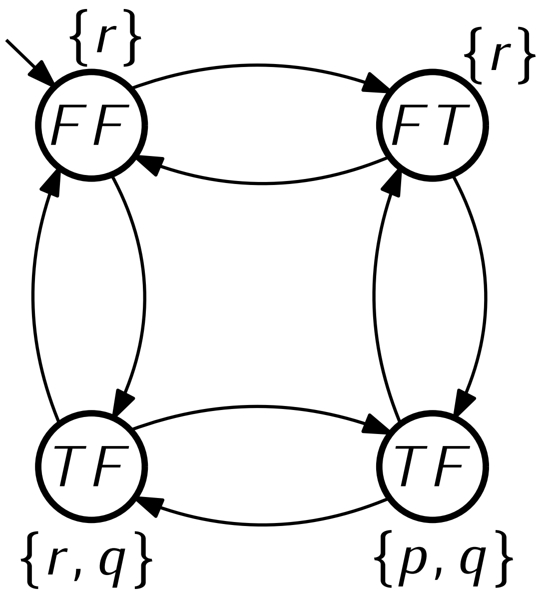
Labelling Function $\cancel{\mathcal{L} : S \to \mathcal{P}(AP)}$

$$\mathcal{L} : AP \to \mathcal{F}(x, y) \qquad \begin{aligned} p &\equiv x \wedge y \\ q &\equiv x \\ r &\equiv \neg(x \wedge y) \equiv \neg p \end{aligned}$$

States	binary		truth values		Boolean formula
	x	y	x	y	
0	0	0	F	F	$\neg x \wedge \neg y$
1	0	1	F	T	$\neg x \wedge y$
2	1	0	T	F	$x \wedge \neg y$
3	1	1	T	T	$x \wedge y$

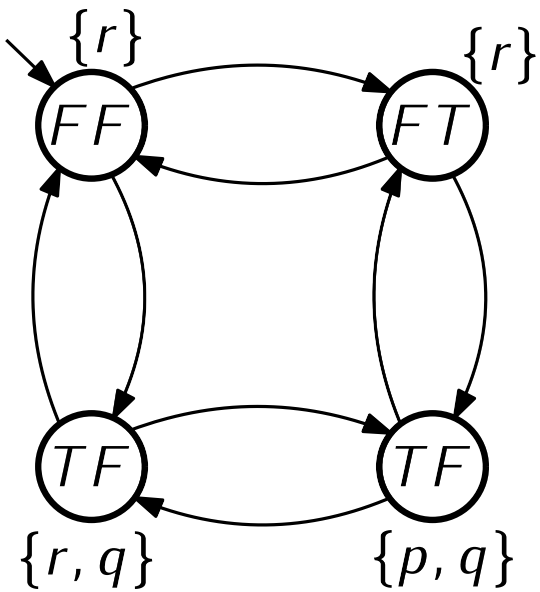
Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



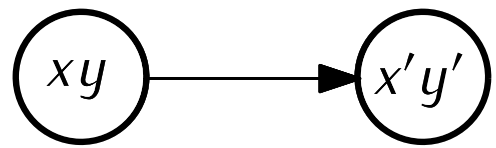
Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



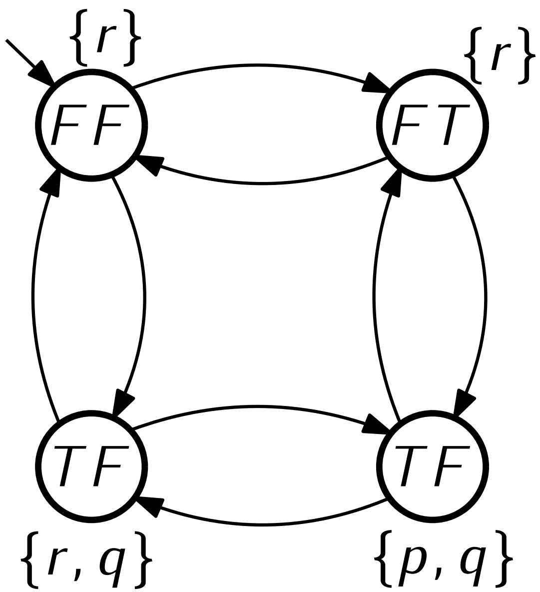
Transitions:

Let the "next" state variables be $V' = \{x', y'\}$



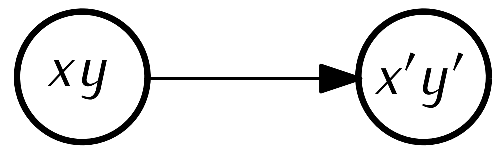
Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Transitions:

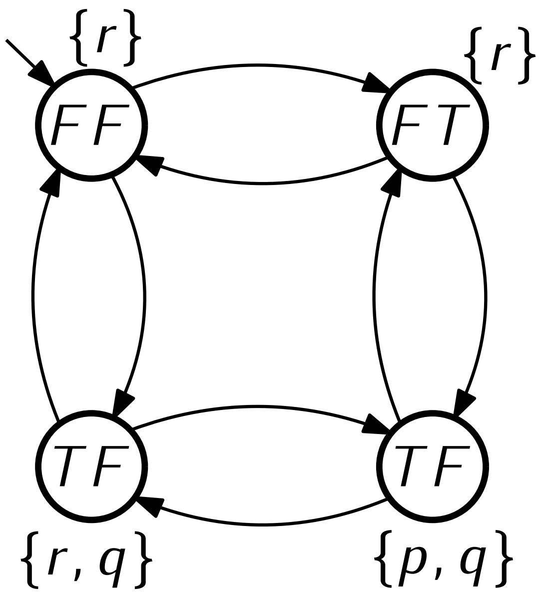
Let the "next" state variables be $V' = \{x', y'\}$



$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

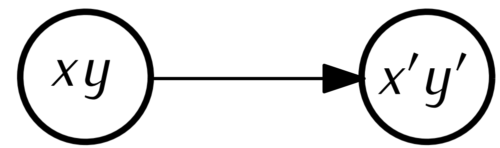
Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Transitions:

Let the “next” state variables be $V' = \{x', y'\}$

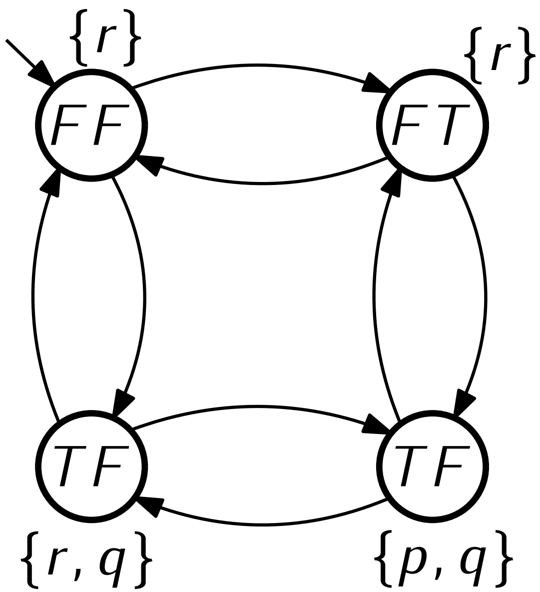


$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

“we can get from one state to the next by keeping one variable the same and negating the other”

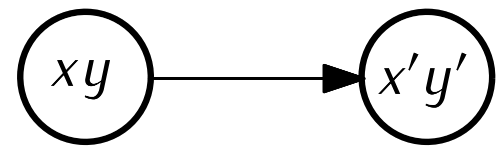
Symbolic Model Representation

Represent $\mathcal{M}$ using Boolean logic.



Transitions:

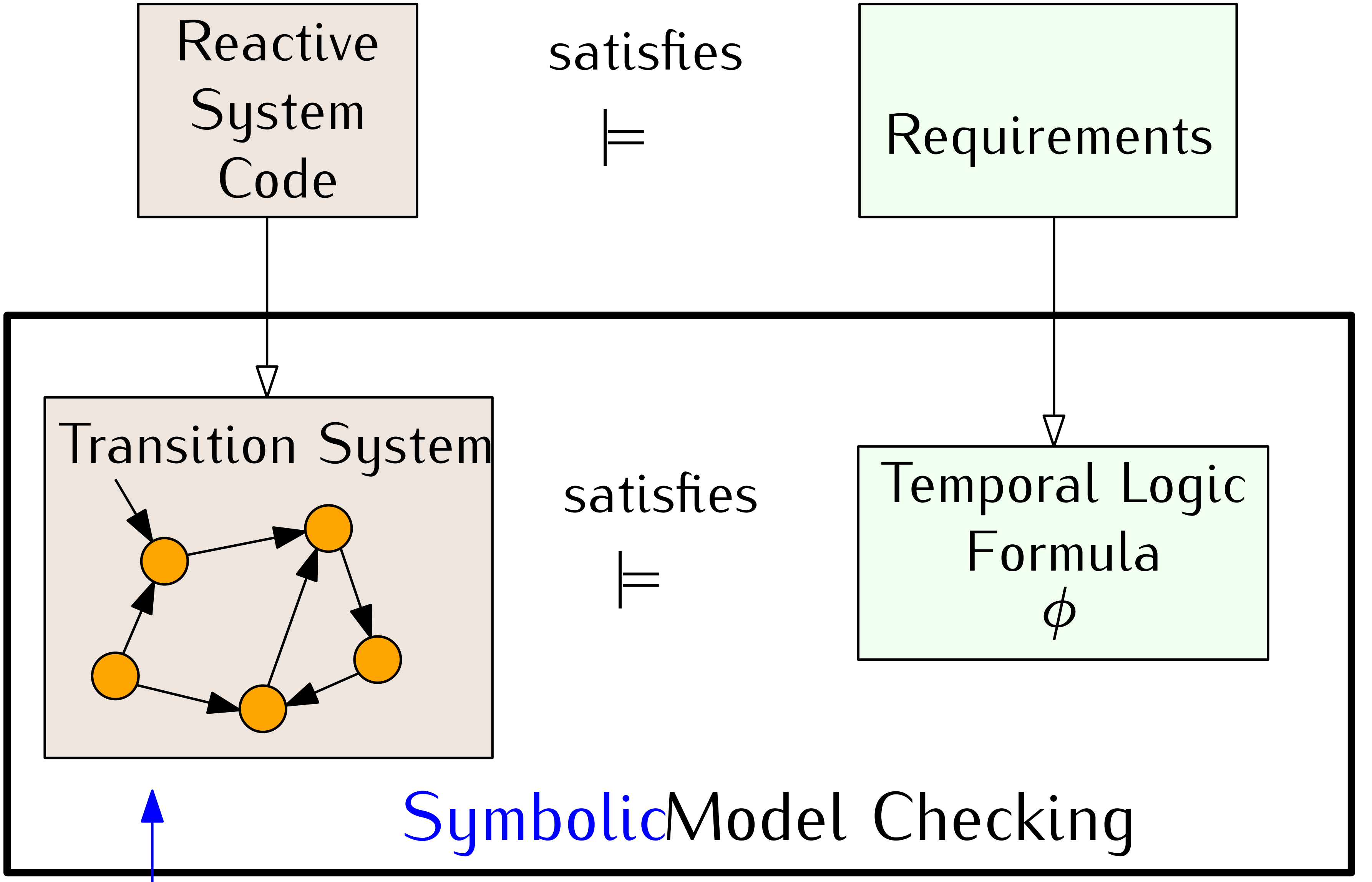
Let the "next" state variables be $V' = \{x', y'\}$



$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Explicit transitions	$(0,1)$	$(2,3)$	$(1,3)$	$(0,2)$
	$(1,0)$	$(3,2)$	$(3,1)$	$(2,0)$

"we can get from one state to the next by keeping one variable the same and negating the other"

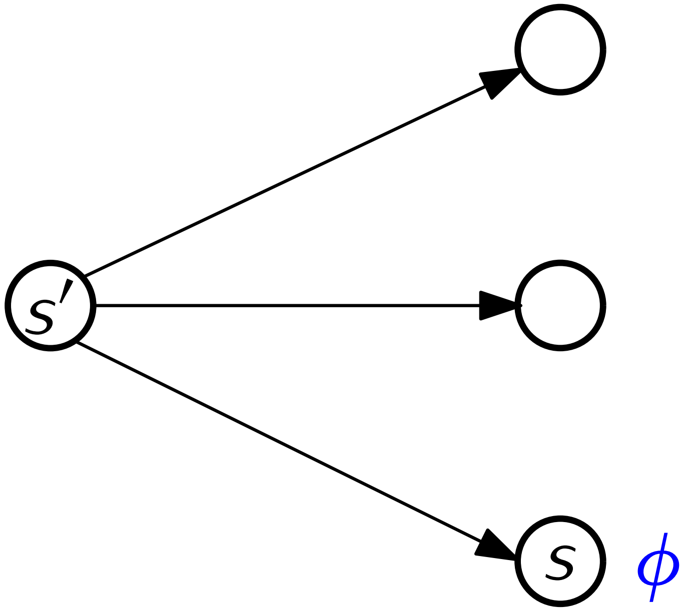


Represent $\mathcal{M}$ using Boolean logic.

Check $\mathcal{M} \models \phi$ by logic manipulations.

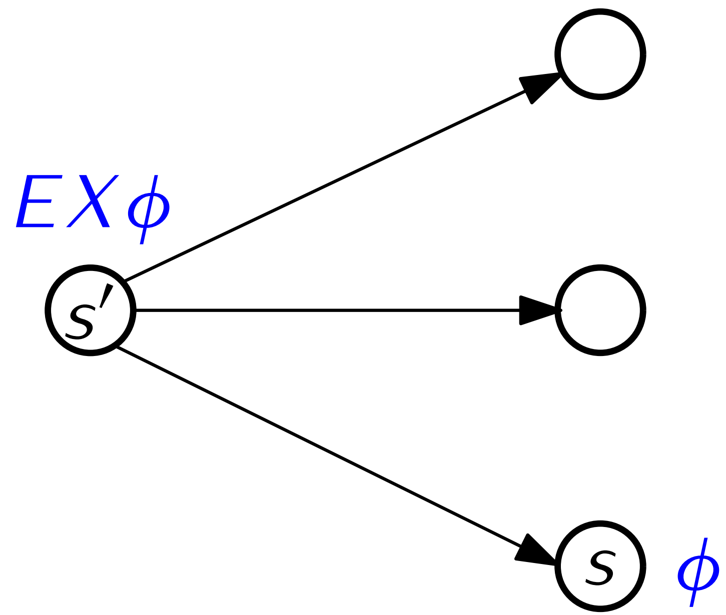
The Algorithm for $EX\ \phi$

After labelling all states s that satisfy ϕ, label and state s' with $EX\phi$ if there is a transition from s' to s.



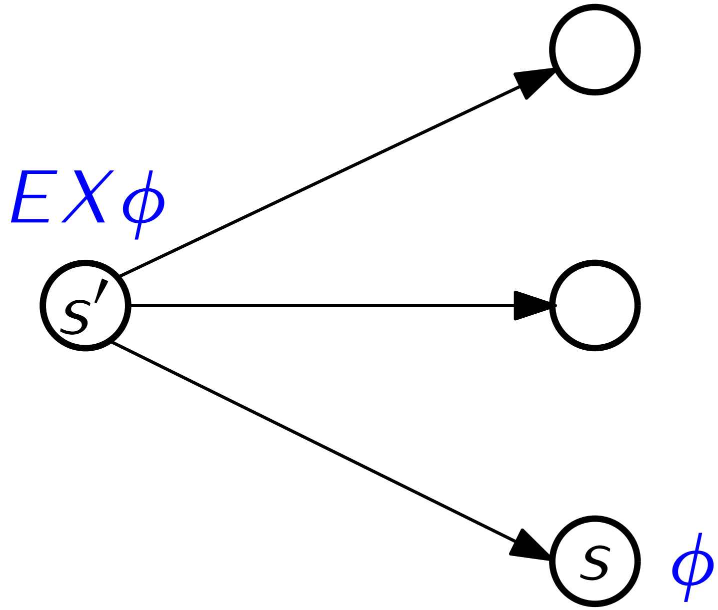
The Algorithm for $EX\ \phi$

After labelling all states s that satisfy ϕ, label and state s' with $EX\phi$ if there is a transition from s' to s.



The Algorithm for $EX\ \phi$

After labelling all states s that satisfy ϕ, label and state s' with $EX\phi$ if there is a transition from s' to s.



Call this process $\mathrm{SAT}_{EX}(\phi)$

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

$$EX\ \phi \equiv \exists V' \;\; R \;\wedge\; \phi[\,V' \;/\; V]$$

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

$$\underline{EX\ \phi} \equiv \exists V' \ \ R\ \wedge\ \phi[\ V'\ /\ V]$$

exists a path where ϕ holds in the next state

$$\equiv$$

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

$$\underbrace{EX\ \phi}_{} \equiv \underbrace{\exists V'}_{}\ R \land \phi[\,V'\,/\,V\,]$$

exists a path where ϕ holds in the next state $\equiv$ there is some assignment for the next state variables

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

$$\underbrace{EX\ \phi}_{\text{exists a path where } \phi \text{ holds in the next state}} \equiv \underbrace{\exists V'}_{\text{there is some assignment for the next state variables}}\ \underbrace{R}_{\text{obeys the transition relation}} \wedge\ \phi[\ V'\ /\ V]$$

exists a path where ϕ holds in the next state $\equiv$

there is some assignment for the next state variables

obeys the transition relation

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

$$\underline{EX\ \phi} \equiv \underline{\exists V'}\ \ \underline{R}\ \wedge\ \underline{\phi[\ V'\ /\ V]}$$

exists a path where ϕ holds in the next state

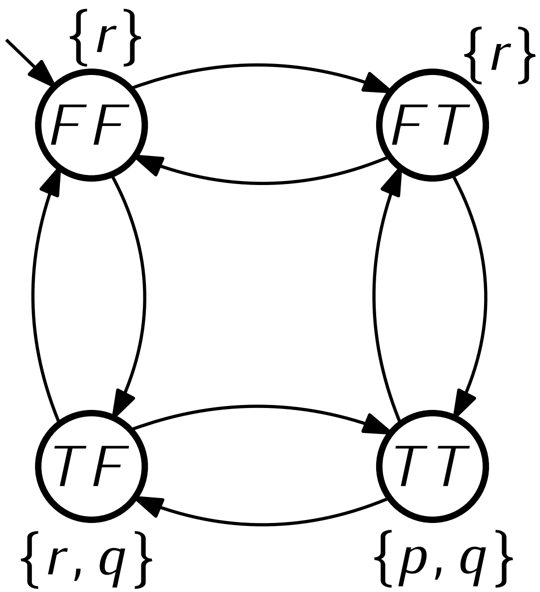
$\equiv$

there is some assignment for the next state variables

obeys the transition relation

ϕ holds when variables are updated with the new state variables

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

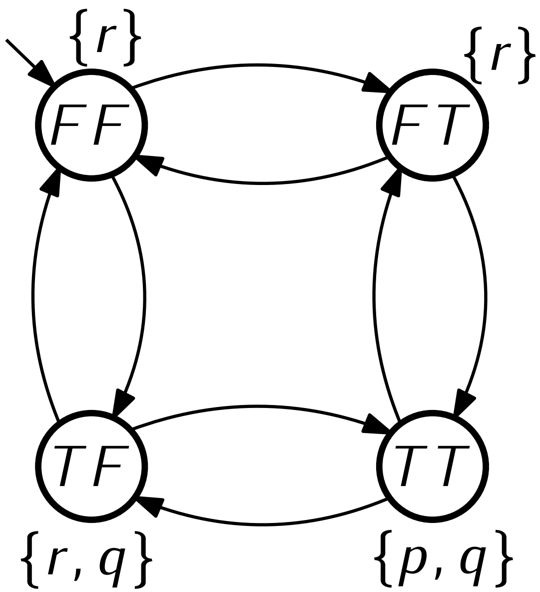
Labelling Function $\quad \mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\quad \mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

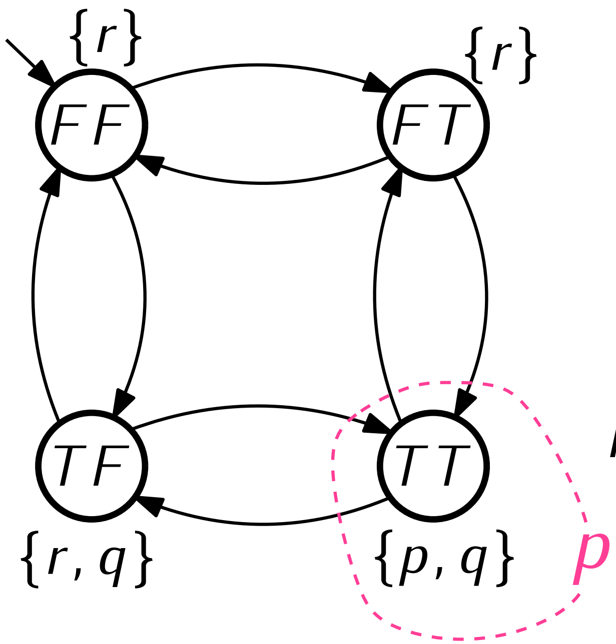
$$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$$

Transition Relation:

$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

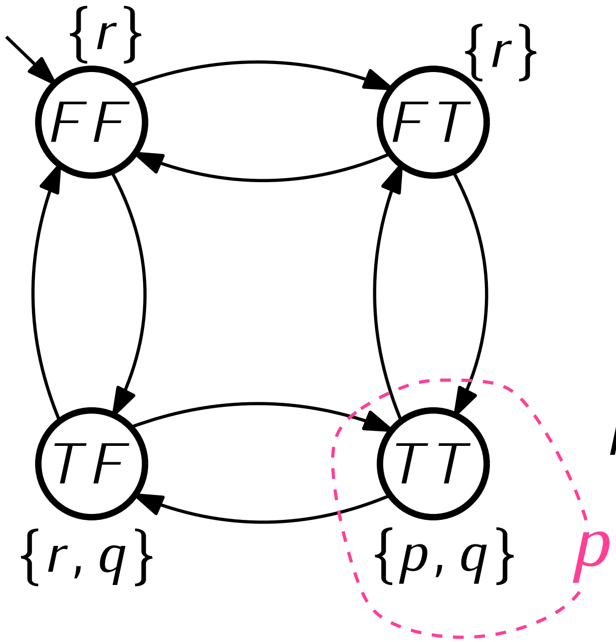
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

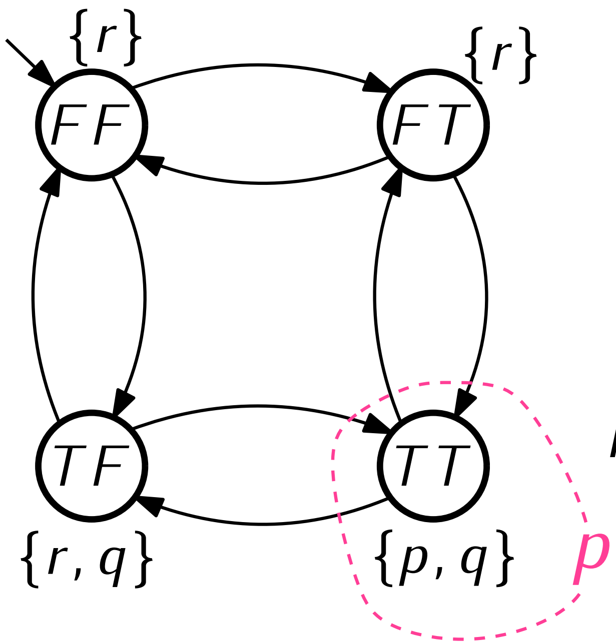
Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V' \ / V \ ]$$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

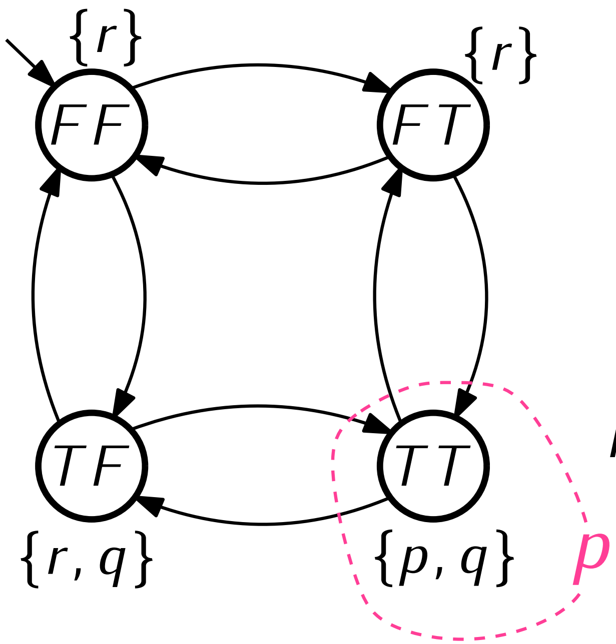
$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V' \ /V\ ]$$

$$EX\ p \equiv \exists x', y' \ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

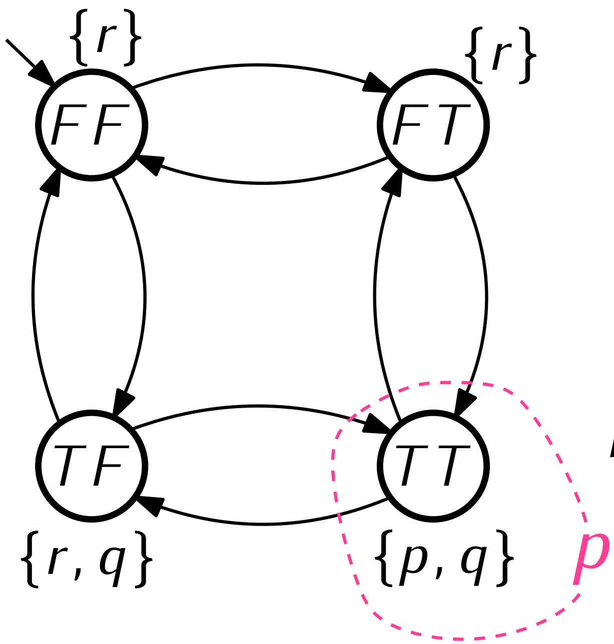
Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V' \ / V\ ]$$

$$EX\ p \equiv \exists x', y' \ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$$

...some Boolean simplifications ...

Symbolic Model Checking



Initial State: $\neg x \land \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \land y \qquad q \equiv x \qquad r \equiv \neg(x \land y)$

Transition Relation:

$$R \equiv (x' = x \land y' = \neg y) \lor (x' = \neg x \land y' = y)$$

Let's compute $EX\ p$

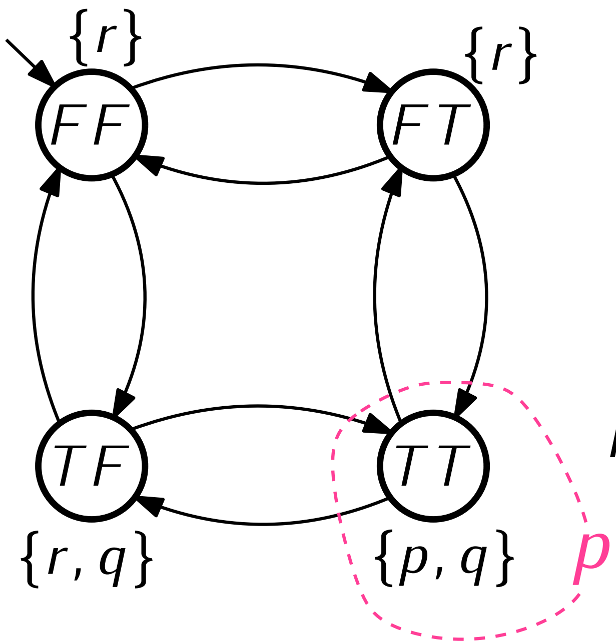
$$EX\ p \equiv \exists V' \ \ R \ \land \ p[\ V' \ /V\ ]$$

$$EX\ p \equiv \exists x', y' \ \ (x' = x \land y' = \neg y) \lor (x' = \neg x \land y' = y) \ \land \ (x' \land y')$$

...some Boolean simplifications ...

$$EX\ p \equiv \exists x', y' \ \ (x' \land x \land y' \land \neg y) \lor (x' \land \neg x \land y' \land y)$$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V'\ \ R \wedge p[\ V'\ /V\ ]$$

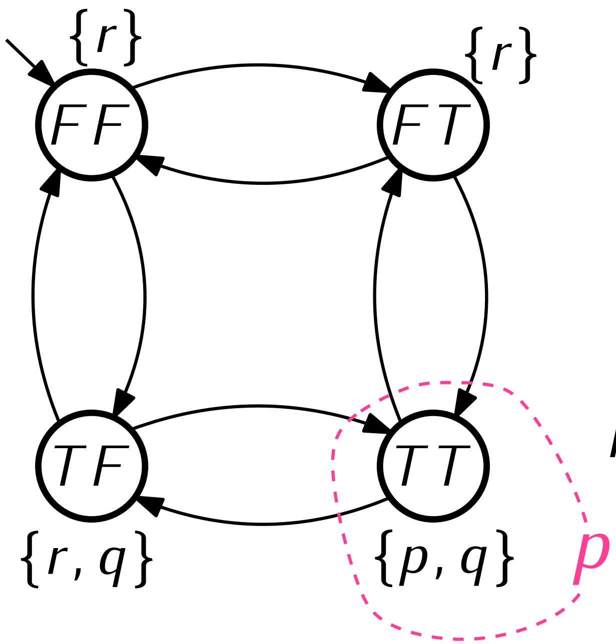
$$EX\ p \equiv \exists x', y'\ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \wedge (x' \wedge y')$$

...some Boolean simplifications ...

$$EX\ p \equiv \exists x', y'\ \ (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$$

...existential quantifer elimination ...

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \to \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ R \wedge p[\ V' / V\ ]$$

$$EX\ p \equiv \exists x', y' \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$$

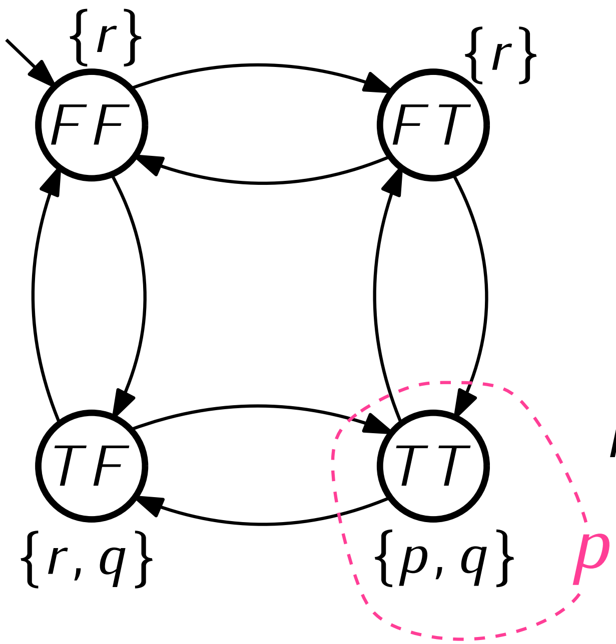
…some Boolean simplifications …

$$EX\ p \equiv \exists x', y' \ (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$$

…existential quantifer elimination …

$$EX\ p \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$$

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V' \ / V\ ]$$

$$EX\ p \equiv \exists x', y' \ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$$

…some Boolean simplifications …

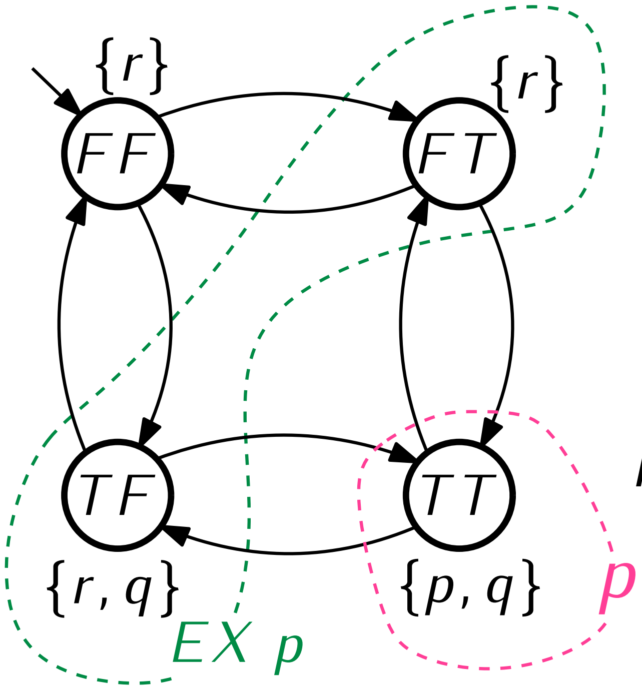
$$EX\ p \equiv \exists x', y' \ \ (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$$

…existential quantifer elimination …

$$EX\ p \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$$

Which states does this formula represent?

Symbolic Model Checking



Initial State: $\neg x \wedge \neg y$

Atomic Propositions: $AP = \{p, q, r\}$

Labelling Function $\mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EX\ p$

$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V' \ / V\ ]$$

$$EX\ p \equiv \exists x', y' \ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$$

...some Boolean simplifications ...

$$EX\ p \equiv \exists x', y' \ \ (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$$

...existential quantifer elimination ...

$$EX\ p \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$$

Which states does this formula represent?

Symbolic Model Checking

All of the boolean operations we have described for performing symbolic model checking (conjunction, disjunction, existential variable elimination) can be accomplished by:

1. Boolean algebra
2. Using BDDs
3. Using a theorem prover

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \exists V' \;\; R \;\wedge\; \phi[\, V' \,/\, V]$$

Example: $R \;\equiv\; (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

$$\phi \;\equiv\; p \;\equiv\; x \wedge y$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
  (exists ((x_ Bool) (y_ Bool))
    (and
      (or
        (and (= x_ x) (= y_ (not y)))
        (and (= x_ (not x)) (= y_ y)))
      (and x_ y_))))
(apply qe)
(check-sat)
```

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \boxed{\exists V'}\ R\ \wedge\ \phi[\ V'\ /\ V]$$

Example: $R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

$$\phi \equiv p \equiv x \wedge y$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
 (exists ((x_ Bool) (y_ Bool))
   (and
     (or
       (and (= x_ x) (= y_ (not y)))
       (and (= x_ (not x)) (= y_ y)))
     (and x_ y_))))
(apply qe)
(check-sat)
```

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \exists V'\ \boxed{R} \wedge\ \phi[\ V'\ /\ V]$$

Example: $R \equiv \boxed{(x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)}$

$$\phi \equiv p \equiv x \wedge y$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
  (exists ((x_ Bool) (y_ Bool))
    (and
      (or
        (and (= x_ x) (= y_ (not y)))
        (and (= x_ (not x)) (= y_ y)))
      (and x_ y_))))
(apply qe)
(check-sat)
```

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \exists V' \ \ R \ \wedge \ \boxed{\phi[\ V' \ / \ V]}$$

Example: $R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

$$\phi \equiv \boxed{p \equiv x \wedge y}$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
  (exists ((x_ Bool) (y_ Bool))
    (and
      (or
        (and (= x_ x) (= y_ (not y)))
        (and (= x_ (not x)) (= y_ y)))
      (and x_ y_))))
(apply qe)
(check-sat)
```

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \exists V'\ \ R\ \ \boxed{\wedge}\ \phi[\ V'\ /\ V]$$

Example: $R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

$$\phi \equiv p \equiv x \wedge y$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
  (exists ((x_ Bool) (y_ Bool))
    (and
      (or
        (and (= x_ x) (= y_ (not y)))
        (and (= x_ (not x)) (= y_ y)))
      (and x_ y_))))
(apply qe)
(check-sat)
```

Symb. Mod. Check. using a Theorem Prover

We can translate the $EX\ \phi$ formula into Z3.

$$EX\ \phi \equiv \exists V' \;\; R \;\wedge\; \phi[\ V' \ /\ V]$$

Example: $R \;\equiv\; (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

$$\phi \;\equiv\; p \;\equiv\; x \wedge y$$

```
(declare-const x Bool)
(declare-const y Bool)
(assert
  (exists ((x_ Bool) (y_ Bool))
    (and
      (or
        (and (= x_ x) (= y_ (not y)))
        (and (= x_ (not x)) (= y_ y)))
      (and x_ y_))))
(apply qe)
(check-sat)
```