CS 181u Applied Logic

Lecture 16

Reasoning About Knowledge

Example: Muddy Children

There are N children playing outside.

Some number k of the children have gotten mud on their foreheads.

Teacher says the following:

- 0. 'At least one of you has mud on their forehead. "
- 1. "Does any of you know that you have mud on your own forehead?"
- 2. "Does any of you know that you have mud on your own forehead?"
- 3. ...

Assume that the children are perceptive, intelligent, truthful, and that they all answer in unison.

What will happen?

Example: Muddy Children

Theorem: For the first $(k-1)$ questions, all children will answer "no", and on the k-th question, all children with muddy foreheads will answer "yes".

Proof.

Excercise: Can be proved by induction on k.

Example: Muddy Children

Weirdness:
Assume $k > 1$.
Let $p =$ "at least one child has a muddy forehead".
Then, initially, all children already know p.

Question:
If the teacher had not said "at least one of you has a muddy forehead", but then still asked:

- 1. "Does any of you know that you have mud on your own forehead?"
- 2. "Does any of you know that you have mud on your own forehead?"
- 3. ...

could the children still have deduced who has a muddy forehead?

In other words, did the teacher give the children any new information?

Example: Muddy Children

Theorem: If the teacher does not announce p, then no children can ever conclude that their own foreheads are muddy.

Proof: Induction on q, the number of questions.

Claim: no matter how many children have a muddy forehead, all the children answer "No" to the teacher's first q questions.

Base case, $q = 1$: all the children answer "No" to the first question, since a child cannot tell apart a situation where they have mud on their forehead from one that is identical in all respects except that they do not have a muddy forehead.

Induction: By IH, children answer "No" to first q questions.

Upon the $(q + 1)$th question, child i still cannot tell apart a situation where they have mud on their forehead from one that is identical in all respects except that they do not have a muddy forehead, since by the IH, the children will answer "No" to the first q questions whether or not child i has a muddy forehead.

Thus, again, they do not know whether their own forehead is muddy.

Example: Muddy Children

Theorem: If the teacher does not announce p, then no children can ever conclude that their own foreheads are muddy.

Weirdness: By announcing something that the children already know, p, the teacher somehow provided some information. What is that information?

all children know p AND all children know that all children know p

Example: Aces and Eights

- A simple game that involves some sophisticated reasoning about knowledge.
- Deck consisting of just four aces and four eights.
- Three players.
- Six cards are dealt out, two to each player.
- The remaining two cards are left face down.
- Without looking at the cards, each of the players raises them up to his or her forehead, so that the other two players can see them but they cannot.
- All of the players take turns trying to determine which cards they're holding.
- If a player does not know which cards they are holding, the player must say so.

Example: Aces and Eights

Scenario 1:

Alice goes first, holds two aces.

Bob, goes second, holds two eights.

Both Alice and Bob say that they cannot determine what cards they are holding.

What cards are you holding?

(Hint: consider what would have happened if you held two aces or two eights.)

Example: Aces and Eights

Scenario 2:

You go first.

Alice goes second, holds two eights.

Bob, goes third, holds an ace and an eight.

No one is able to determine what they hold at their first turn.

What cards are you holding?

(Hint: consider what would have happened if you held two aces or two eights.)

Example: Aces and Eights

Scenario 3:

Alice goes first, holds two eights.

You go second.

Bob, goes third, holds an ace and an eight.

No one is able to determine what they hold at their first turn.

Alice cannot determine her cards at her second turn.

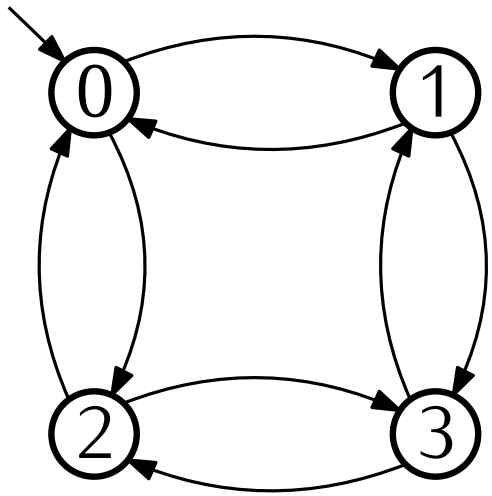
What cards are you holding?

Explicit Model Representation

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

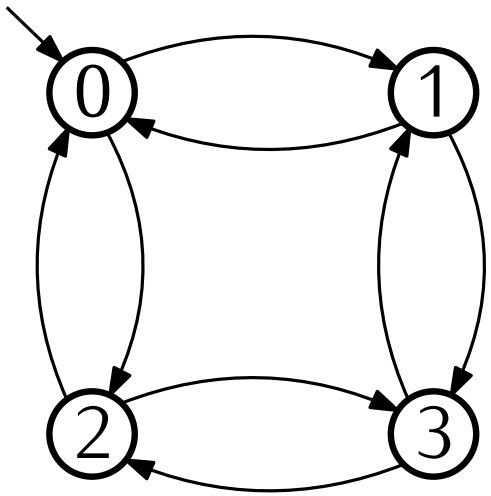
Explicit Model Representation

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



Explicit Model Representation

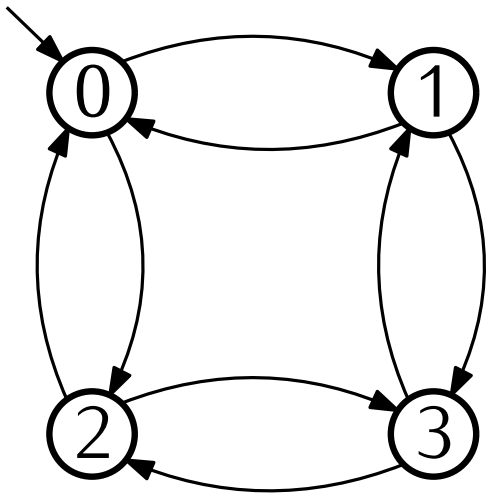
The transition system $\mathcal{M}$ is specified by literally listing out all of the pieces.



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

Explicit Model Representation

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

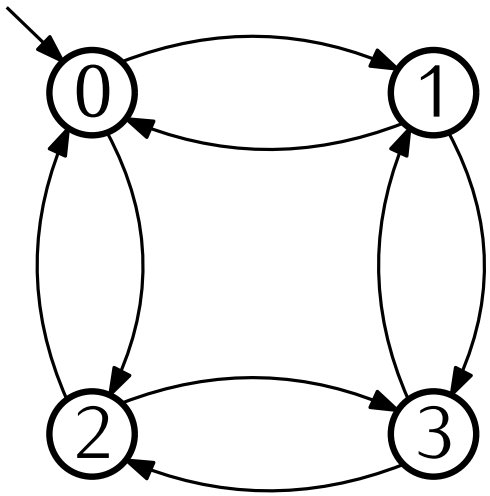


States: $S = \{0, 1, 2, 3\}$

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

Explicit Model Representation

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



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

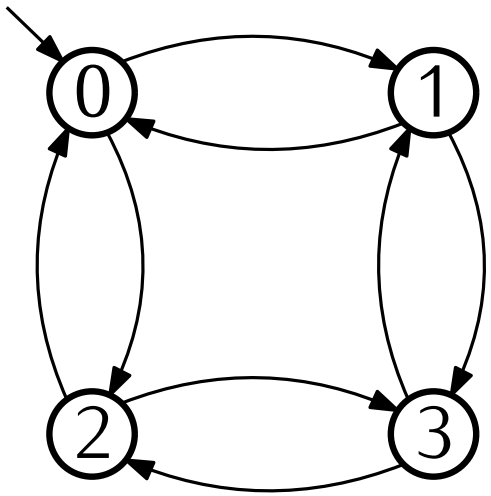
Initial States: $I = \{0\}$

Transitions:

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

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: $I = \{0\}$

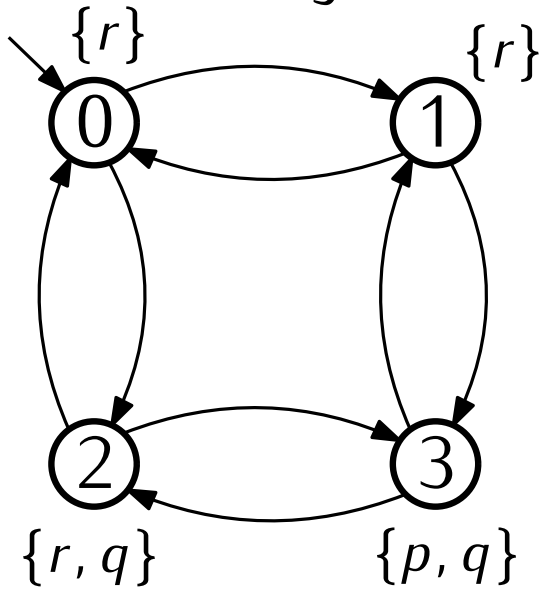
Transitions:

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

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

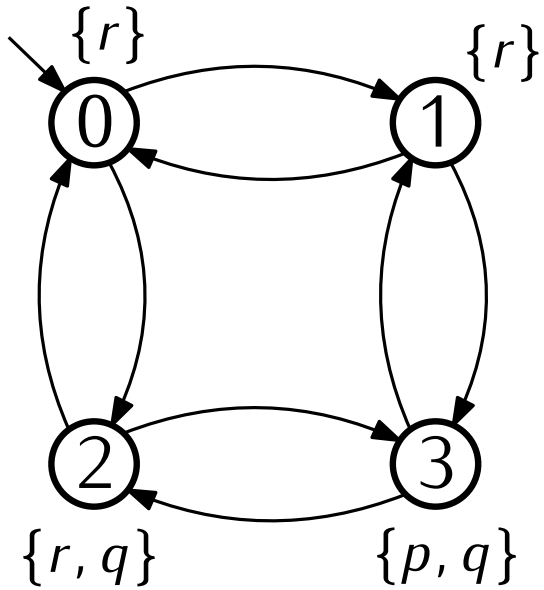
Labelling Function $\mathcal{L} : S \to \mathcal{P}(AP)$

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

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

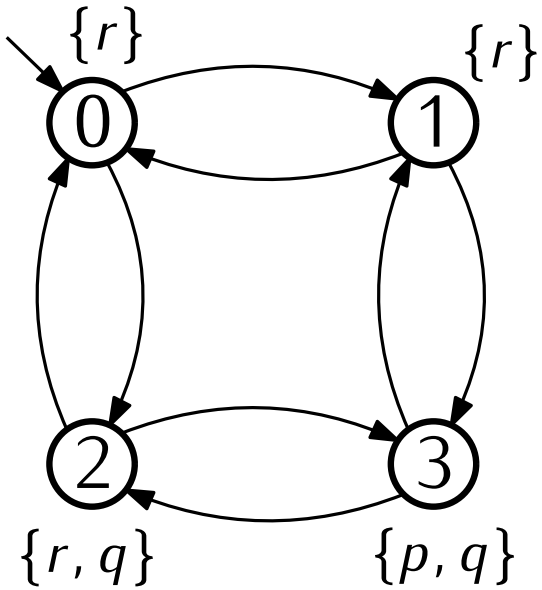
Symbolic Model Representation

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



Symbolic Model Representation

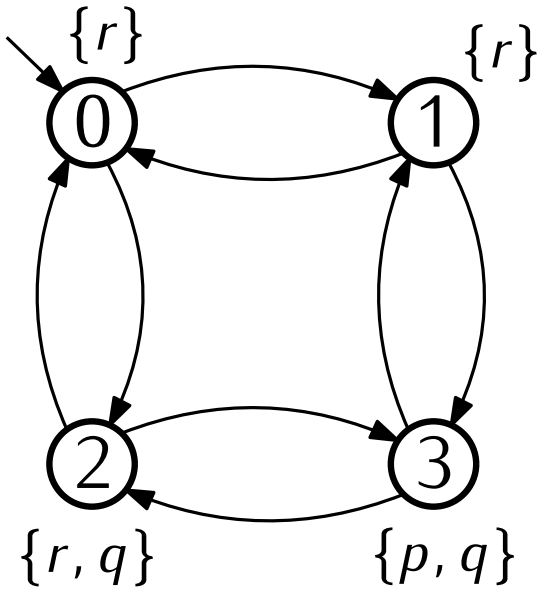
Represent $\mathcal{M}$ using Booelan logic.



States		
0		
1		
2		
3		

Symbolic Model Representation

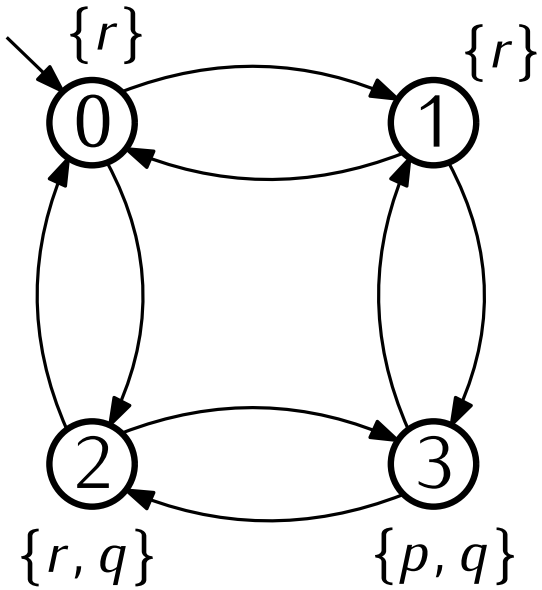
Represent $\mathcal{M}$ using Booelan logic.



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

Symbolic Model Representation

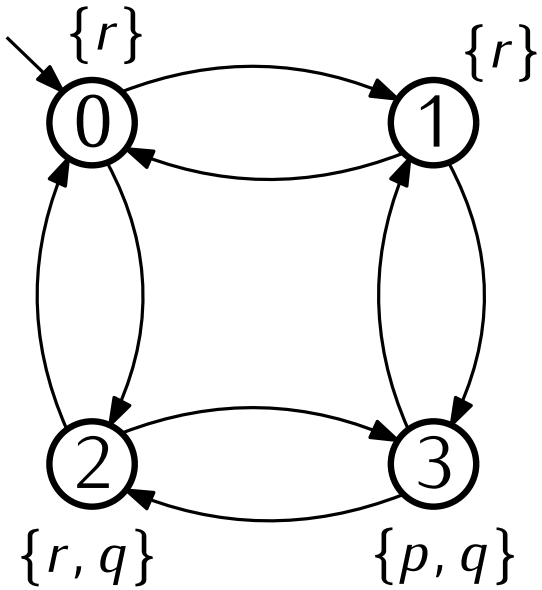
Represent $\mathcal{M}$ using Booelan logic.



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

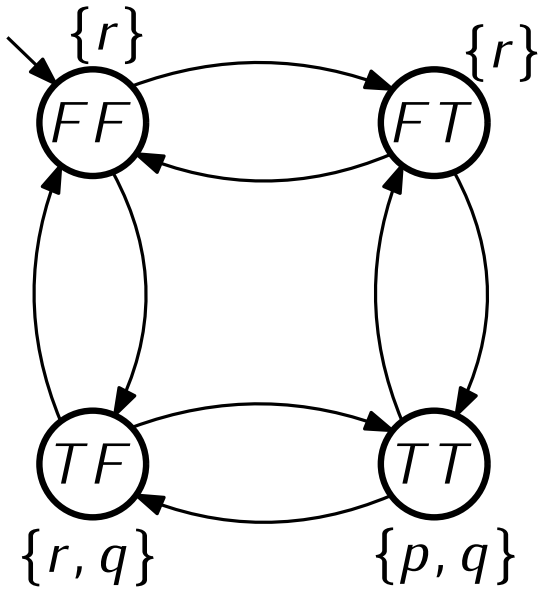
Symbolic Model Representation

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



Boolean state variables

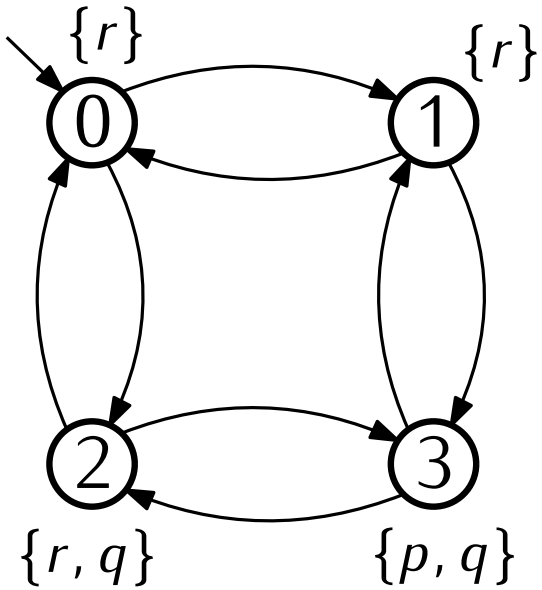
$$V = \{x, y\}$$



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

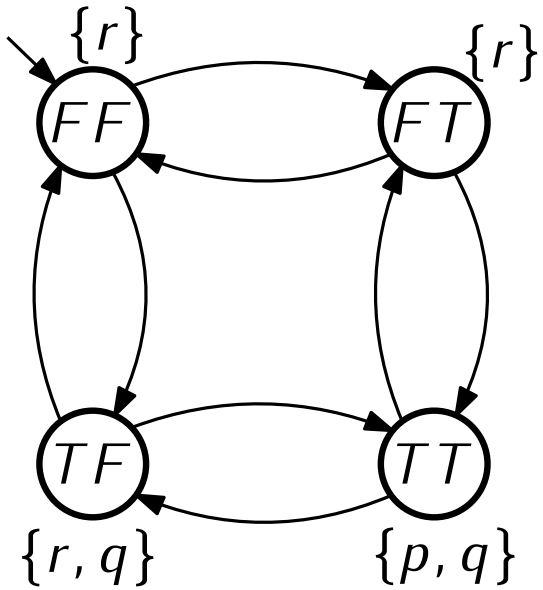
Symbolic Model Representation

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



Boolean state variables

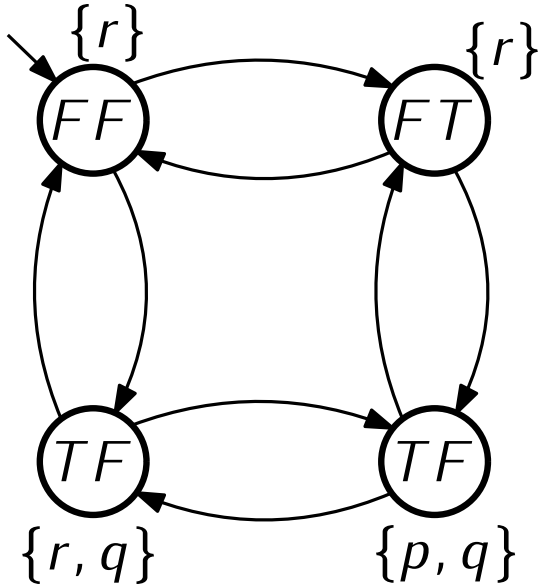
$$V = \{x, y\}$$



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

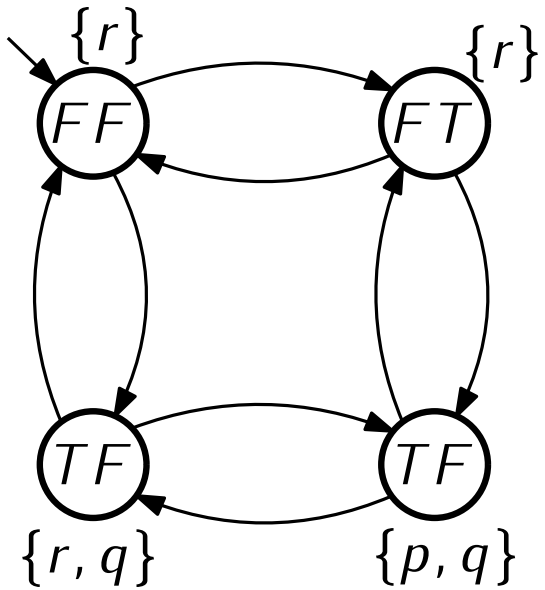
Represent $\mathcal{M}$ using Boolean logic.



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.

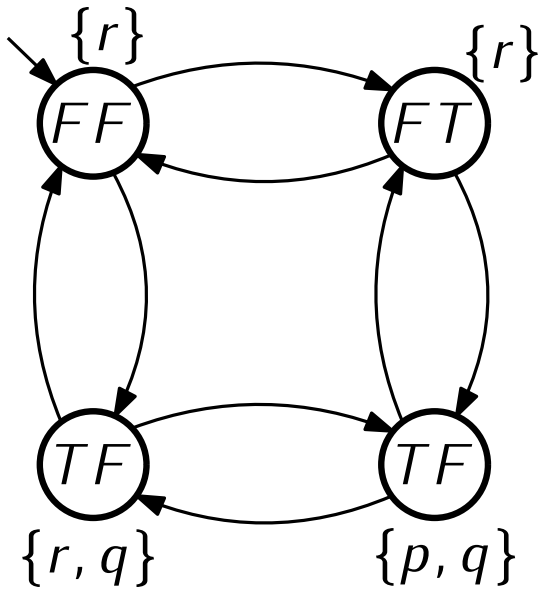


Initial State: $\neg x \wedge \neg y$

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.



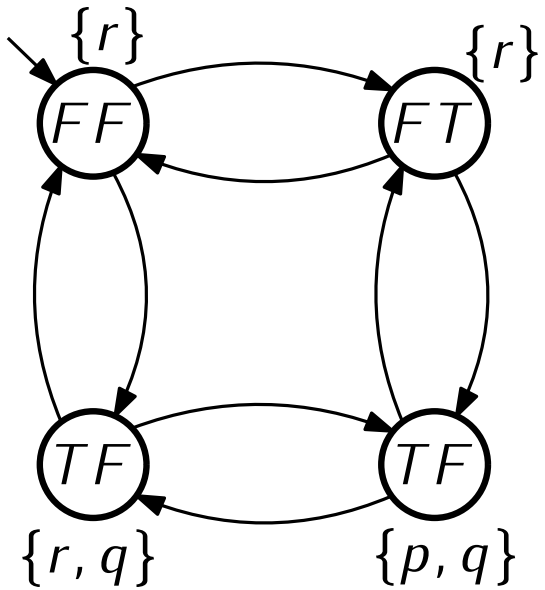
Initial State: $\neg x \wedge \neg y$

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: $\neg x \wedge \neg y$

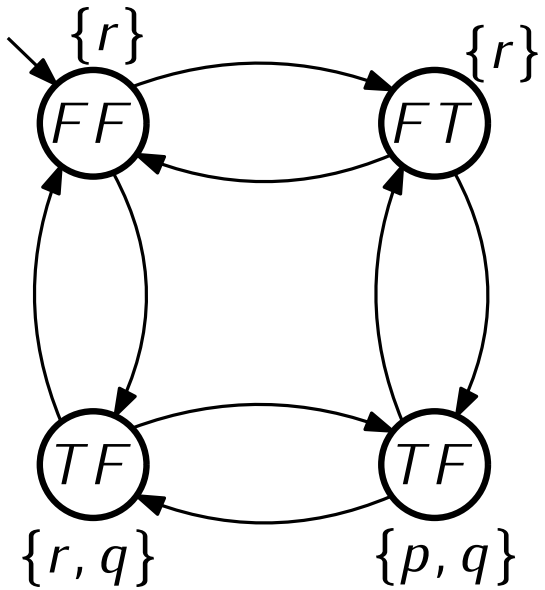
Atomic Propositions: $AP = \{p, q, r\}$

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: $\neg x \wedge \neg y$

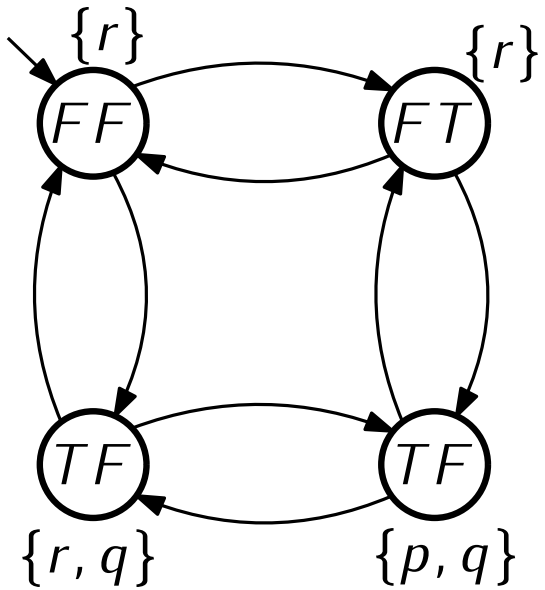
Atomic Propositions: $AP = \{p, q, r\}$

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

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

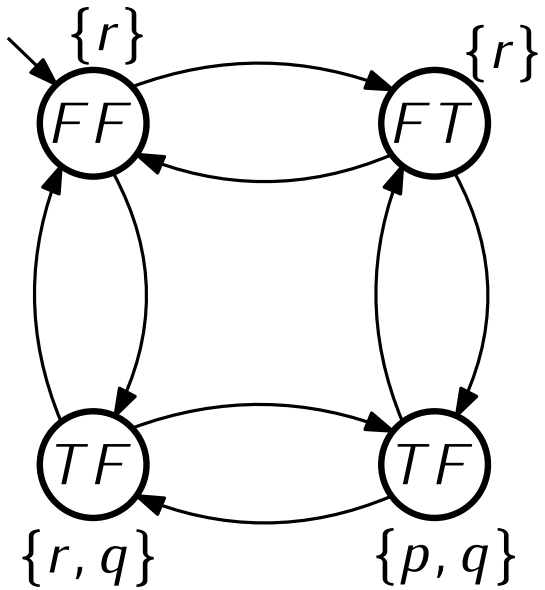
Labelling Function ~~$\mathcal{L} : S \to \mathcal{P}(AP)$~~

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

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

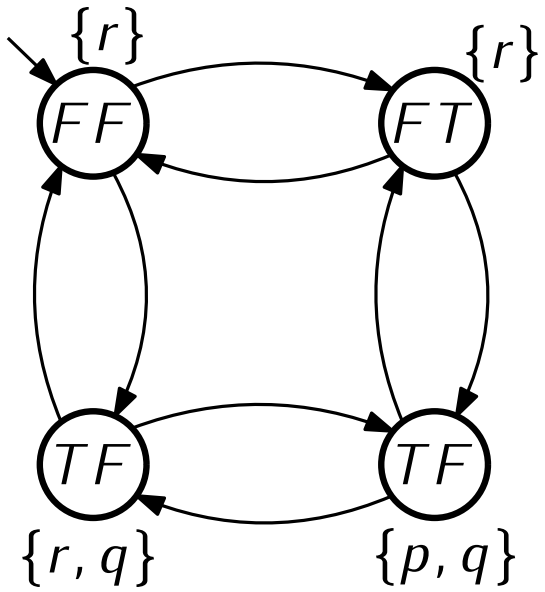
$\mathcal{L} : AP \to \mathcal{F}(x, y)$

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

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	$x \wedge y$

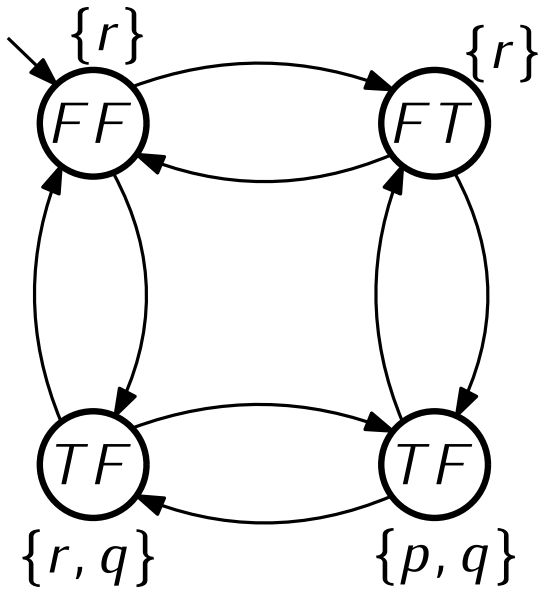
Symbolic Model Representation

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



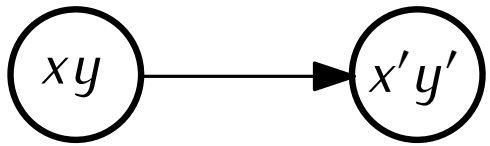
Symbolic Model Representation

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



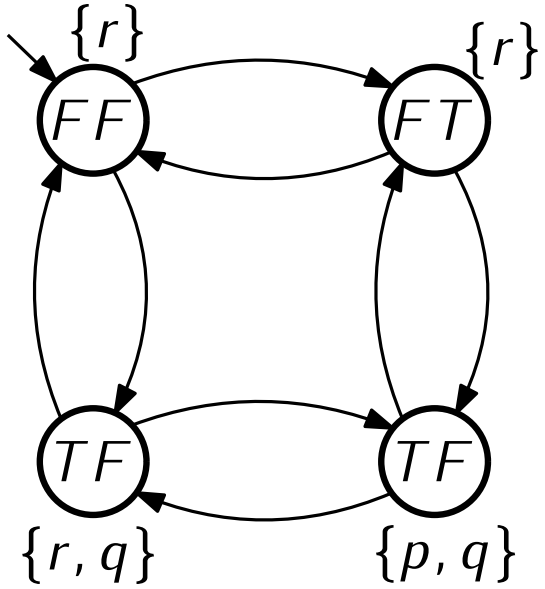
Transitions:

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



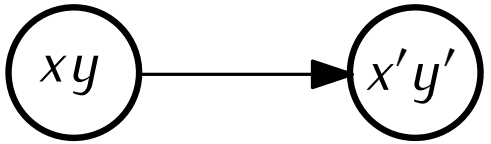
Symbolic Model Representation

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



Transitions:

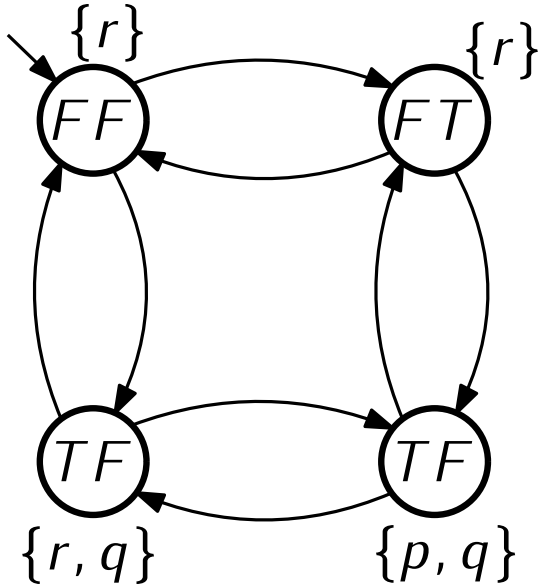
Let the "next" state variables be $V' = \{x', y'\}$



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

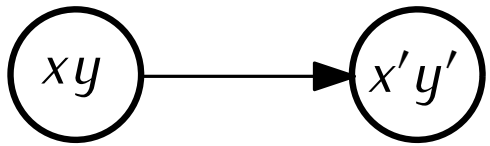
Symbolic Model Representation

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



Transitions:

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

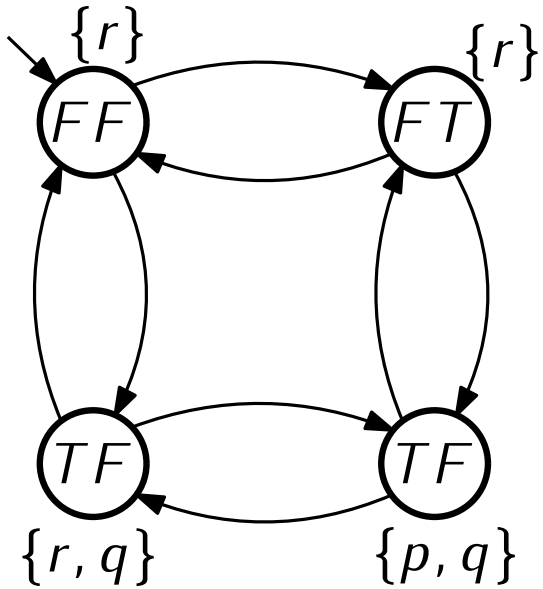


$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

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

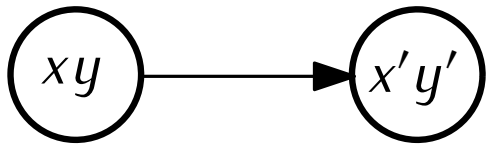
Symbolic Model Representation

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



Transitions:

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

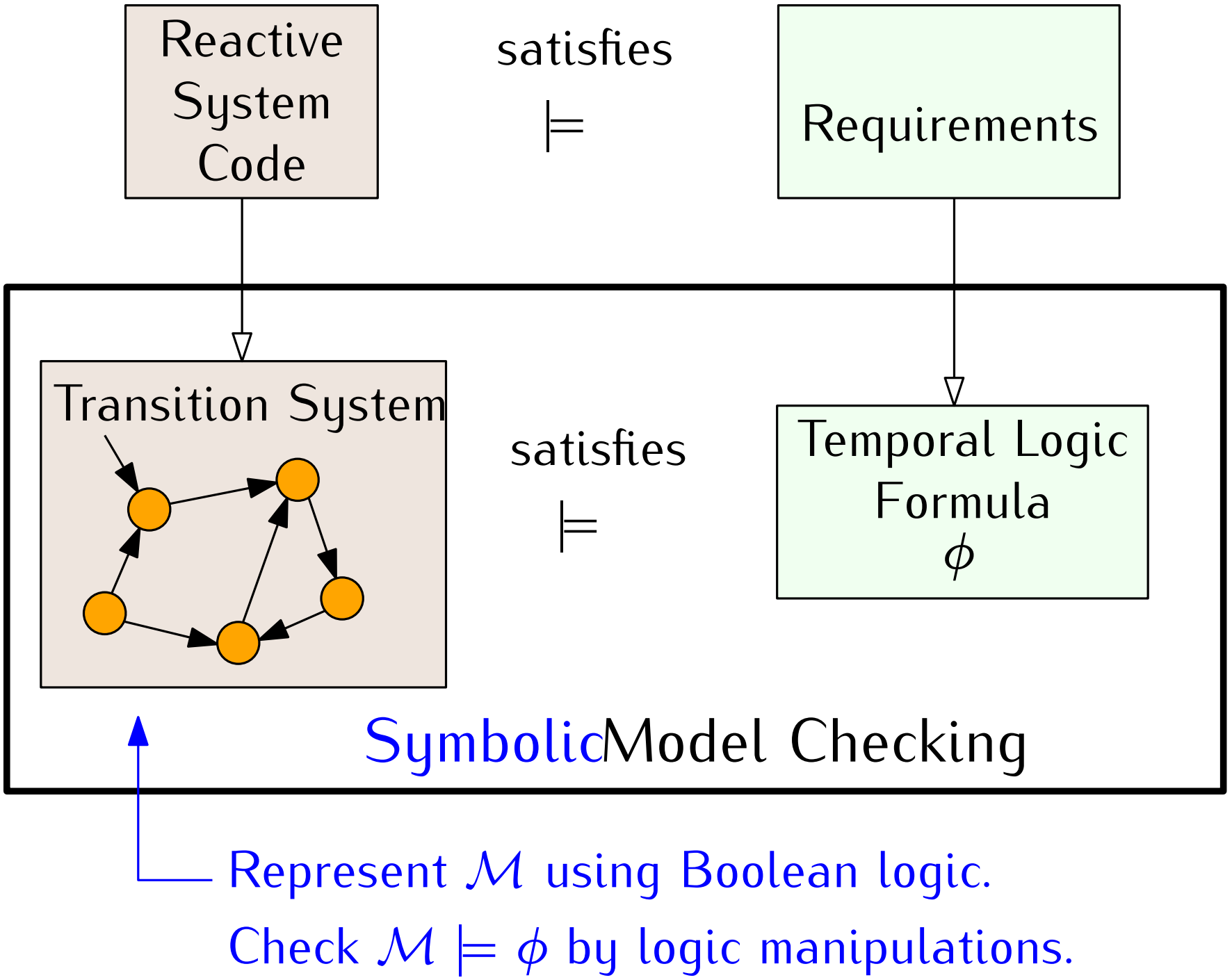


$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Explicit transitions

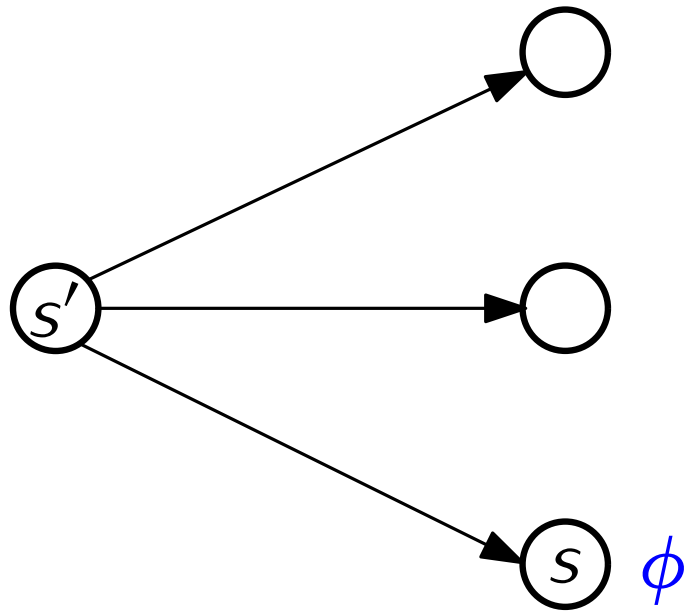
$(0,1)$ $(2,3)$ $(1,3)$ $(0,2)$
$(1,0)$ $(3,2)$ $(3,1)$ $(2,0)$

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


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

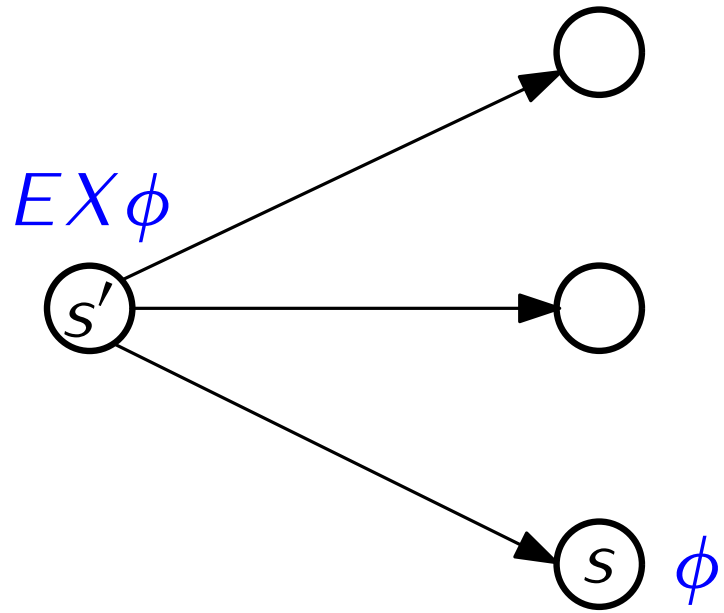
The Algorithm for $EX\ \phi$

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



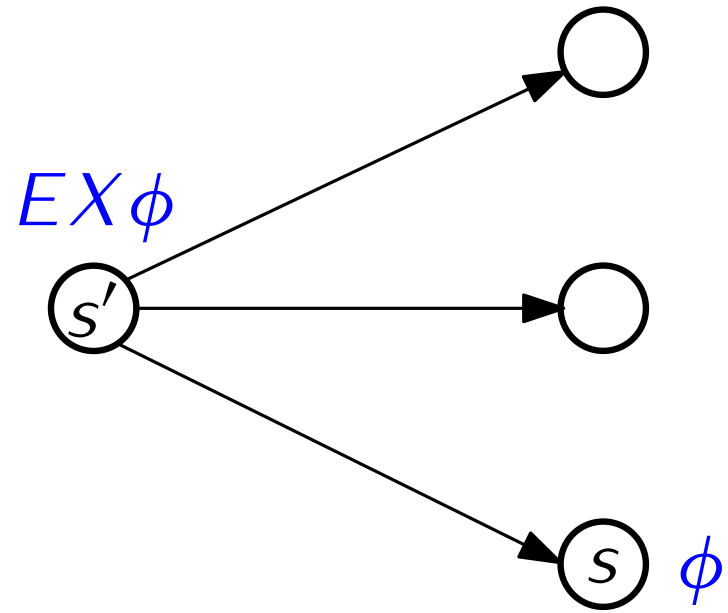
The Algorithm for $EX\ \phi$

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.

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

exists a path where ϕ holds in the next state

$\equiv$

Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

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

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.

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

exists a path where ϕ holds in the next state

$\equiv$

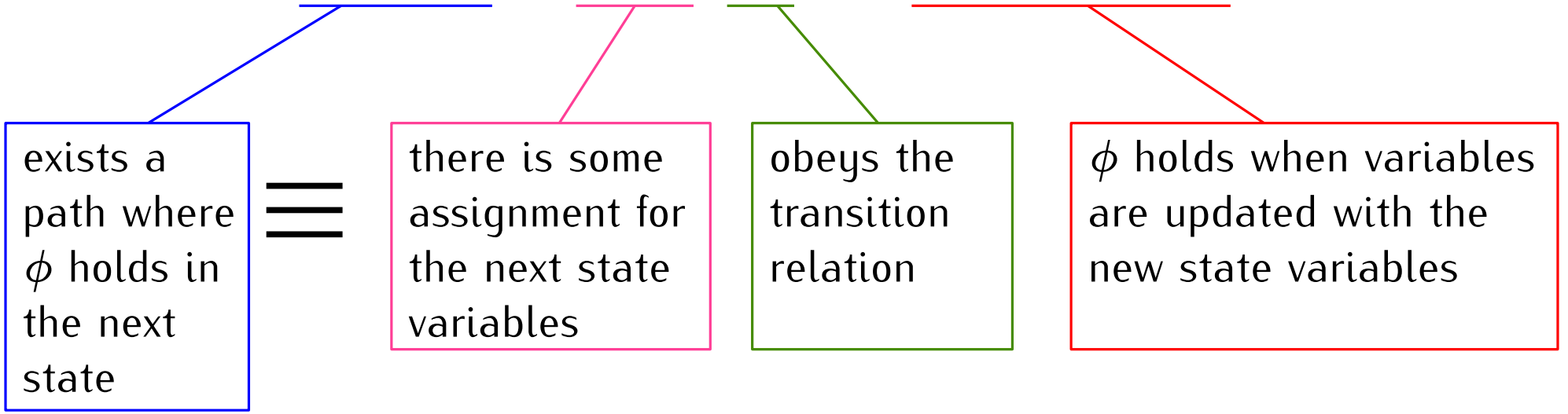
there is some assignment for the next state variables

obeys the transition relation

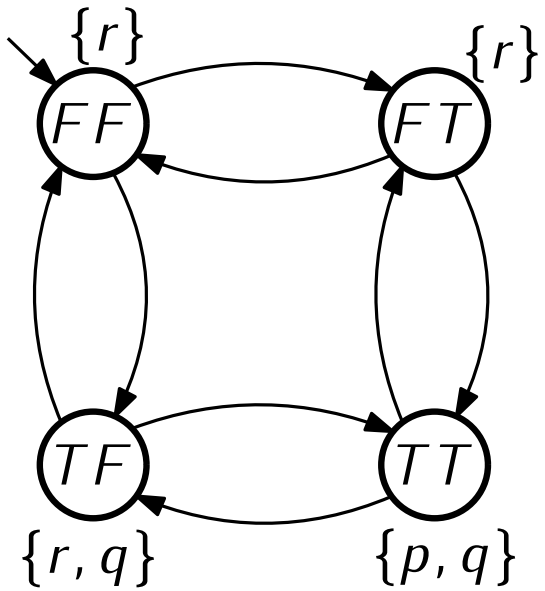
Symbolic Model Checking

How to compute $EX\ \phi$ symbolically.

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



Symbolic Model Checking



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

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

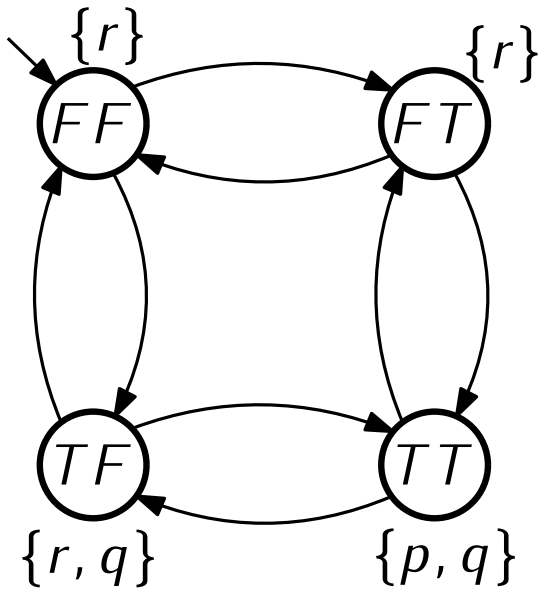
Labelling Function $\mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

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

Transition Relation:

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

Symbolic Model Checking



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

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

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

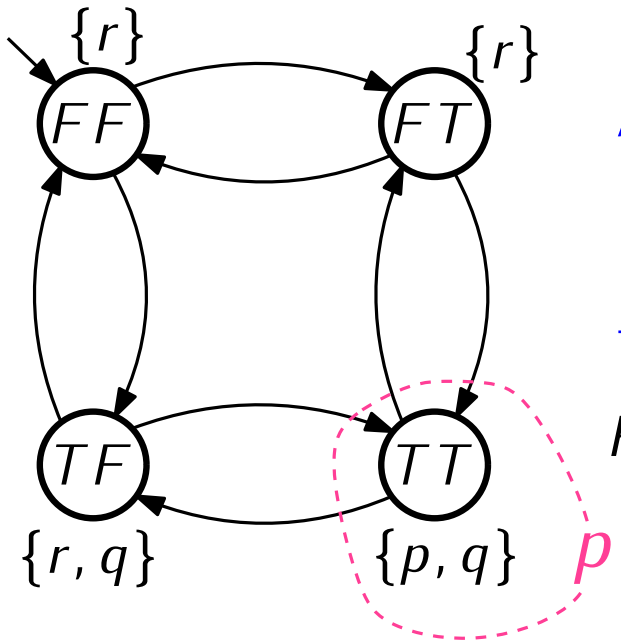
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

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

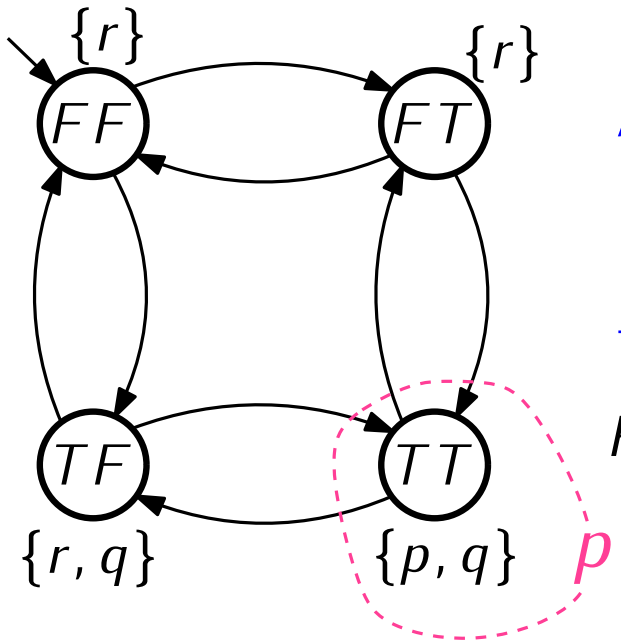
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

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

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

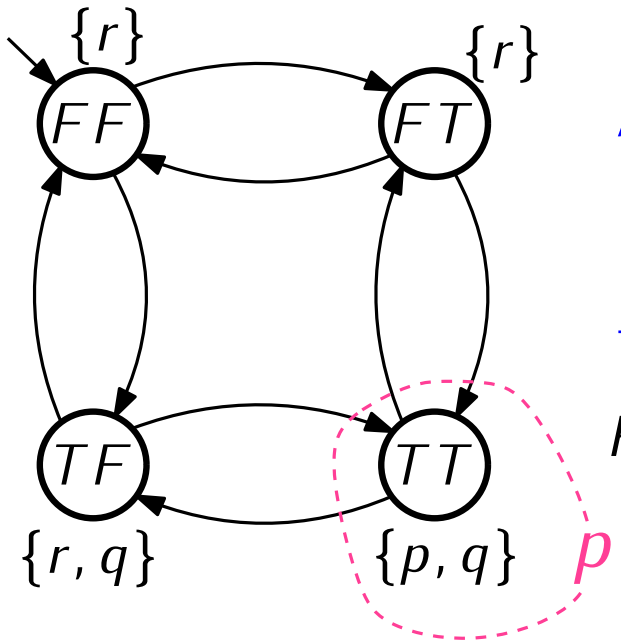
Transition Relation:

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

Let's compute $EX\ p$

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

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:

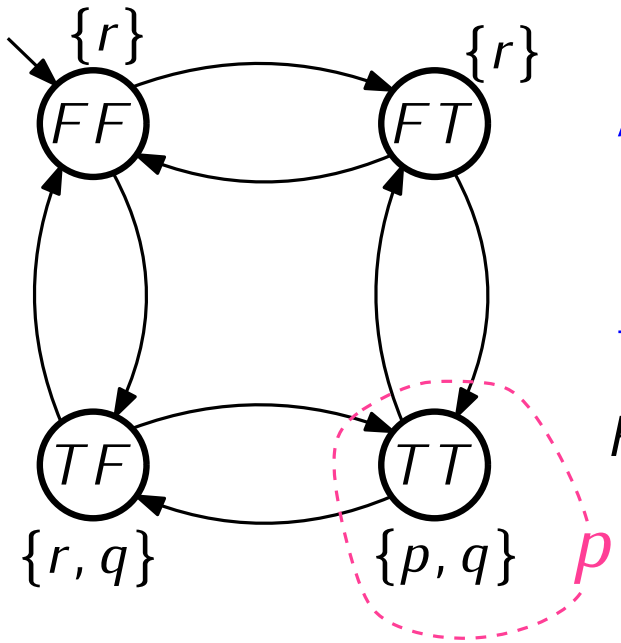
$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$

Let's compute $EX\ p$

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

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

Symbolic Model Checking



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

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

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

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

Transition Relation:

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

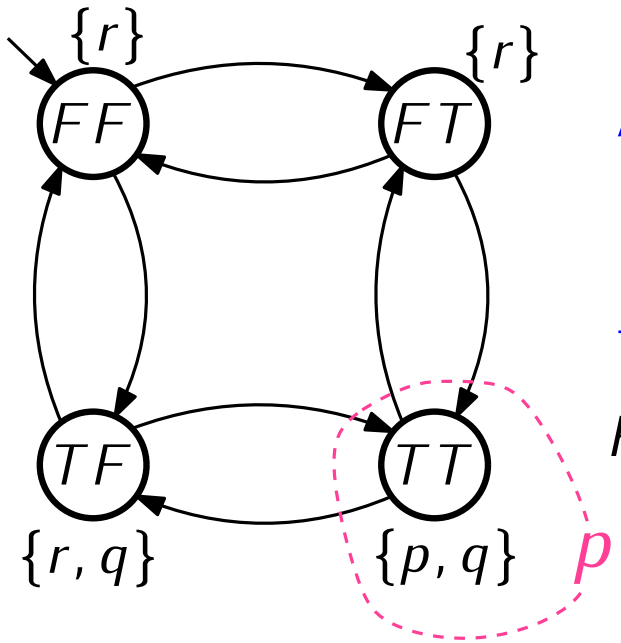
Let's compute $EX\ p$

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

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

...some Boolean simplifications ...

Symbolic Model Checking



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

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 $EX\ p$

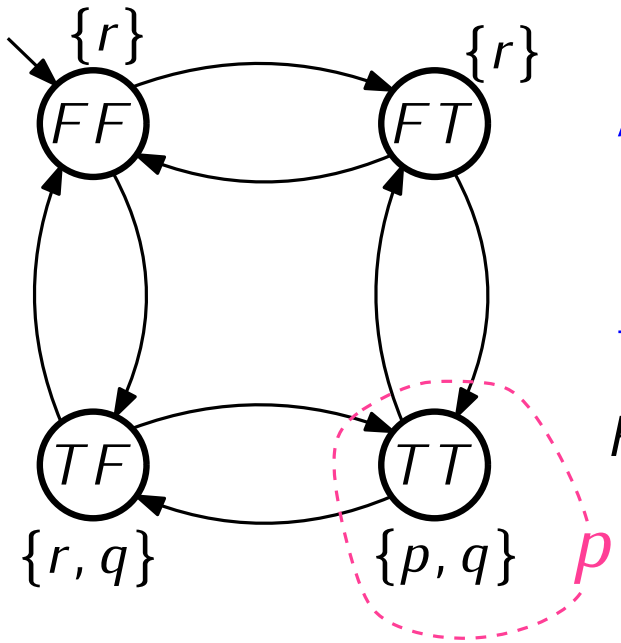
$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V'\ /V\ ]$

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

...some Boolean simplifications ...

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

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 $EX\ p$

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

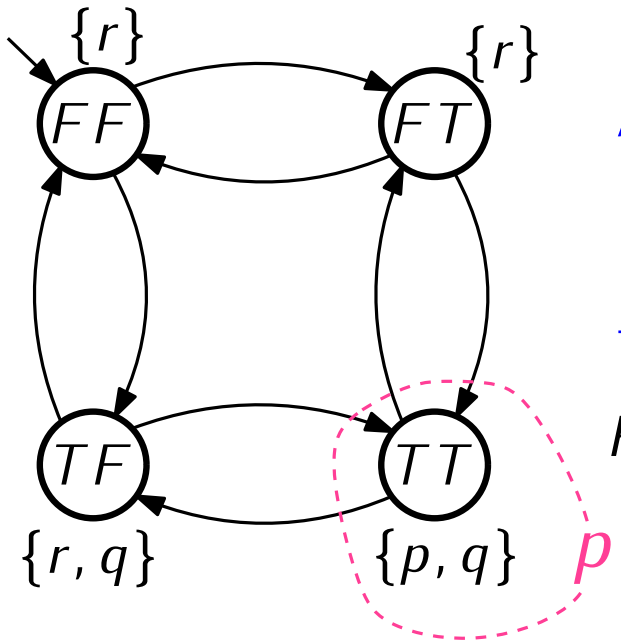
$EX\ p \equiv \exists x', y' \ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y) \ \wedge \ (x' \wedge y')$

...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 ...

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 $EX\ p$

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

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

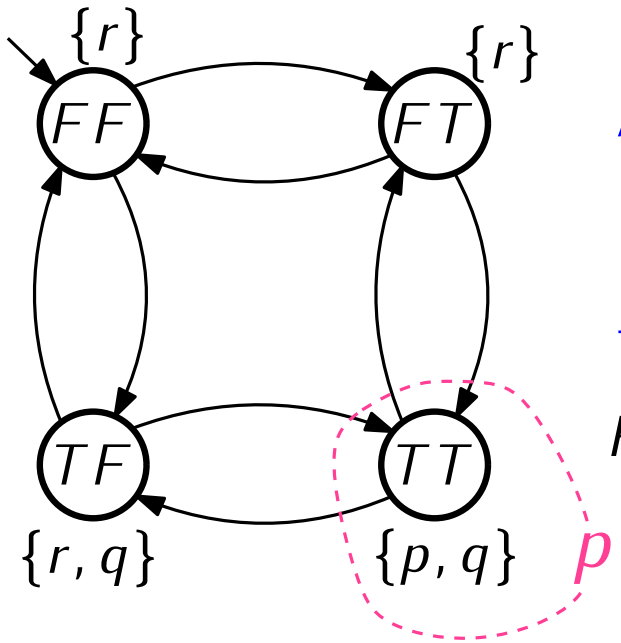
...some Boolean simplifications ...

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

...existential quantifer elimination ...

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

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 ...

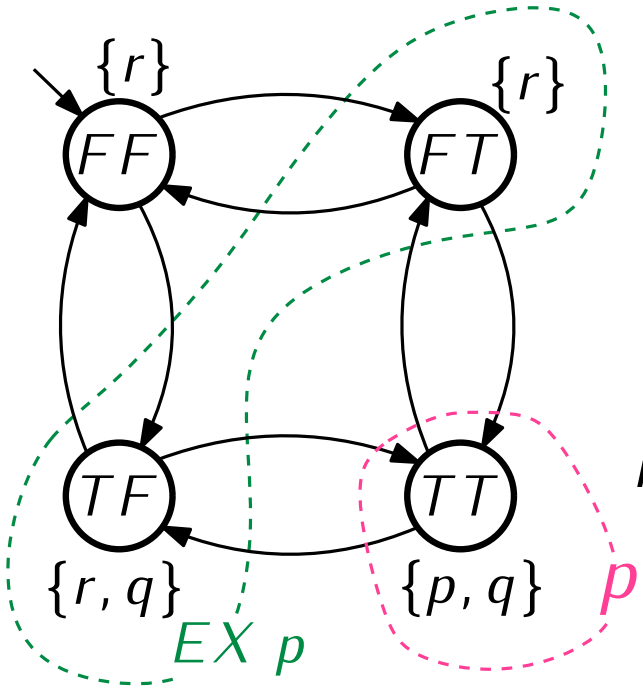
$EX\ p \equiv \exists x', y' \ \ (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$

... 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.

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

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.

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

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)
```