CS 181u Applied Logic

Lecture 14

Symbolic Model Checking

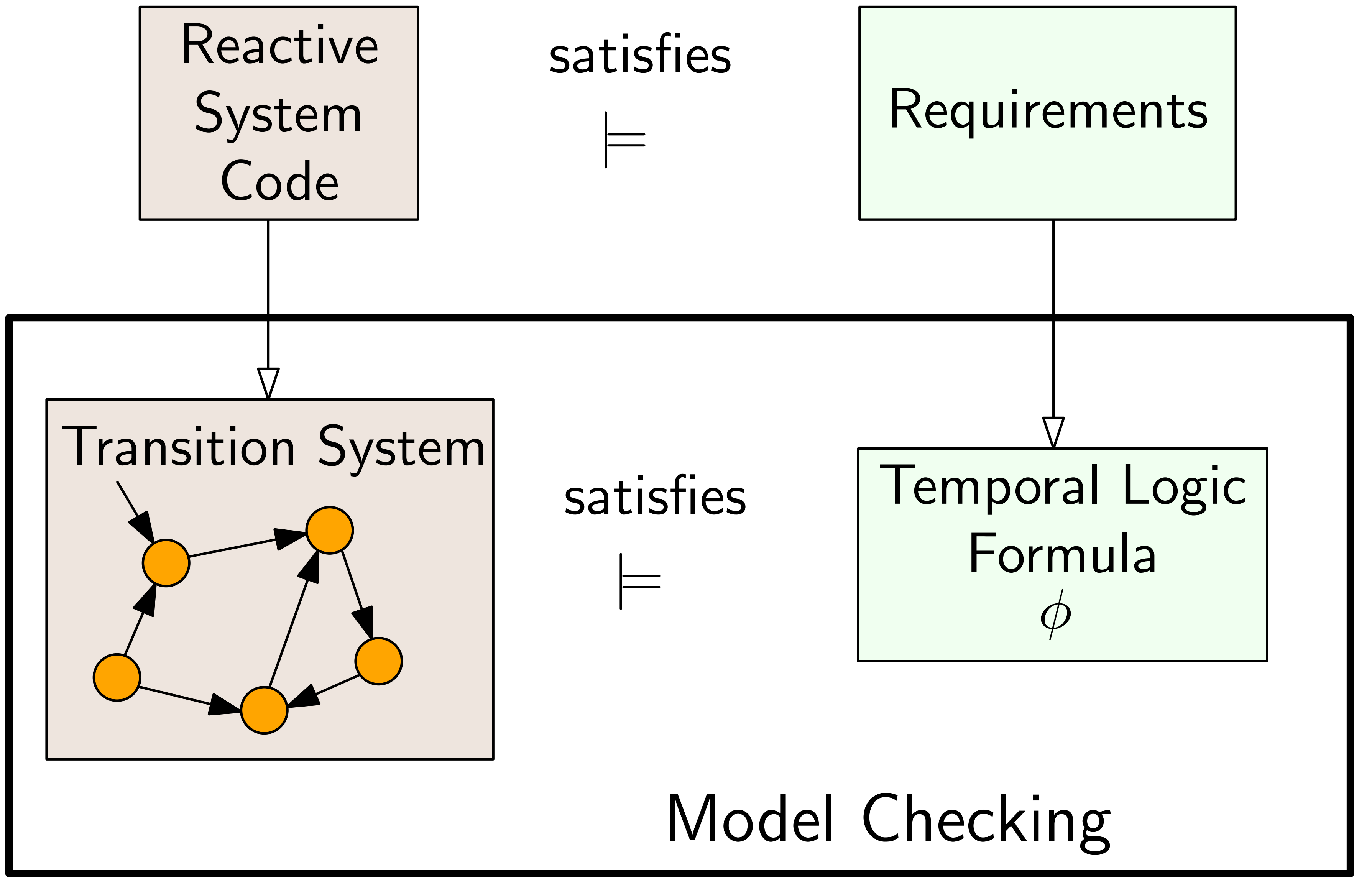
In Computer Aided Verification 2018

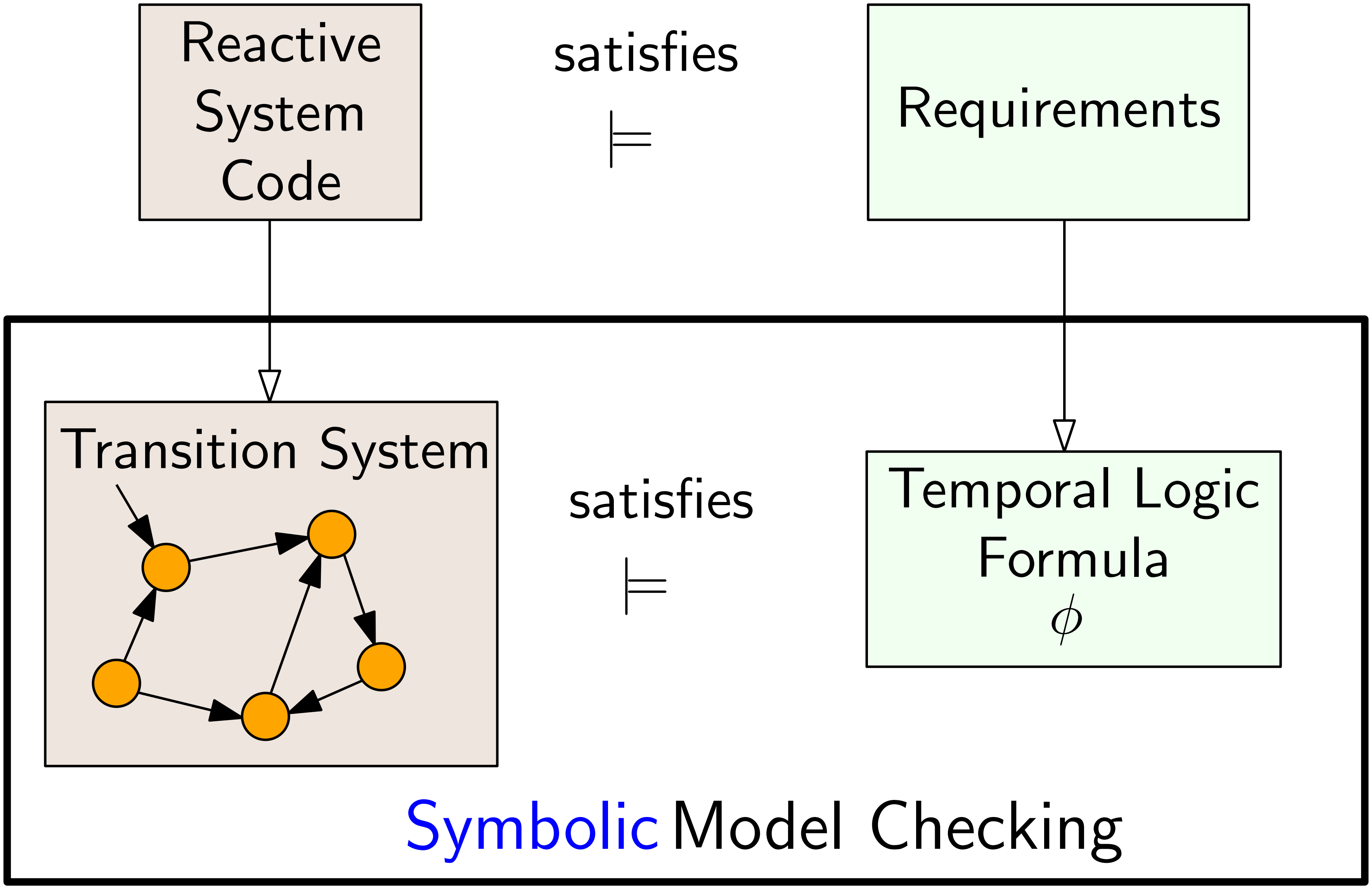
Algorithms for Model Checking HyperLTL and HyperCTL*

Bernd Finkbeiner[1], Markus N. Rabe[1], and César Sánchez[2]

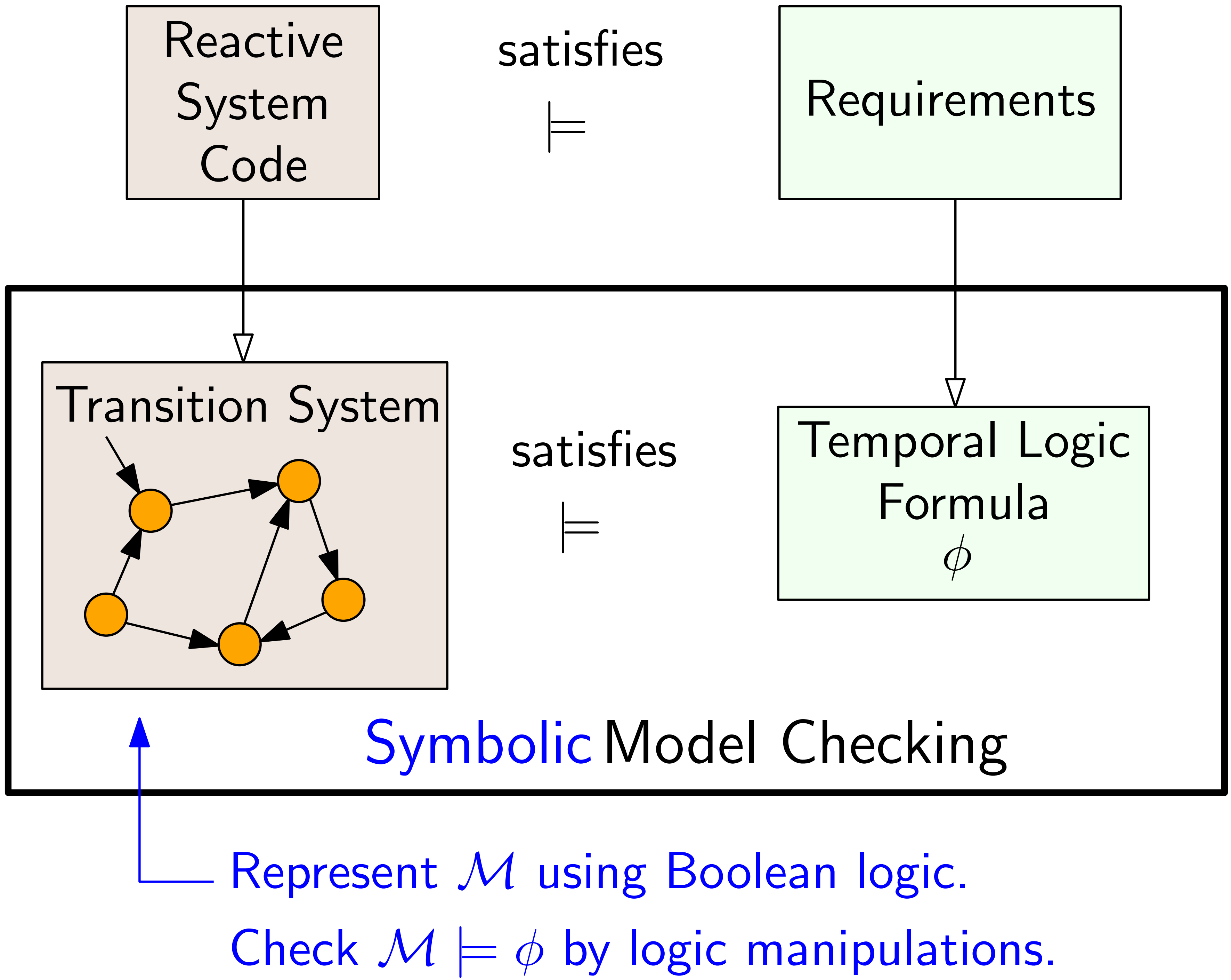
[1]Saarland University, [2]IMDEA Software Institute

Abstract. We present an automata-based algorithm for checking finite state systems for hyperproperties specified in HyperLTL and HyperCTL*. For the alternation-free fragments of HyperLTL and HyperCTL* the automaton construction allows us to leverage existing model checking technology. Along several case studies, we demonstrate that the approach enables the verification of real hardware designs for properties that could not be checked before. We study information flow properties of an I2C bus master, the symmetric access to a shared resource in a mutual exclusion protocol, and the functional correctness of encoders and decoders for error resistant codes.


Reactive System Code
satisfies
$\models$
Requirements
Transition System
satisfies
$\models$
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: $\quad f = \neg x \wedge \neg y$

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

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

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

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

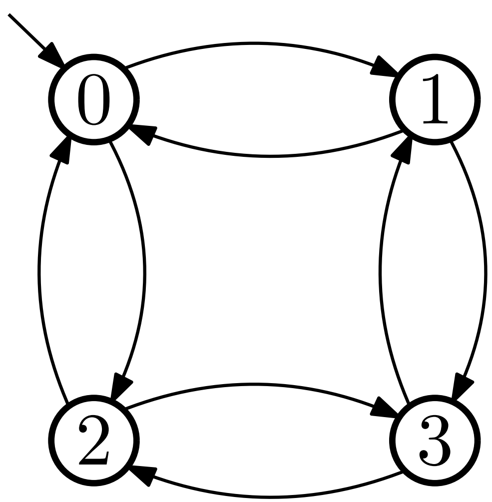
No more y

Explicit Model Representation

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

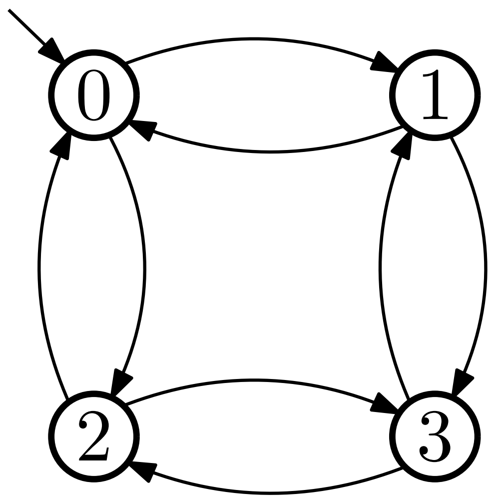
Explicit Model Representation

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



Explicit Model Representation

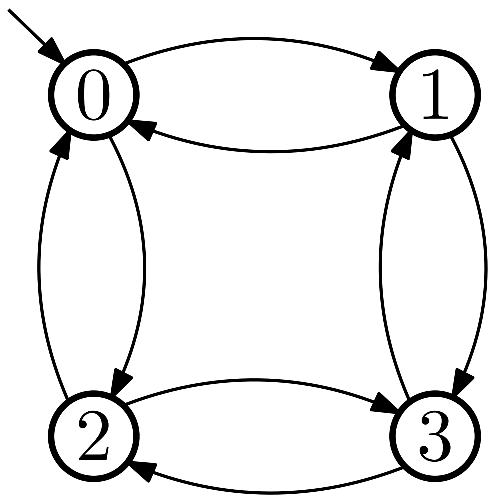
The transition system $\mathcal{M}$ is specified by literally listing out all of the pieces.



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

Explicit Model Representation

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

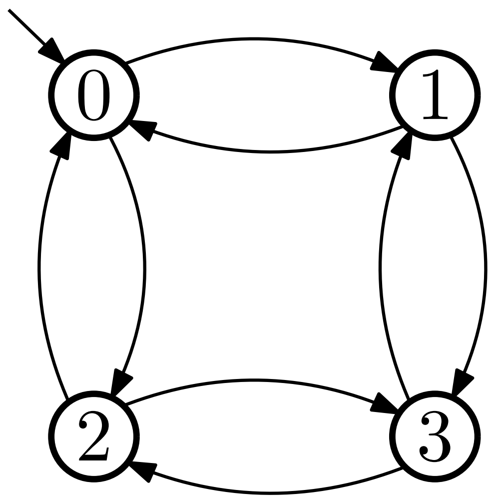


States: $S = \{0, 1, 2, 3\}$

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

Explicit Model Representation

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



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

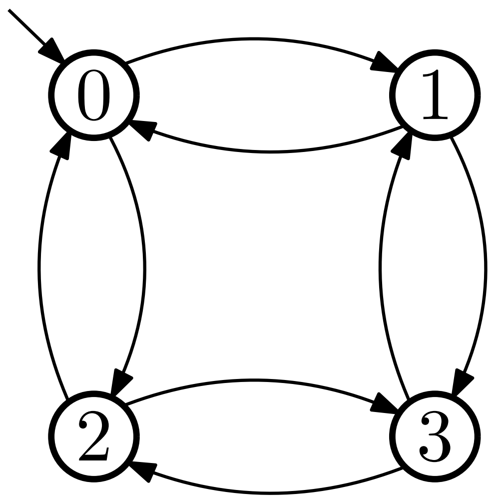
Initial States: $I = \{0\}$

Transitions:

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

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: $I = \{0\}$

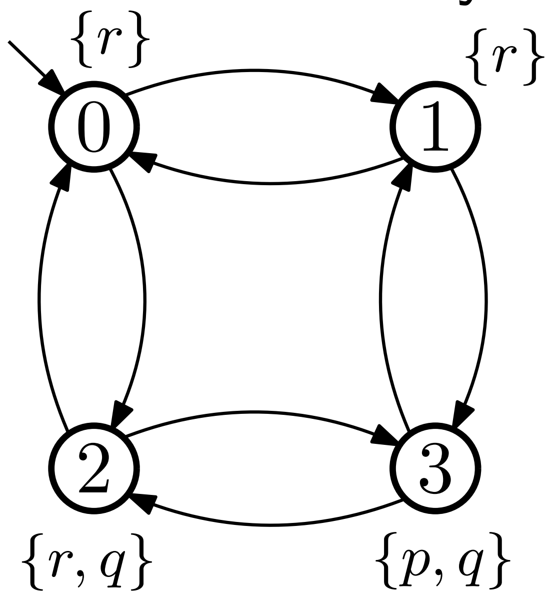
Transitions:

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

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:

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

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

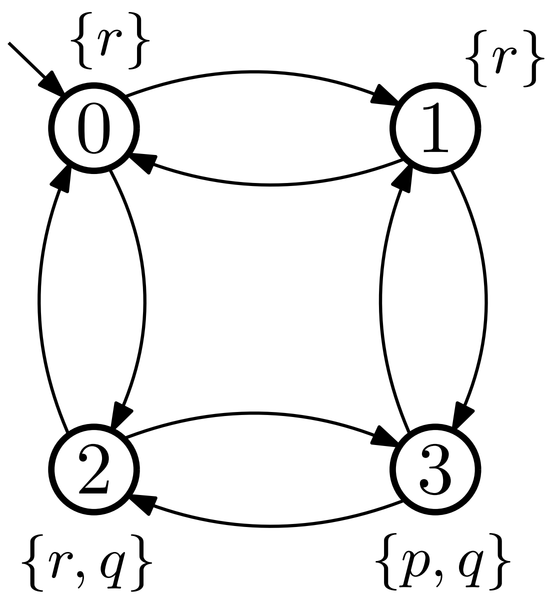
Labelling Function $\mathcal{L} : S \to \mathcal{P}(AP)$

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

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

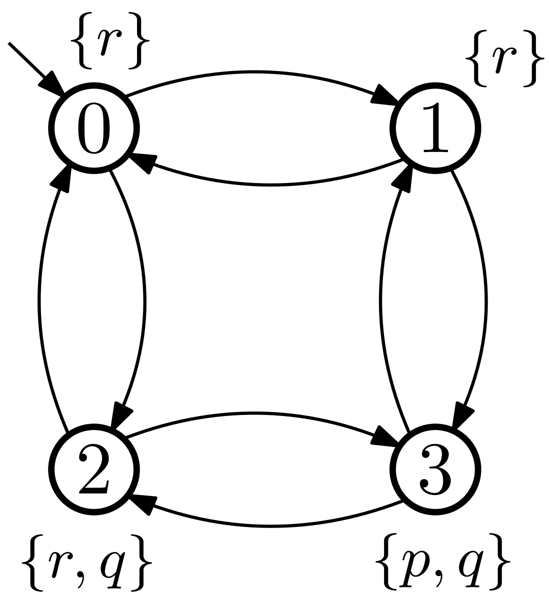
Symbolic Model Representation

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



Symbolic Model Representation

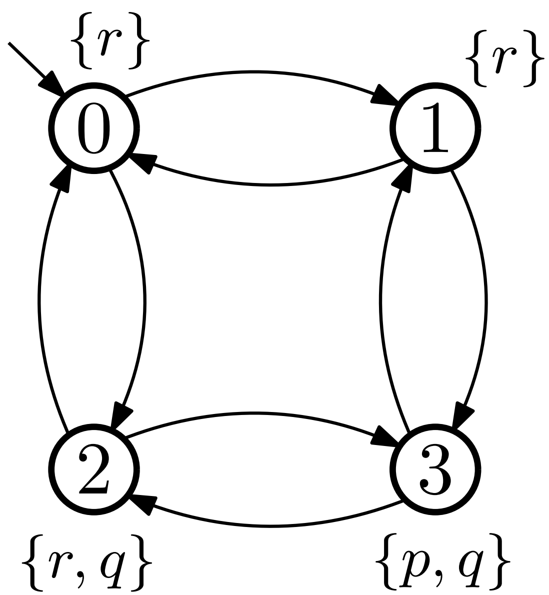
Represent $\mathcal{M}$ using Booelan logic.



States		
0		
1		
2		
3		

Symbolic Model Representation

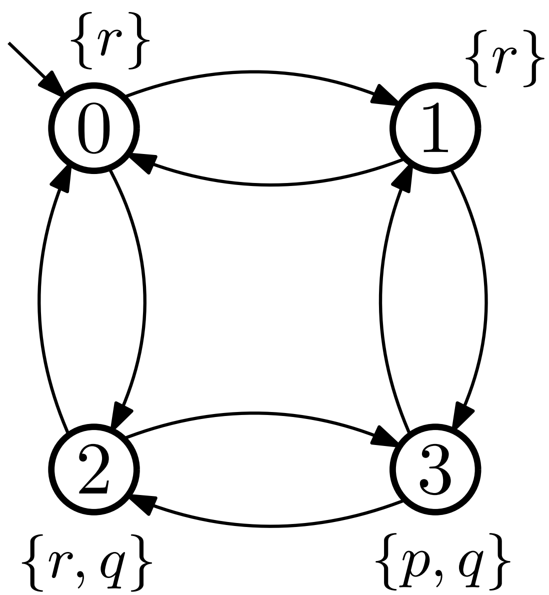
Represent $\mathcal{M}$ using Booelan logic.



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

Symbolic Model Representation

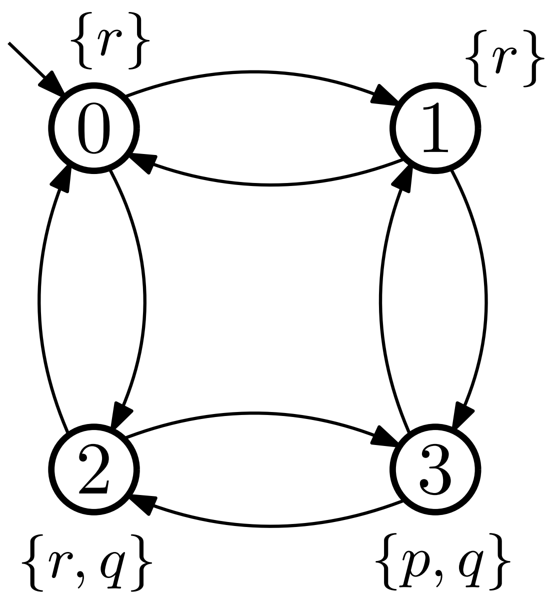
Represent $\mathcal{M}$ using Booelan logic.



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

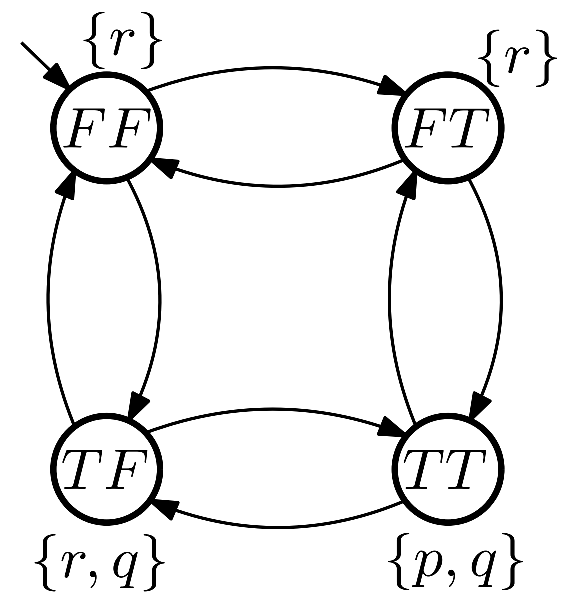
Symbolic Model Representation

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



Boolean state variables

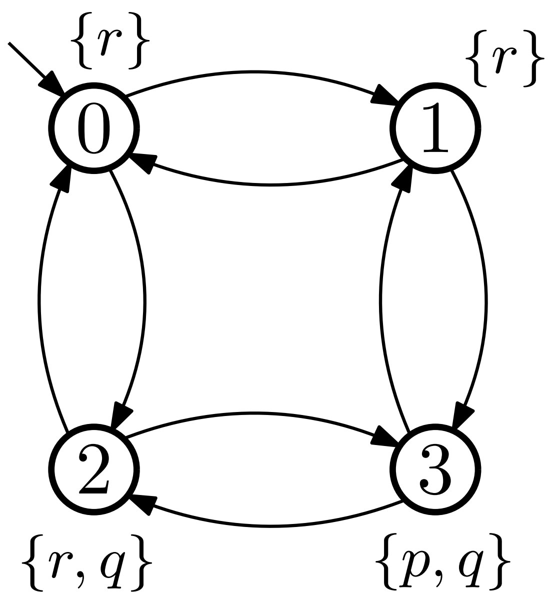
$$V = \{x, y\}$$



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

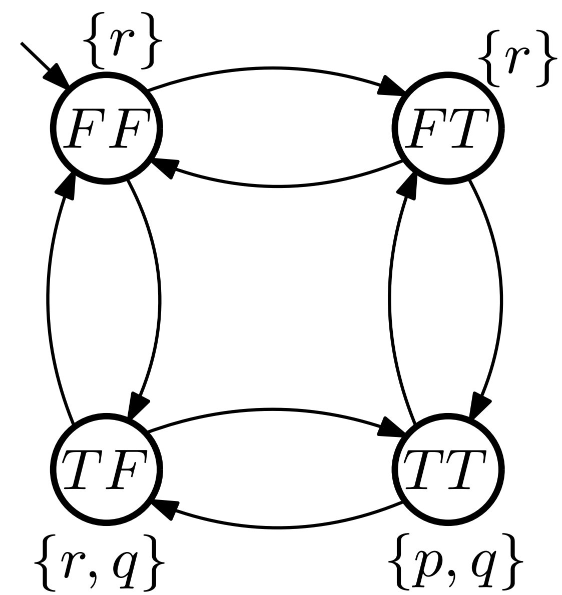
Symbolic Model Representation

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



Boolean state variables

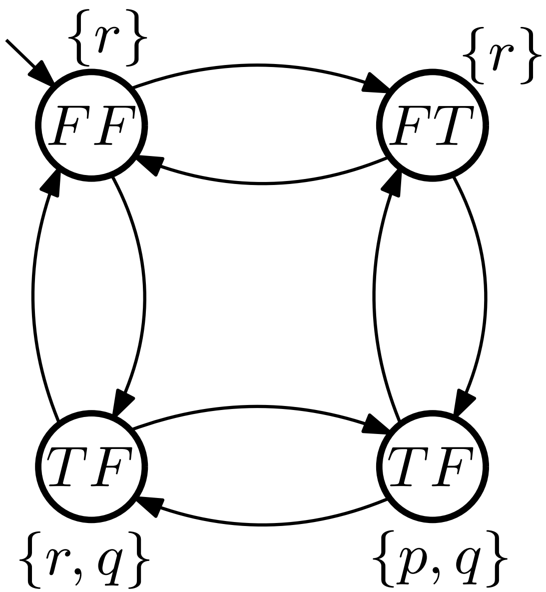
$$V = \{x, y\}$$



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

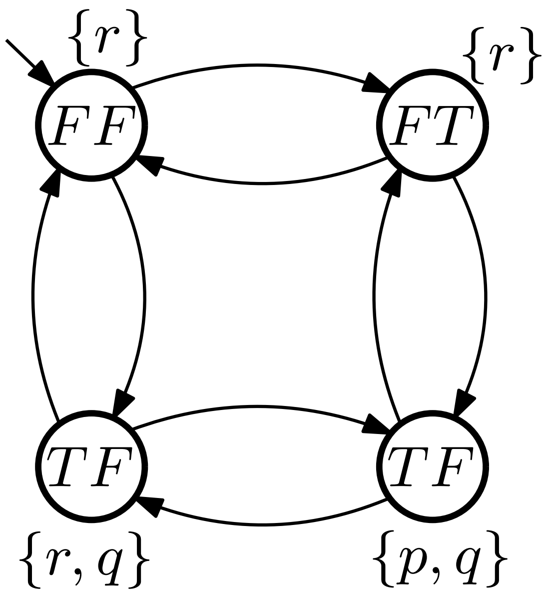
Represent $\mathcal{M}$ using Boolean logic.



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

Symbolic Model Representation

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

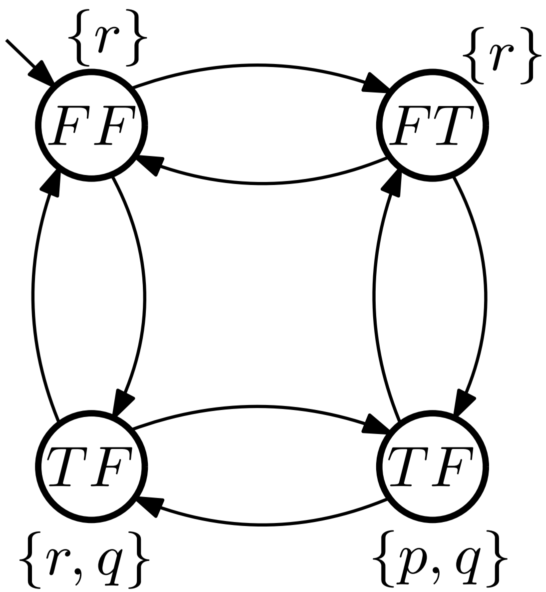


Initial State: $\neg x \wedge \neg y$

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.



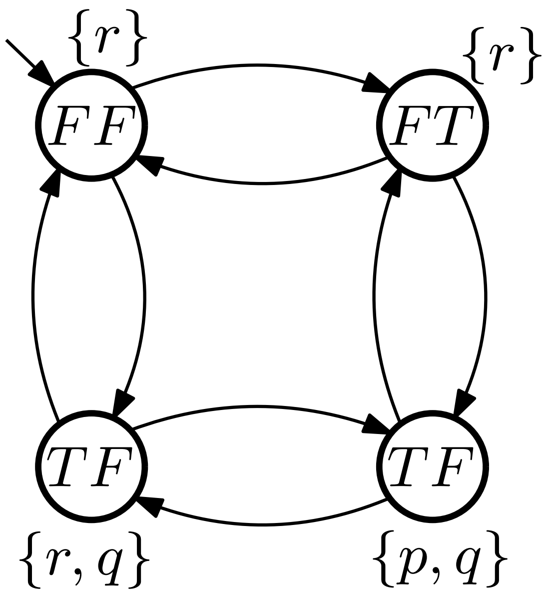
Initial State: $\neg x \wedge \neg y$

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

States	binary x	binary y	truth values x	truth values y	Boolean formula
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: $\neg x \wedge \neg y$

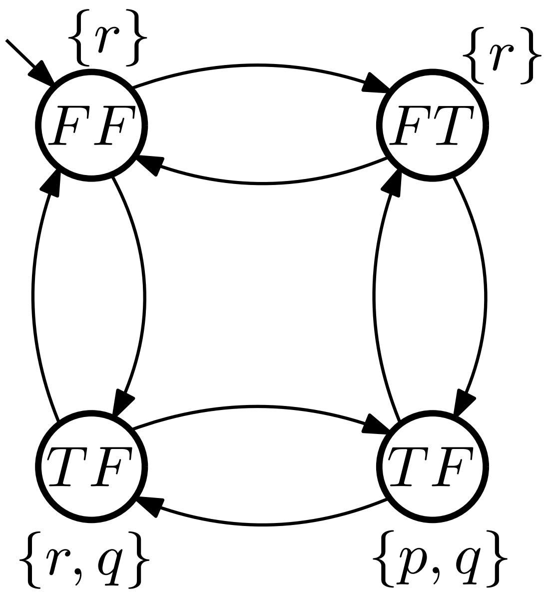
Atomic Propositions: $AP = \{p, q, r\}$

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: $\neg x \wedge \neg y$

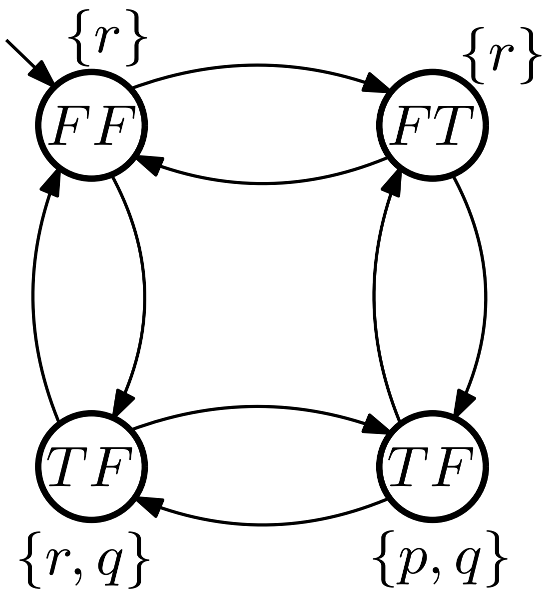
Atomic Propositions: $AP = \{p, q, r\}$

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

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

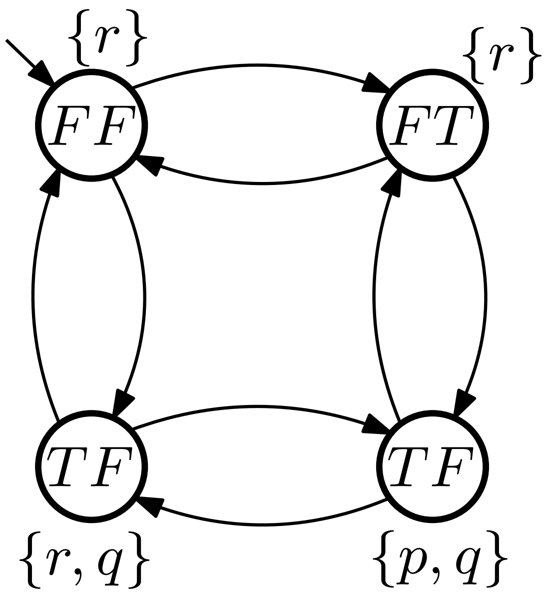
Labelling Function $\cancel{\mathcal{L} : S \to \mathcal{P}(AP)}$

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

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

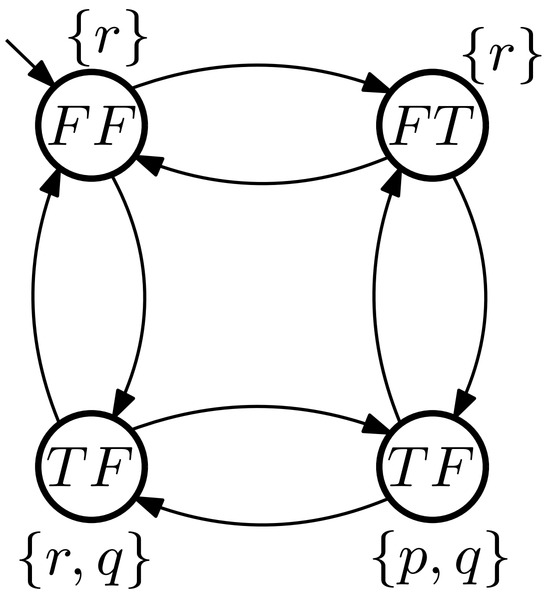
Labelling Function $\cancel{\mathcal{L} : S \to \mathcal{P}(AP)}$

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

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	$x \wedge y$

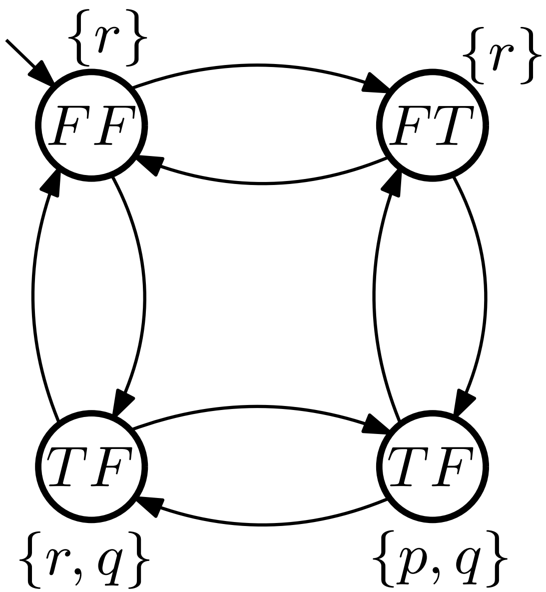
Symbolic Model Representation

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



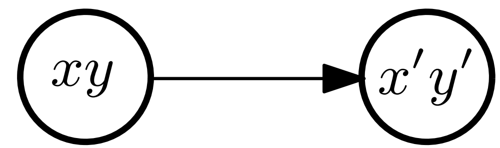
Symbolic Model Representation

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



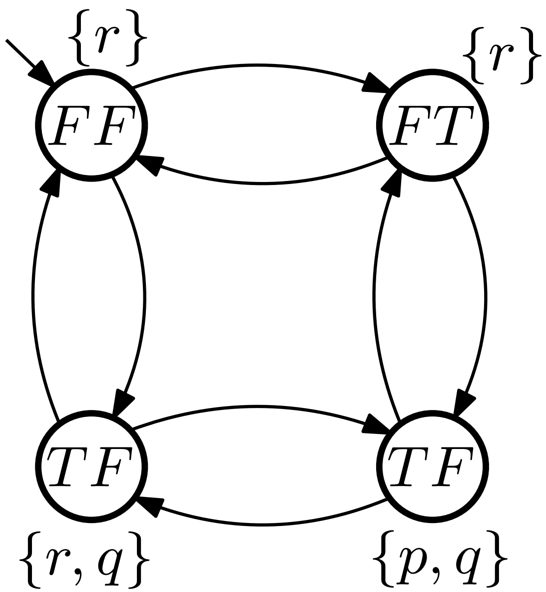
Transitions:

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



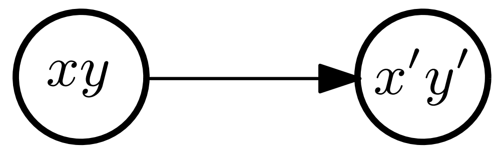
Symbolic Model Representation

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



Transitions:

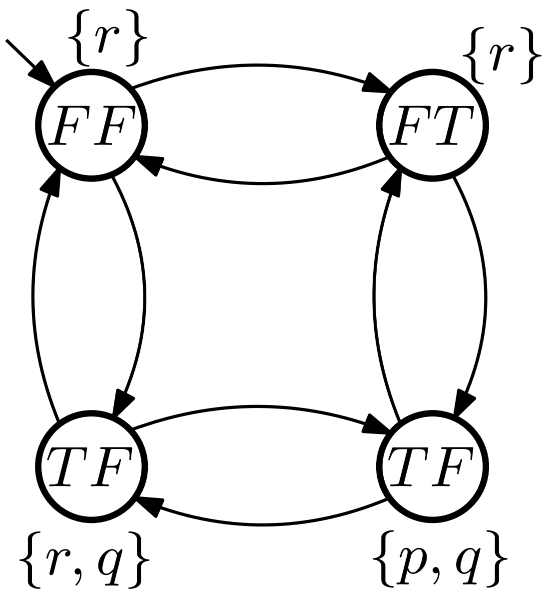
Let the "next" state variables be $V' = \{x', y'\}$



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

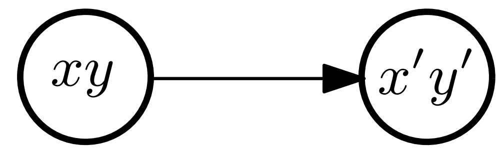
Symbolic Model Representation

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



Transitions:

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

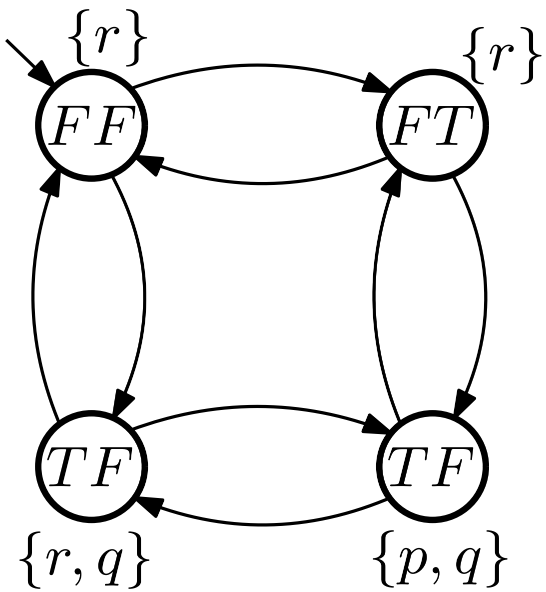


$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

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

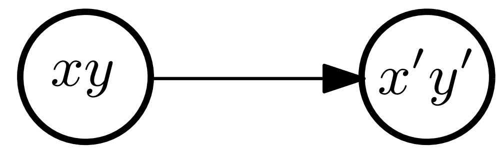
Symbolic Model Representation

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



Transitions:

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

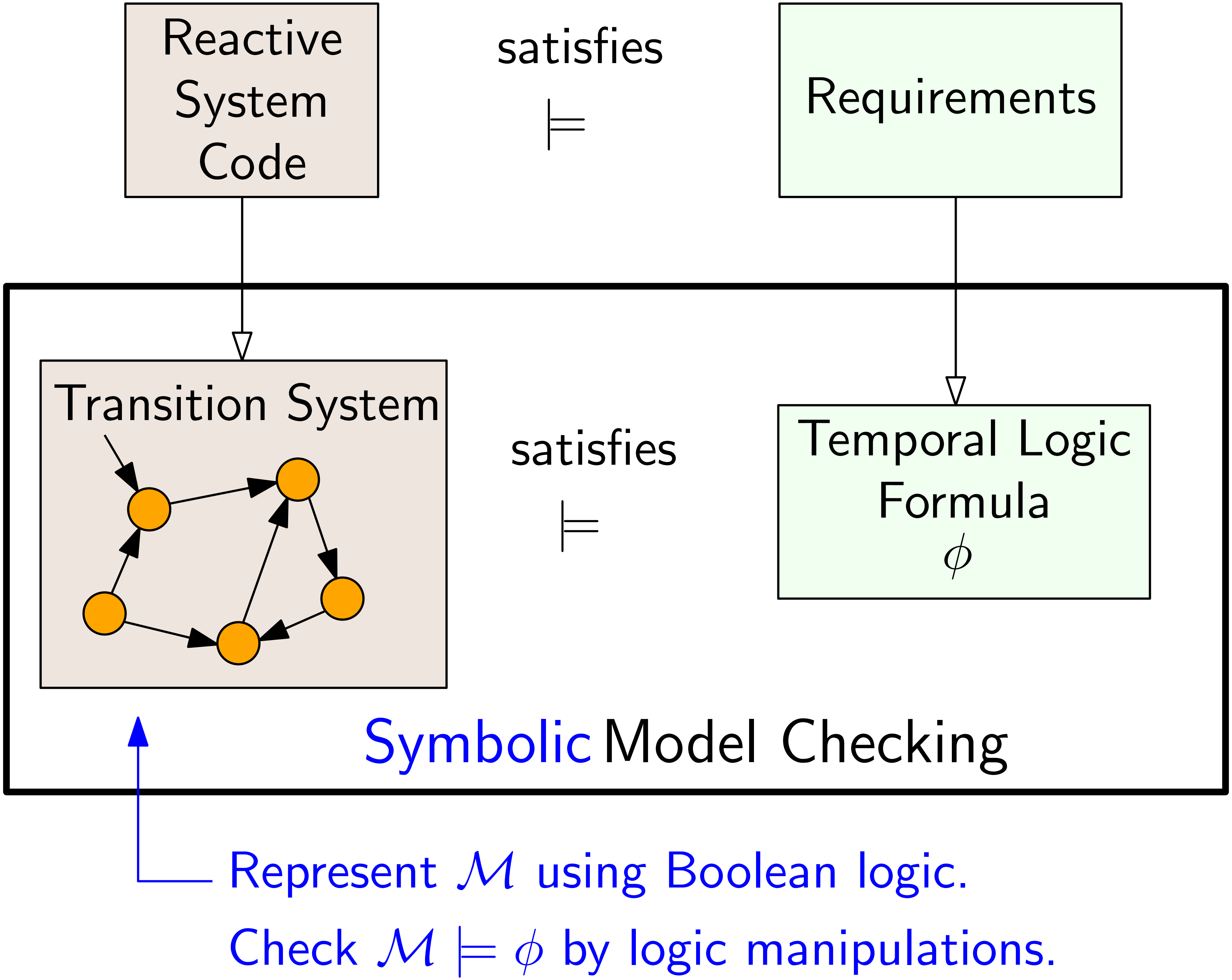


$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Explicit transitions

$$\begin{array}{llcll} (0,1) & (2,3) & \quad & (1,3) & (0,2) \\ (1,0) & (3,2) & \quad & (3,1) & (2,0) \end{array}$$

"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
Symbolic Model Checking
Represent $\mathcal{M}$ using Boolean logic.
Check $\mathcal{M} \models \phi$ by logic manipulations.

Symbolic Model Checking

How to check if $\mathcal{M} \models \phi_{CTL}$?

Symbolic Model Checking

How to check if $\mathcal{M} \models \phi_{CTL}$?

Convert ϕ_{CTL} in existential negation normal form.

ENNF uses only $EG, EU, EX, \bot, \top, \neg, \wedge, \vee, AP$.

Symbolic Model Checking

How to check if $\mathcal{M} \models \phi_{CTL}$?

Convert ϕ_{CTL} in existential negation normal form.

ENNF uses only $EG, EU, EX, \bot, \top, \neg, \wedge, \vee, AP$.

$S_\phi = \mathsf{SAT}(\phi) =$ (Set of states of $\mathcal{M}$ that satisfy ϕ.)

case

ϕ is $\top$: return S

ϕ is p_i : return $\{s : p \in L(S)\}$

ϕ is $\phi \wedge \psi$: return $\mathsf{SAT}(\phi) \cap \mathsf{SAT}(\psi)$

ϕ is $\phi \wedge \psi$: return $\mathsf{SAT}(\phi) \cup \mathsf{SAT}(\psi)$

ϕ is $\neg\phi$: return S – $\mathsf{SAT}(\phi)$

ϕ is $EX\phi$: return $\mathsf{SAT}_{EX}(\phi)$

ϕ is $\phi EU \psi$: return $\mathsf{SAT}_{EU}(\phi)$

ϕ is $EG\phi$: return $\mathsf{SAT}_{EG}(\phi)$

esac

Symbolic Model Checking

How to check if $\mathcal{M} \models \phi_{CTL}$?

Convert ϕ_{CTL} in existential negation normal form.

ENNF uses only $EG, EU, EX, \bot, \top, \neg, \wedge, \vee, AP$.

$S_\phi = \mathsf{SAT}(\phi) =$ (Set of states of $\mathcal{M}$ that satisfy ϕ.)

case

ϕ is $\top$: return S

ϕ is p_i : return $\{s : p \in L(S)\}$

ϕ is $\phi \wedge \psi$: return SAT(ϕ)$\cap$ SAT(ψ)

ϕ is $\phi \wedge \psi$: return SAT(ϕ)$\cup$ SAT(ψ)

ϕ is $\neg\phi$: return S – SAT(ϕ)

ϕ is $EX\phi$: return $\mathsf{SAT}_{EX}(\phi)$

ϕ is $\phi EU \psi$: return $\mathsf{SAT}_{EU}(\phi)$

ϕ is $EG\phi$: return $\mathsf{SAT}_{EG}(\phi)$

esac

We gave special algorithms for each of these operators

Symbolic Model Checking

How to check if $\mathcal{M} \models \phi_{CTL}$?

Convert ϕ_{CTL} in existential negation normal form.

ENNF uses only $EG, EU, EX, \bot, \top, \neg, \wedge, \vee, AP$.

$S_\phi = \text{SAT}(\phi) =$ (Set of states of $\mathcal{M}$ that satisfy ϕ.)

case

- ϕ is $\top$: return S
- ϕ is p_i : return $\{s : p \in L(S)\}$
- ϕ is $\phi \wedge \psi$: return $\text{SAT}(\phi) \cap \text{SAT}(\psi)$
- ϕ is $\phi \wedge \psi$: return $\text{SAT}(\phi) \cup \text{SAT}(\psi)$
- ϕ is $\neg\phi$: return $S - \text{SAT}(\phi)$
- ϕ is $EX\phi$: return $\text{SAT}_{EX}(\phi)$
- ϕ is $\phi EU \psi$: return $\text{SAT}_{EU}(\phi)$
- ϕ is $EG\phi$: return $\text{SAT}_{EG}(\phi)$

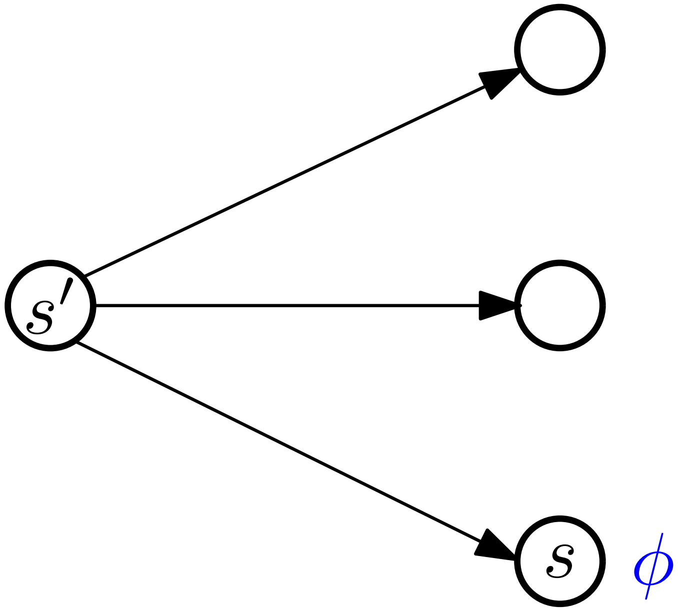
esac

We gave special algorithms for each of these operators

Check if $I \subseteq S$. If so, $\mathcal{M} \models \phi_{CTL}$.

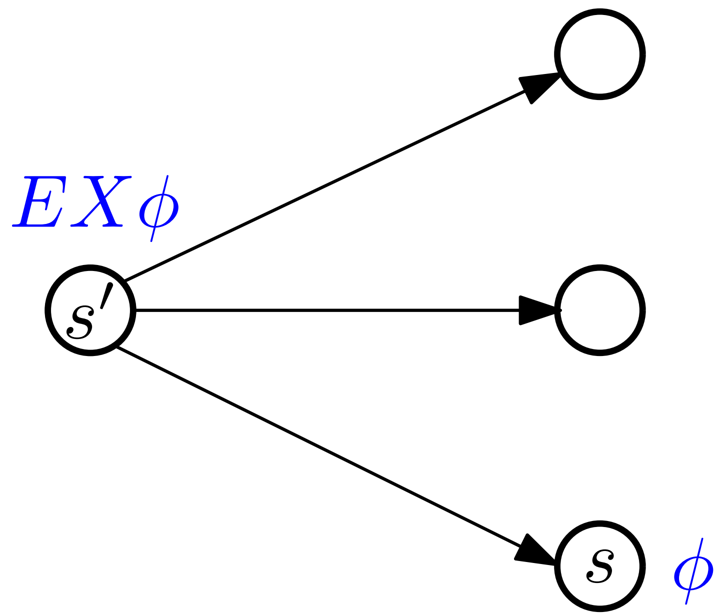
The Algorithm for $EX\ \phi$

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



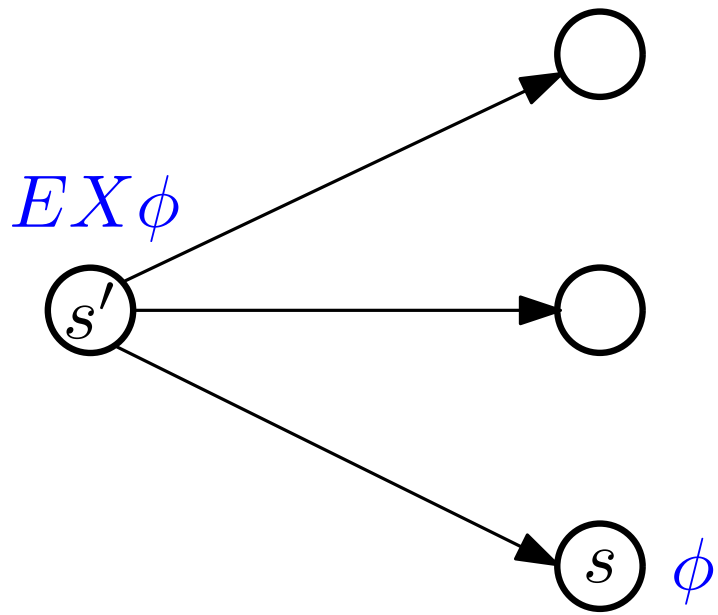
The Algorithm for $EX\ \phi$

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



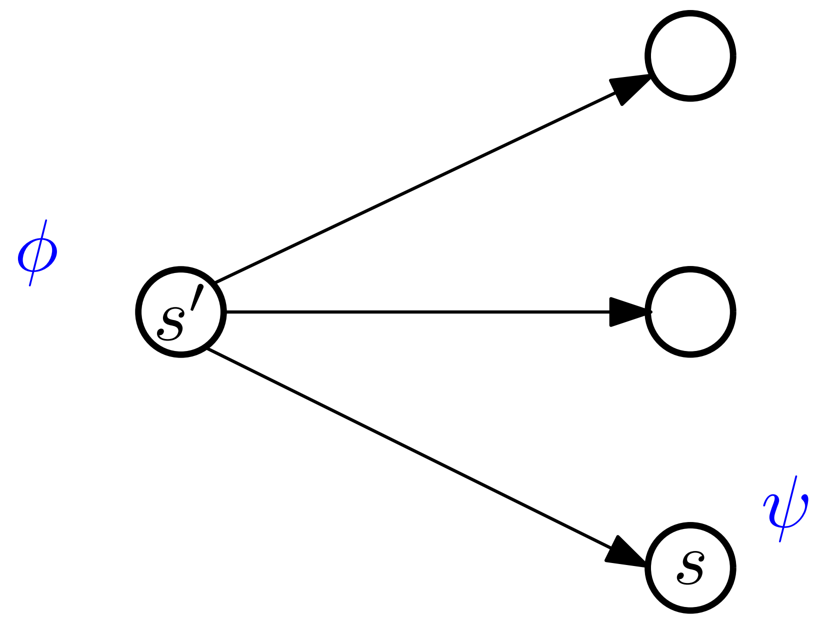
The Algorithm for $EX\ \phi$

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

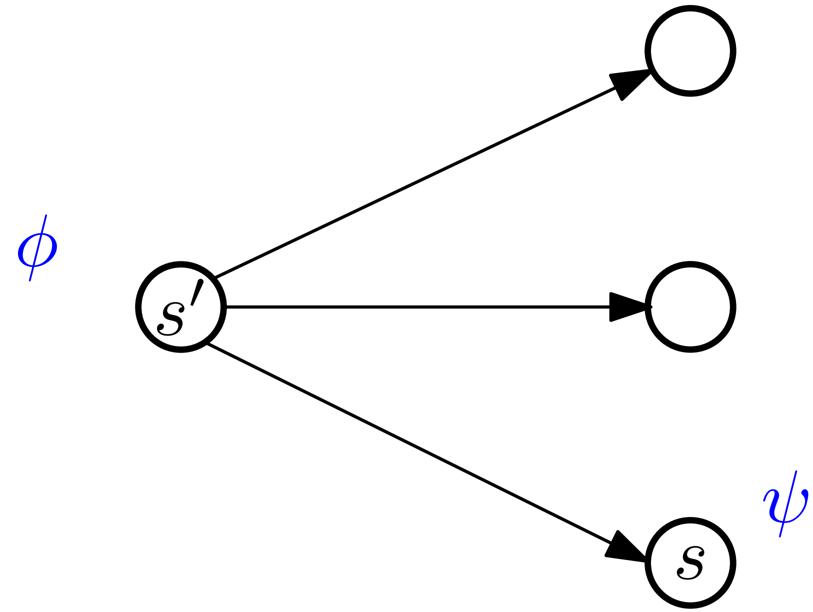


Call this process $\mathsf{SAT}_{EX}(\phi)$

The Algorithm for $\phi\ EU\ \psi$

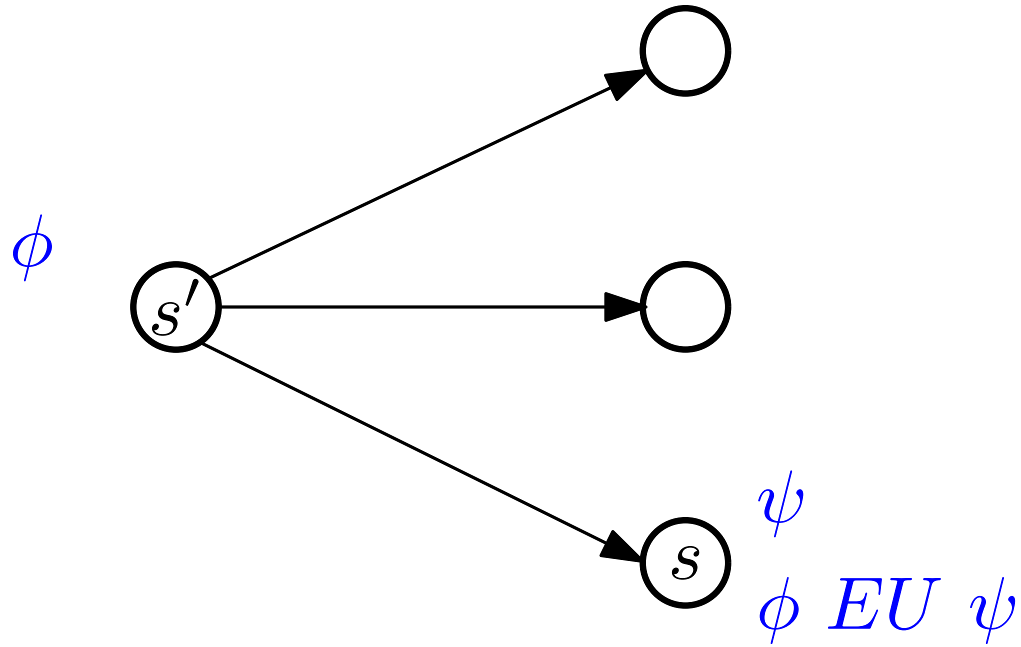


The Algorithm for $\phi \ EU \ \psi$



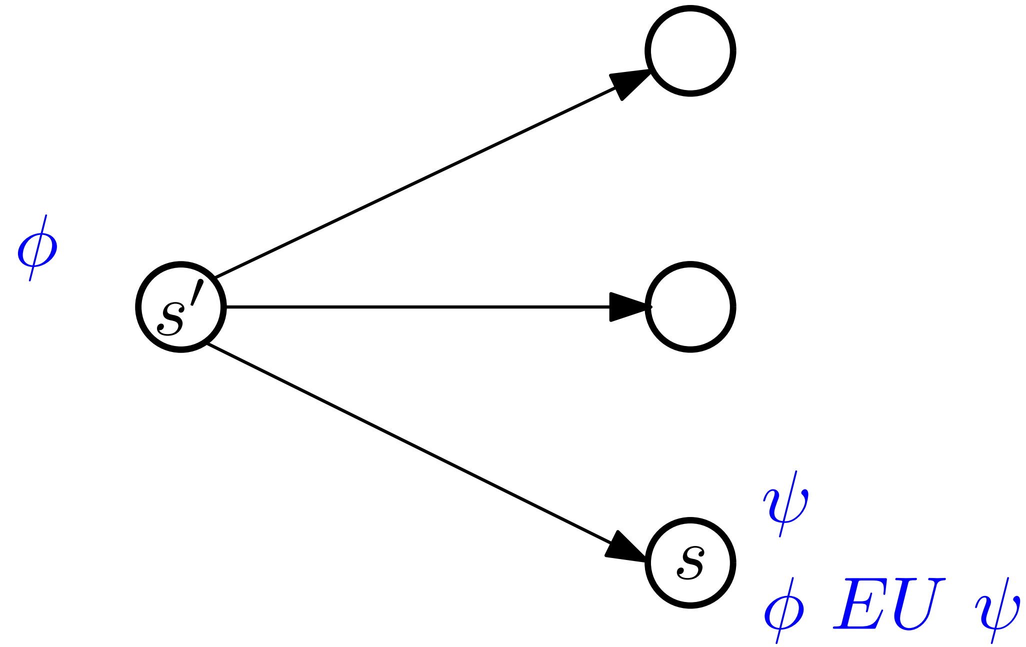
If a state is labelled with ψ label it with $\phi \ EU \ \psi$.

The Algorithm for $\phi\ EU\ \psi$



If a state is labelled with ψ label it with $\phi\ EU\ \psi$.

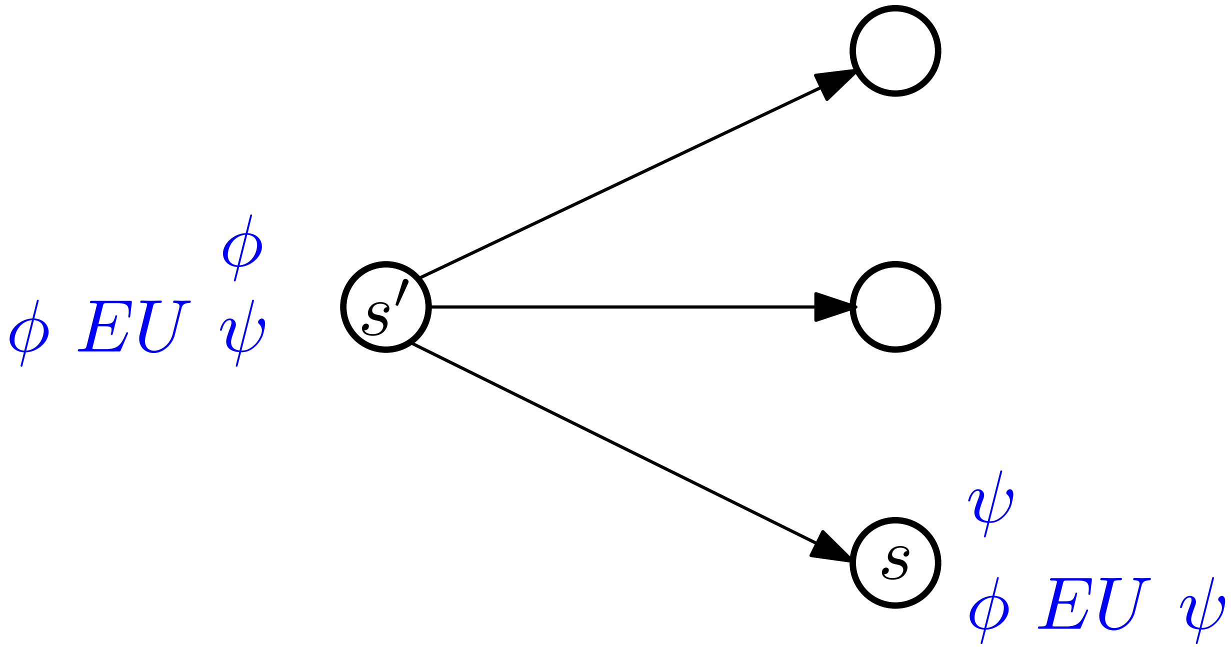
The Algorithm for $\phi\ EU\ \psi$



If a state is labelled with ψ label it with $\phi\ EU\ \psi$.

For any state s' labelled with ϕ, if at least one successor state s is labelled with $\phi\ EU\ \psi$, then label s' with $\phi\ EU\ \psi$ as well. Repeat until labels stop changing.

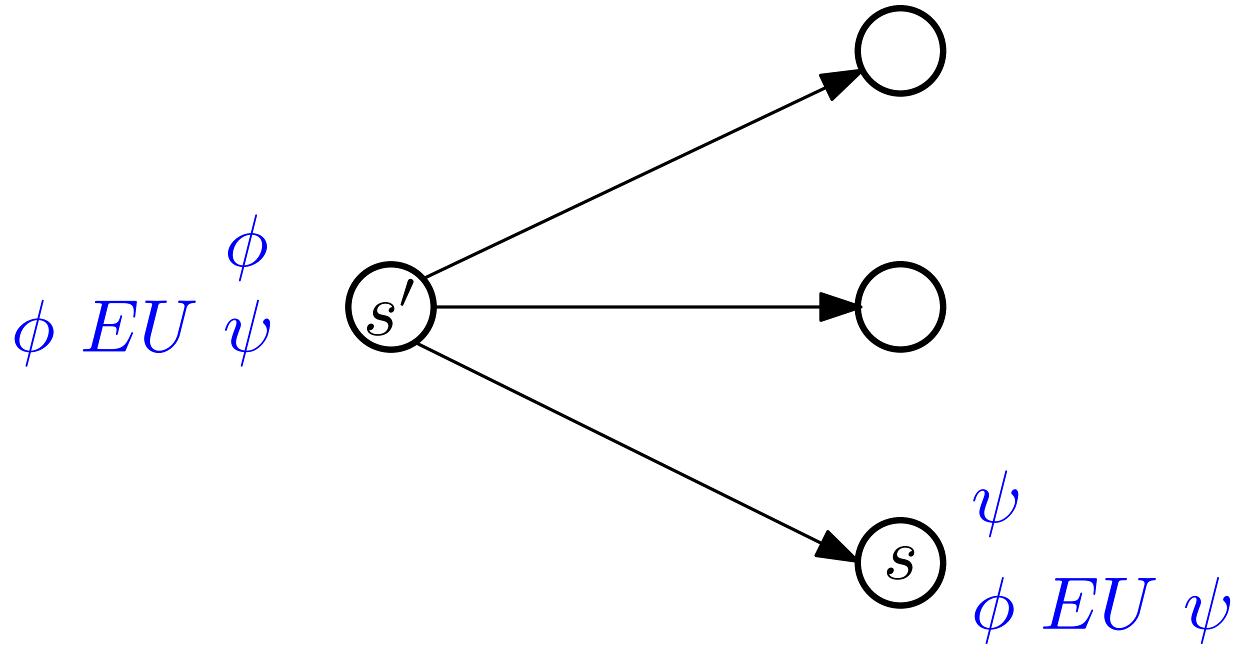
The Algorithm for $\phi\ EU\ \psi$



If a state is labelled with ψ label it with $\phi\ EU\ \psi$.

For any state s' labelled with ϕ, if at least one successor state s is labelled with $\phi\ EU\ \psi$, then label s' with $\phi\ EU\ \psi$ as well. Repeat until labels stop changing.

The Algorithm for $\phi\ EU\ \psi$

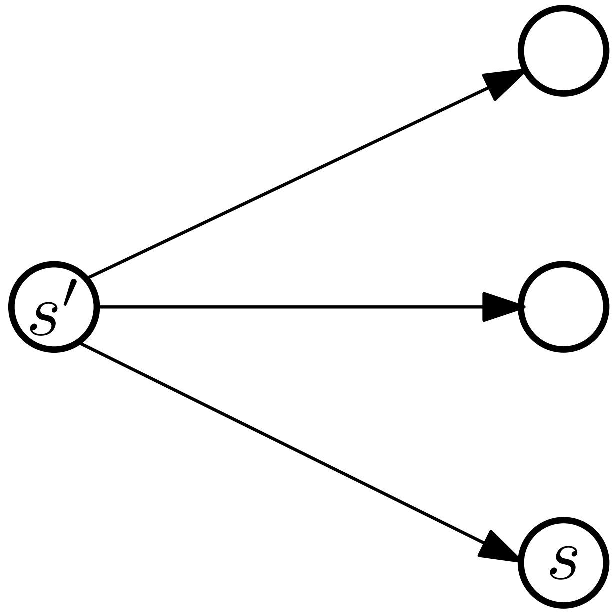


If a state is labelled with ψ label it with $\phi\ EU\ \psi$.

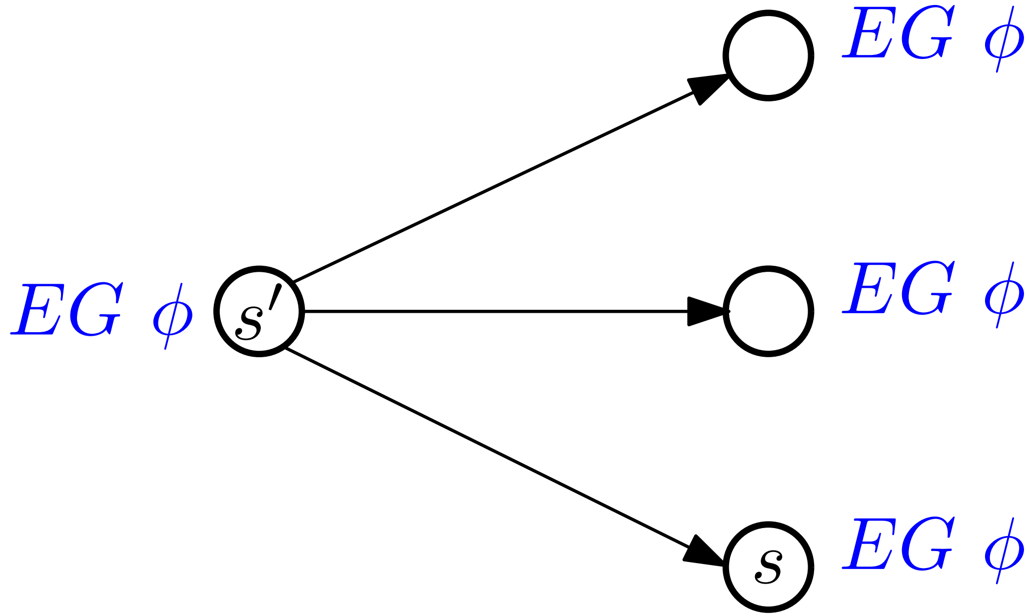
For any state s' labelled with ϕ, if at least one successor state s is labelled with $\phi\ EU\ \psi$, then label s' with $\phi\ EU\ \psi$ as well. Repeat until labels stop changing.

Call this process $\mathsf{SAT}_{EU}(\phi, \psi)$

The Algorithm for $EG\ \phi$

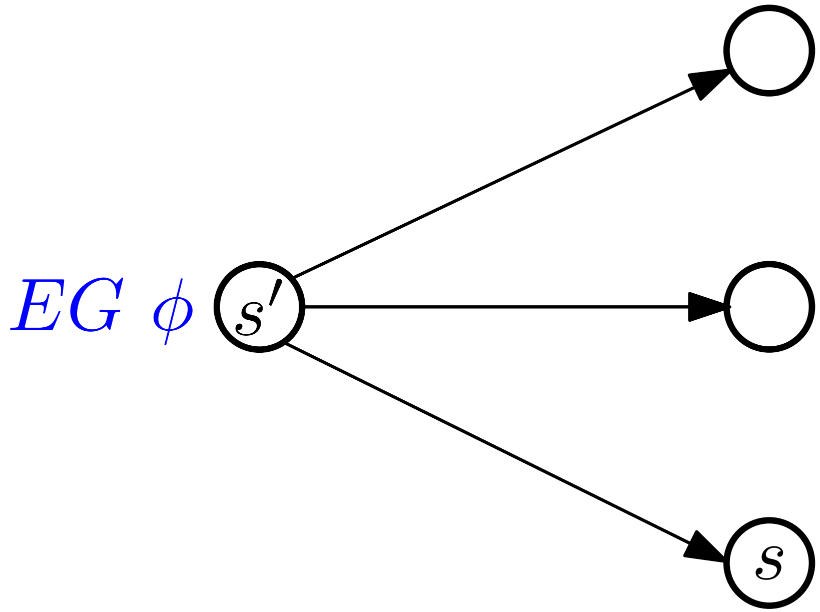


The Algorithm for $EG\ \phi$



Label all states with $EG\ \phi$

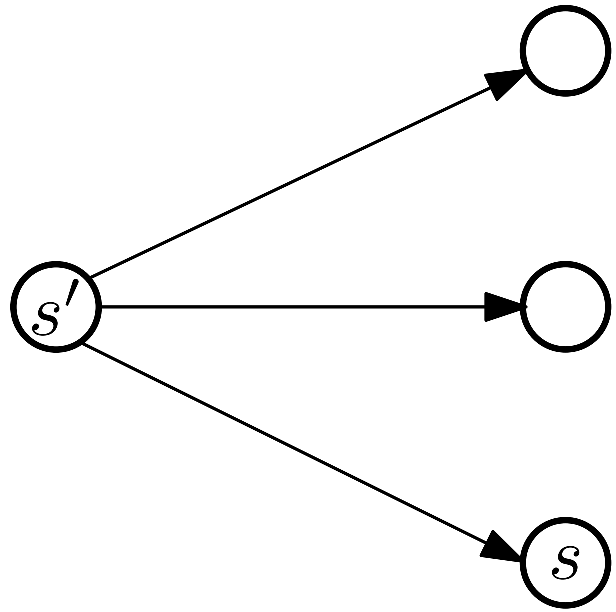
The Algorithm for $EG\ \phi$



Label all states with $EG\ \phi$

Delete $EG\ \phi$ from any state not labelled with ϕ.

The Algorithm for $EG\ \phi$

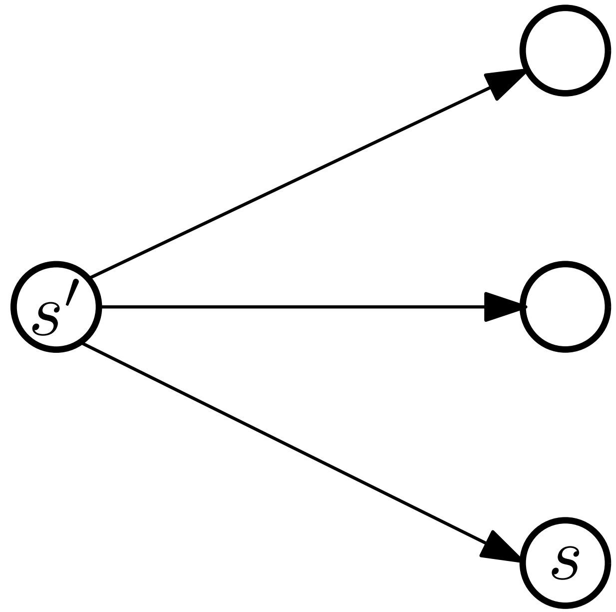


Label all states with $EG\ \phi$

Delete $EG\ \phi$ from any state not labelled with ϕ.

Delete $EG\ \phi$ from any state where none of its successors is labelled with $EG\ \phi$. Repeat until no more labels can be deleted.

The Algorithm for $EG\ \phi$



Label all states with $EG\ \phi$

Delete $EG\ \phi$ from any state not labelled with ϕ.

Delete $EG\ \phi$ from any state where none of its successors is labelled with $EG\ \phi$. Repeat until no more labels can be deleted.

Call this process $\text{SAT}_{EG}(\phi)$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$EX\ \phi \equiv EX\ \phi$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$EX\ \phi \equiv EX\ \phi$$

$$EG\ \phi \equiv \phi \wedge\ EX\ EG\ \phi$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$EX\ \phi \equiv EX\ \phi$$

$$EG\ \phi \equiv \phi \wedge\ EX\ EG\ \phi$$

$$EF\ \phi \equiv \phi \vee\ EX\ EF\ \phi$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$
\begin{aligned}
EX\ \phi &\equiv EX\ \phi \\
EG\ \phi &\equiv \phi \wedge\ EX\ EG\ \phi \\
EF\ \phi &\equiv \phi \vee\ EX\ EF\ \phi \\
\phi\ EU\ \psi &\equiv \psi \vee \psi \wedge\ EX\ (\phi\ EU\ \psi)
\end{aligned}
$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$EX\ \phi \equiv \boxed{EX}\ \phi$$

$$EG\ \phi \equiv \phi \wedge \boxed{EX}\ EG\ \phi$$

$$EF\ \phi \equiv \phi \vee \boxed{EX}\ EF\ \phi$$

$$\phi\ EU\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\ (\phi\ EU\ \psi)$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$\boxed{EX}\ \phi \equiv \boxed{EX}\ \phi$$

$$EG\ \phi \equiv \phi \wedge \boxed{EX}\ EG\ \phi$$

$$EF\ \phi \equiv \phi \vee \boxed{EX}\ EF\ \phi$$

$$\phi\ EU\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\ (\phi\ EU\ \psi)$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$\boxed{EX}\ \phi \equiv \boxed{EX}\ \phi$$

$$\boxed{EG}\ \phi \equiv \phi \wedge \boxed{EX}\,\boxed{EG}\ \phi$$

$$EF\ \phi \equiv \phi \vee \boxed{EX}\ EF\ \phi$$

$$\phi\ EU\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\,(\phi\ EU\ \psi)$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$\boxed{EX}\ \phi \equiv \boxed{EX}\ \phi$$

$$\boxed{EG}\ \phi \equiv \phi \wedge \boxed{EX}\,\boxed{EG}\ \phi$$

$$\boxed{EF}\ \phi \equiv \phi \vee \boxed{EX}\,\boxed{EF}\ \phi$$

$$\phi\ EU\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\,(\phi\ EU\ \psi)$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$\boxed{EX}\ \phi \equiv \boxed{EX}\ \phi$$

$$\boxed{EG}\ \phi \equiv \phi \wedge \boxed{EX}\boxed{EG}\ \phi$$

$$\boxed{EF}\ \phi \equiv \phi \vee \boxed{EX}\boxed{EF}\ \phi$$

$$\phi\ \boxed{EU}\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\ (\phi\ \boxed{EU}\ \psi)$$

Symbolic Model Checking

We need to handle EG, EU, EX symbolically, i.e. by manipulating Boolean formulas.

Recall from Homework:

We can write CTL operators "recursively" using EX.

$$\boxed{EX}\ \phi \equiv \boxed{EX}\ \phi$$

$$\boxed{EG}\ \phi \equiv \phi \wedge \boxed{EX}\boxed{EG}\ \phi$$

$$\boxed{EF}\ \phi \equiv \phi \vee \boxed{EX}\boxed{EF}\ \phi$$

$$\phi\ \boxed{EU}\ \psi \equiv \psi \vee \psi \wedge \boxed{EX}\ (\phi\ \boxed{EU}\ \psi)$$

Main Idea: if we can describe how to do symbolic model checking for $EX\phi$, then we can give recursive algorithms for the other operators.

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.

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

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.

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

exists a path where ϕ holds in the next state

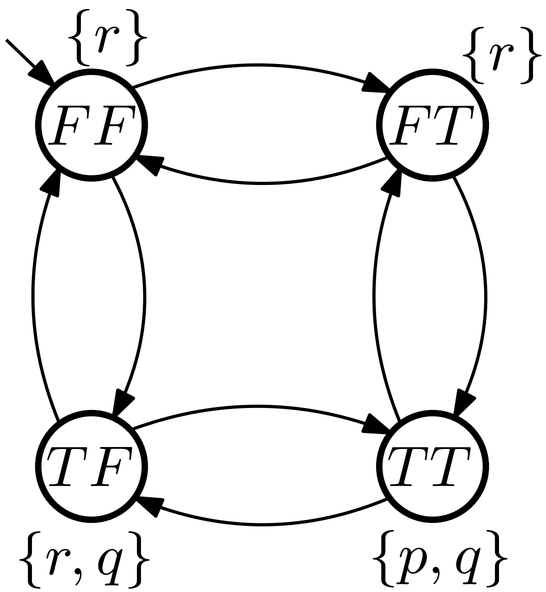
$\equiv$

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

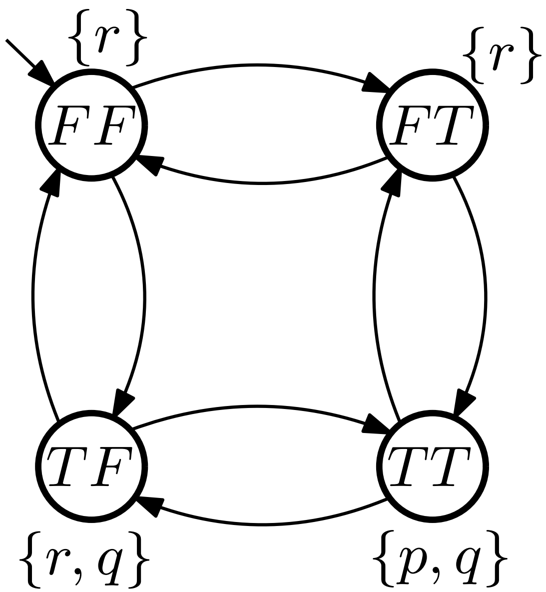
Labelling Function $\quad \mathcal{L} : AP \rightarrow \mathcal{F}(x, y)$

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

Transition Relation:

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

Symbolic Model Checking



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

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

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

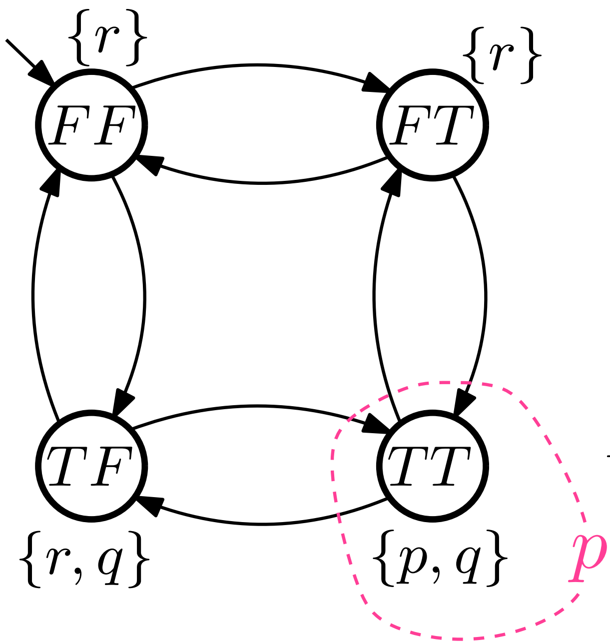
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

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

Let's compute $EX\ p$

Symbolic Model Checking



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

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

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

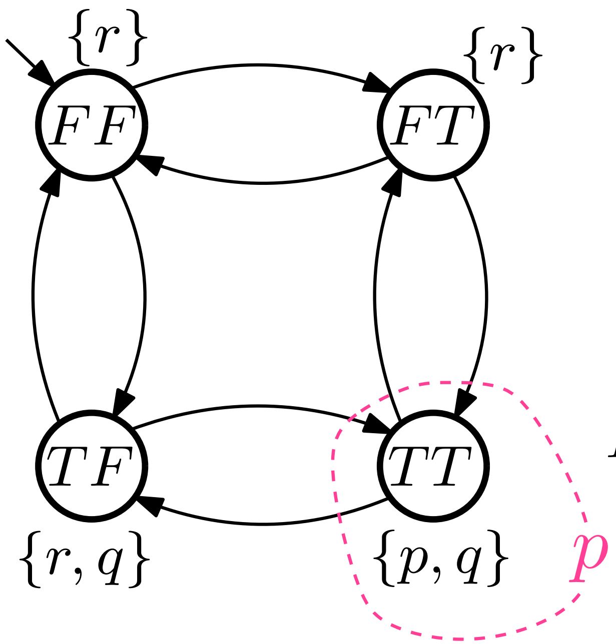
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

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

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

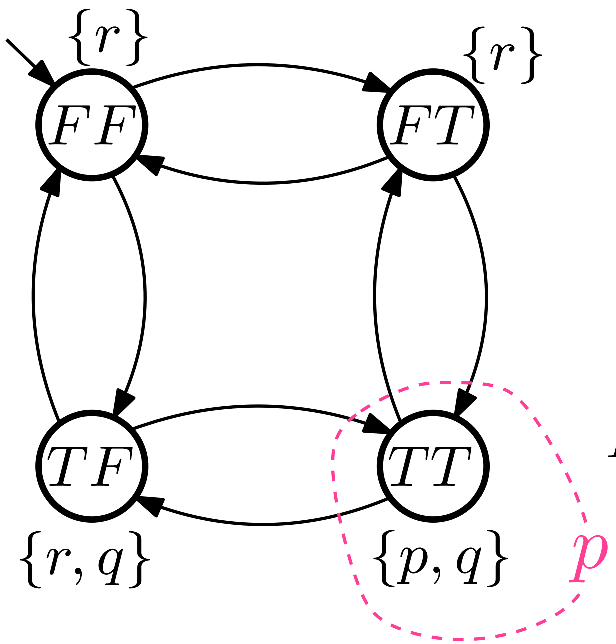
Transition Relation:

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

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:

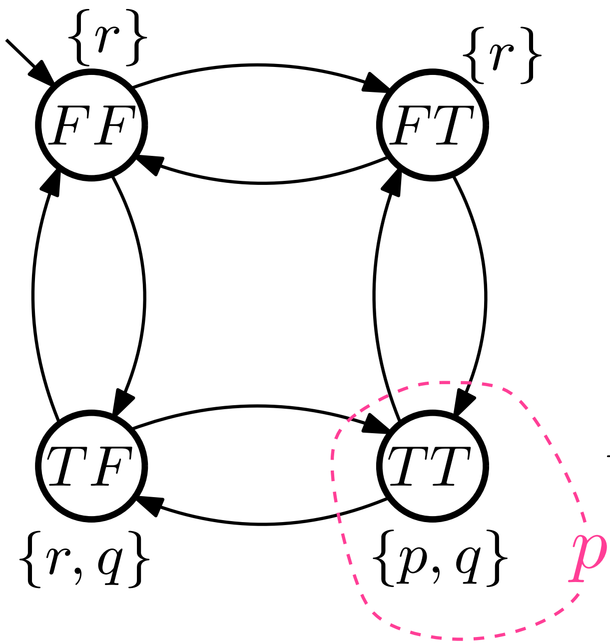
$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

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

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

Transition Relation:

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

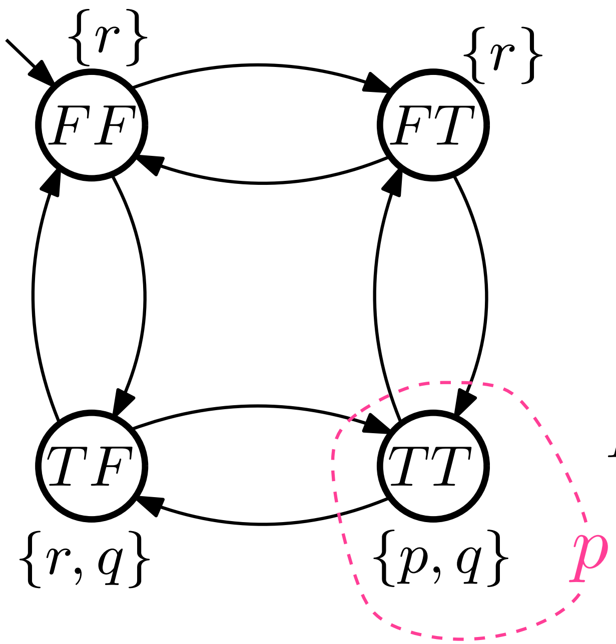
Let's compute $EX\ p$

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

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

. . . some Boolean simplifications . . .

Symbolic Model Checking



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

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

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

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

Transition Relation:

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

Let's compute $EX\ p$

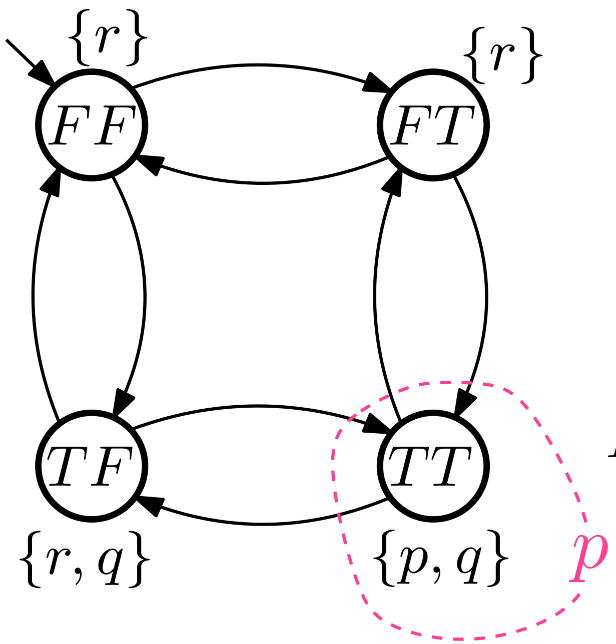
$$EX\ p \equiv \exists V' \ \ R \ \wedge \ p[\ V'\ /V\ ]$$

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

. . . 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 $EX\ p$

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

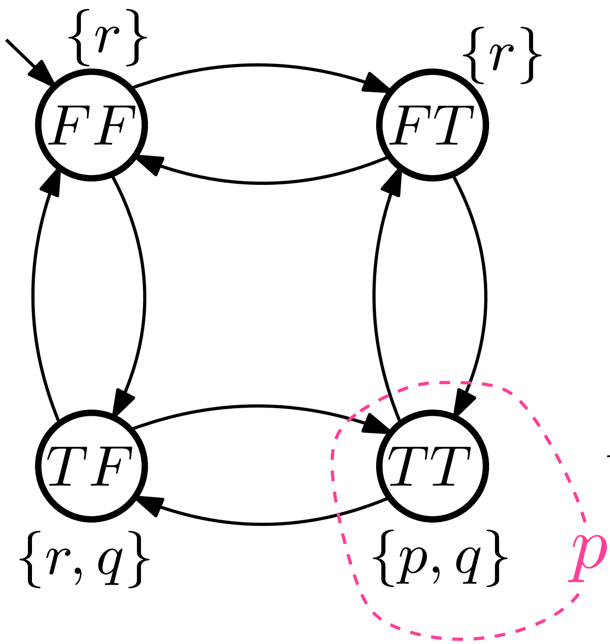
$$EX\ p \equiv \exists x', y'\ \ (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)\ \wedge\ (x' \wedge y')$$

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

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

Transition Relation:

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

Let's compute $EX\ p$

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

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

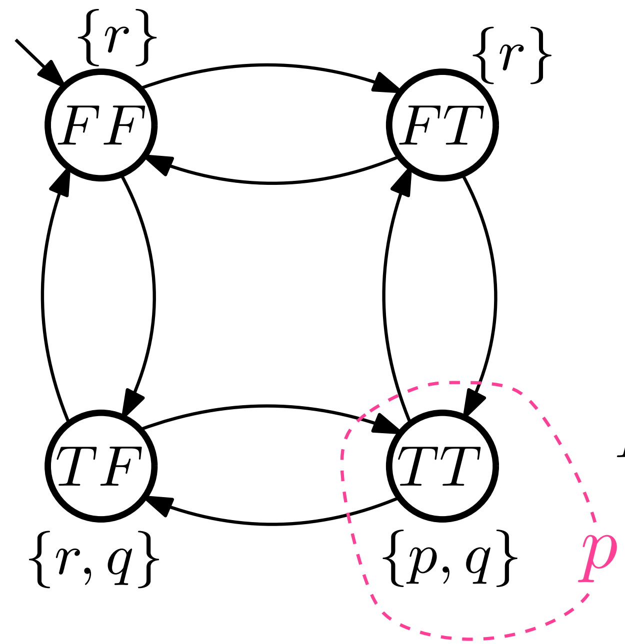
...some Boolean simplifications ...

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

...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 . . .

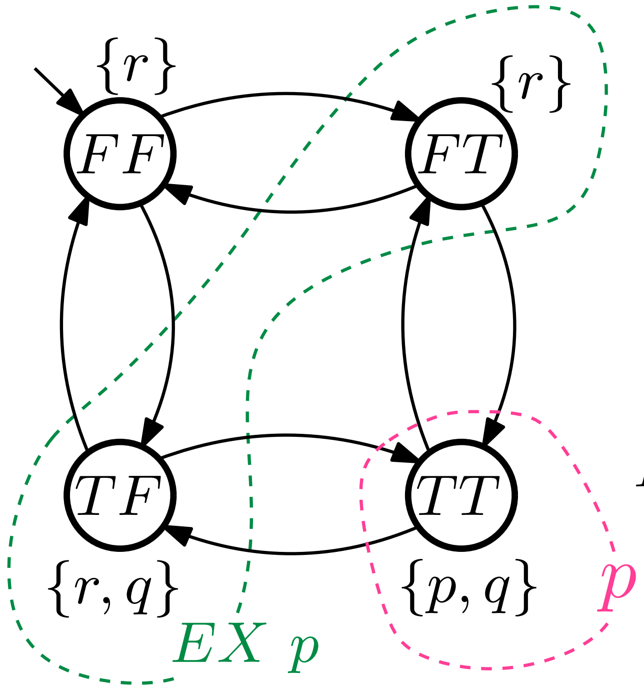
$$EX\ p \equiv \exists x', y' \;\; (x' \wedge x \wedge y' \wedge \neg y) \vee (x' \wedge \neg x \wedge y' \wedge y)$$

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

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

Transition Relation:

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

Let's compute $EX\ p$

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

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

. . . 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

Let's think about EF

Symbolic Model Checking

Let's think about EF

$$EF\ \phi \ \equiv\ \phi \vee\ EX\ EF\ \phi$$

Symbolic Model Checking

Let's think about EF

$$EF\ \phi \ \equiv \ \phi \vee \ EX\ EF\ \phi$$

$$EF\ \phi \ \equiv \ \phi \vee \ EX\ (\phi \vee \ EX\ EF\ \phi)$$

Symbolic Model Checking

Let's think about EF

$$EF\ \phi \ \equiv \ \phi \vee \ EX\ EF\ \phi$$

$$EF\ \phi \ \equiv \ \phi \vee \ EX\ (\phi \vee \ EX\ EF\ \phi)$$

$$EF\ \phi \ \equiv \ \phi \vee \ EX\ \phi \vee \ EX\ EX\ EF\ \phi$$

Symbolic Model Checking

Let's think about EF

$$EF\ \phi \;\equiv\; \phi \vee\ EX\ EF\ \phi$$

$$EF\ \phi \;\equiv\; \phi \vee\ EX\ (\phi \vee\ EX\ EF\ \phi)$$

$$EF\ \phi \;\equiv\; \phi \vee\ EX\ \phi \vee\ EX\ EX\ EF\ \phi$$

$$EF\ \phi \;\equiv\; \phi \vee\ EX\ \phi \vee\ EX\ EX\ \phi\ \vee\ EX\ EX\ EX\ EF\ \phi$$

Symbolic Model Checking

Let's think about EF

$$EF\ \phi\ \equiv\ \phi \vee\ EX\ EF\ \phi$$

$$EF\ \phi\ \equiv\ \phi \vee\ EX\ (\phi \vee\ EX\ EF\ \phi)$$

$$EF\ \phi\ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ EF\ \phi$$

$$EF\ \phi\ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ \phi\ \vee\ EX\ EX\ EX\ EF\ \phi$$

$$EF\ \phi\ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ \phi\ \vee\ EX\ EX\ EX\ \phi \vee \ldots$$

Symbolic Model Checking

Let's think about EF

$$EF\ \phi \ \equiv\ \phi \vee\ EX\ EF\ \phi$$

$$EF\ \phi \ \equiv\ \phi \vee\ EX\ (\phi \vee\ EX\ EF\ \phi)$$

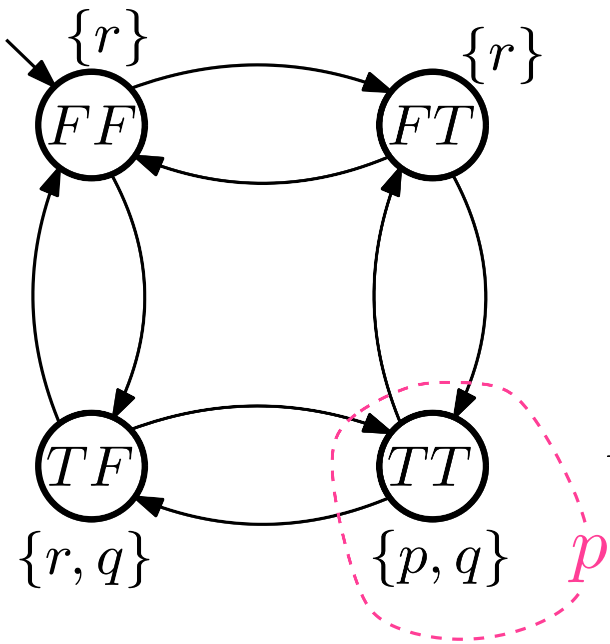
$$EF\ \phi \ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ EF\ \phi$$

$$EF\ \phi \ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ \phi\ \vee\ EX\ EX\ EX\ EF\ \phi$$

$$EF\ \phi \ \equiv\ \phi \vee\ EX\ \phi \vee\ EX\ EX\ \phi\ \vee\ EX\ EX\ EX\ \phi \vee \ldots$$

$$EF\ \phi \ \equiv\ \bigvee_{i=0}^{\infty}\ EX^i\ \phi \qquad \text{(where } EX^0\phi\ =\ \phi\text{)}$$

Symbolic Model Checking



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

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

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

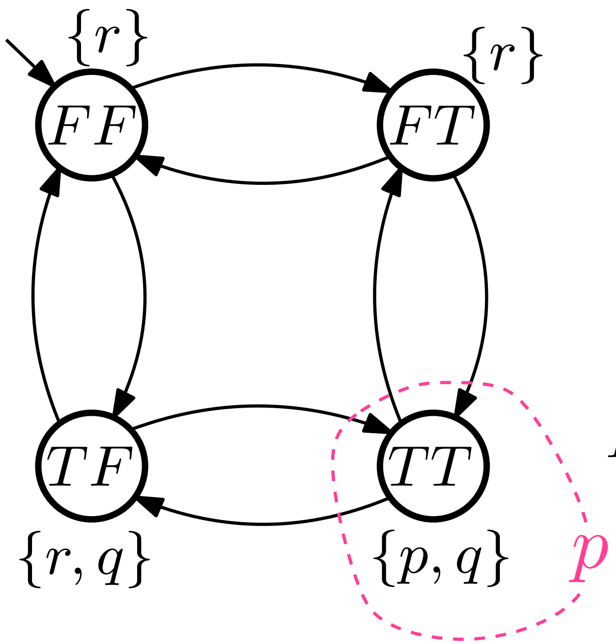
$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$

Transition Relation:

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

Let's compute $EF\ p$

Symbolic Model Checking



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

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

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

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

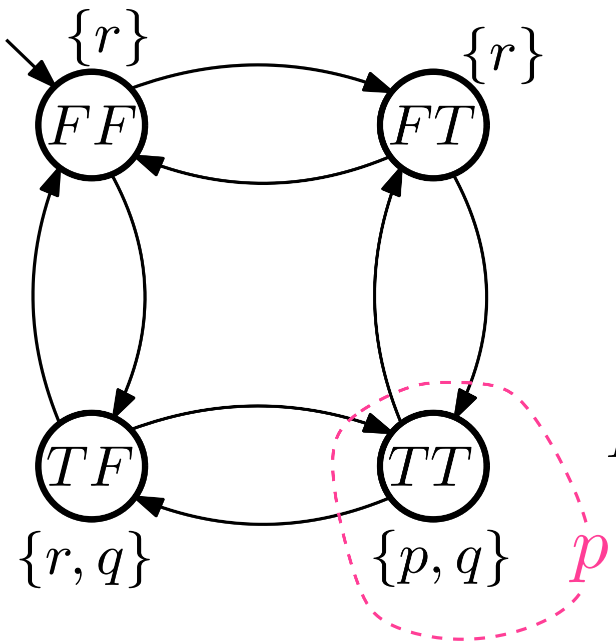
Transition Relation:

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

Let's compute $EF\ p$

$EF\ p \equiv EF\ (x \wedge y)$

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:

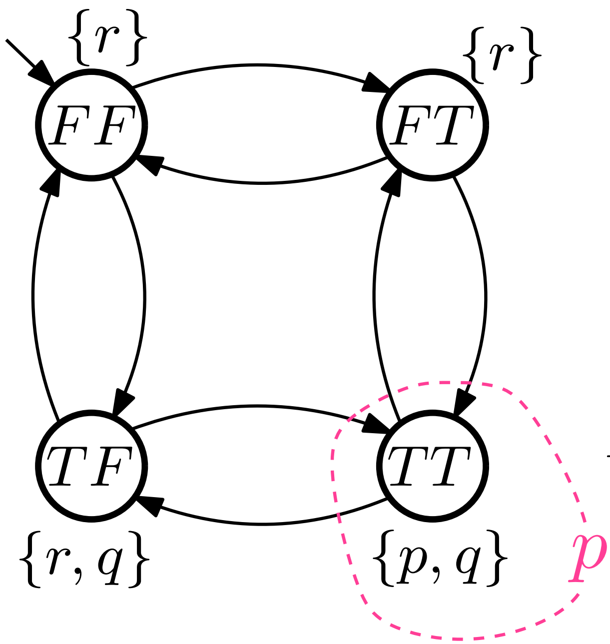
$$R \equiv (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $EF\ p$

$EF\ p \equiv\ EF\ (x \wedge y)$

$\equiv (x \wedge y) \vee\ EX\ EF\ (x \wedge y)$

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:

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

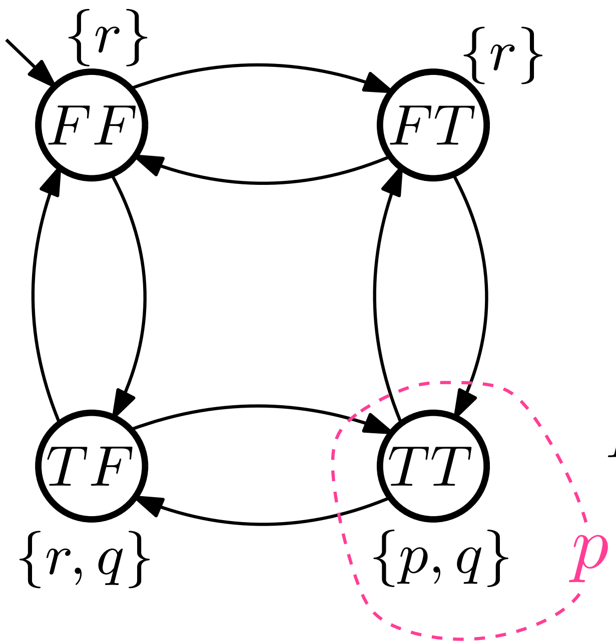
Let's compute $EF\ p$

$EF\ p \equiv\ EF\ (x \wedge y)$

$\equiv (x \wedge y) \vee\ EX\ EF\ (x \wedge y)$

$\equiv (x \wedge y) \vee\ EX(x \wedge y) \vee\ EX\ EX\ EF\ (x \wedge y)$

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 $EF\ p$

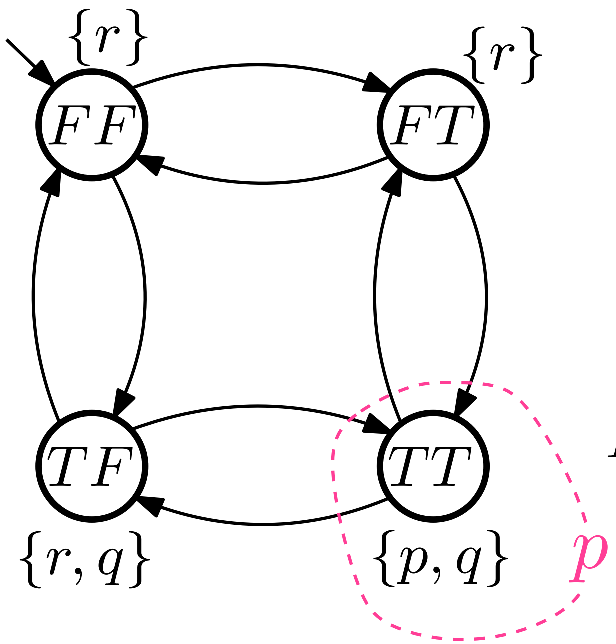
$EF\ p \equiv EF\ (x \wedge y)$

$\equiv (x \wedge y) \vee EX\ EF\ (x \wedge y)$

$\equiv (x \wedge y) \vee EX(x \wedge y) \vee EX\ EX\ EF\ (x \wedge y)$

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

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 $EF\ p$

$$EF\ p \equiv EF\ (x \wedge y)$$

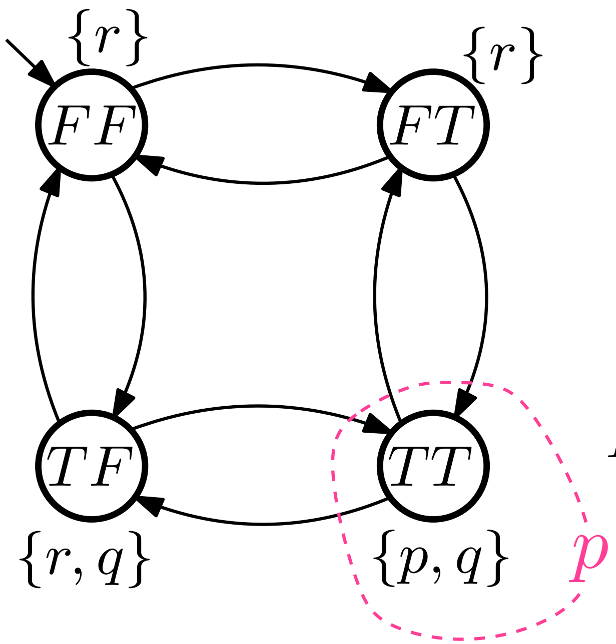
$$\equiv (x \wedge y) \vee EX\ EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee EX(x \wedge y) \vee EX\ EX\ EF\ (x \wedge y)$$

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

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

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:

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

Let's compute $EF\ p$

$$EF\ p \equiv\ EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee\ EX\ EF\ (x \wedge y)$$

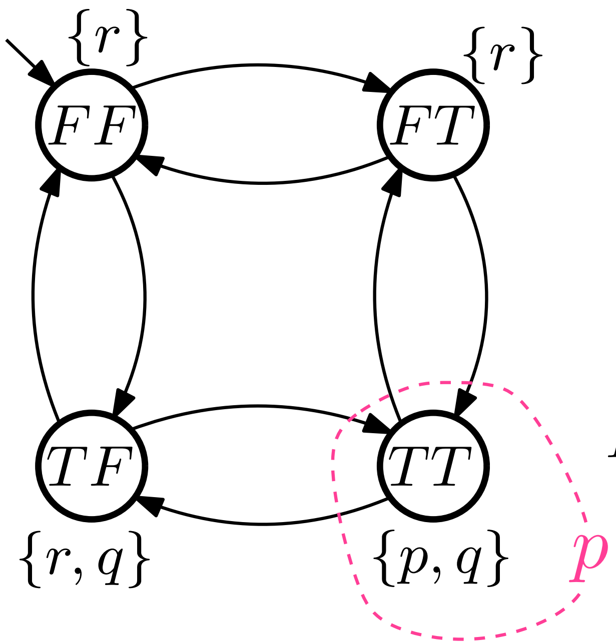
$$\equiv (x \wedge y) \vee\ EX(x \wedge y) \vee\ EX\ EX\ EF\ (x \wedge y)$$

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

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

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

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 $EF\ p$

$$EF\ p \equiv EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee EX\ EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee EX(x \wedge y) \vee EX\ EX\ EF\ (x \wedge y)$$

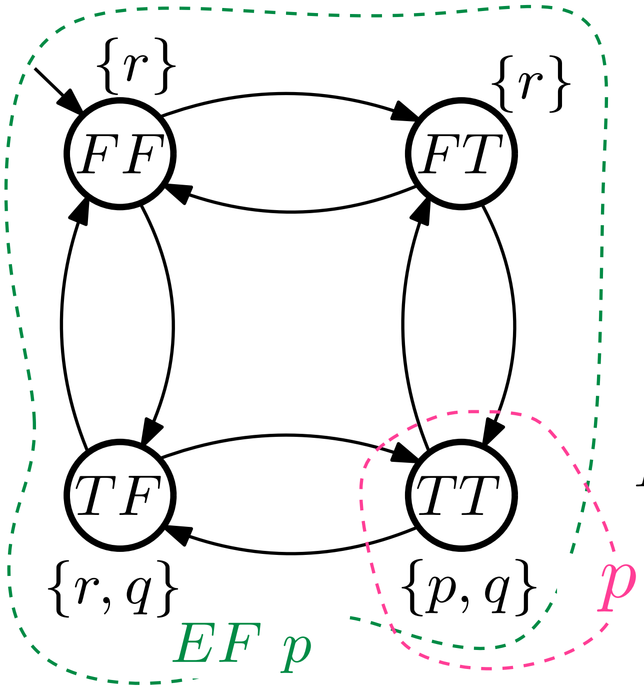
$$\equiv (x \wedge y) \vee (x \wedge \neg y) \vee (\neg x \wedge y) \vee EX\ EX\ (x \wedge y) \vee EX\ EX\ EX\ EF\ (x \wedge y)$$

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

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

$$\equiv (x \wedge y) \vee (x \wedge \neg y) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y) \vee (x \wedge y) \ldots \equiv T$$

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:

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

Let's compute $EF\ p$

$$EF\ p \equiv \ EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee \ EX\ EF\ (x \wedge y)$$

$$\equiv (x \wedge y) \vee \ EX(x \wedge y) \vee \ EX\ EX\ EF\ (x \wedge y)$$

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

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

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

$$\equiv (x \wedge y) \vee \ (x \wedge \neg y) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y) \vee (x \wedge y) \ldots \equiv T$$

Symbolic Model Checking

Let's think about EG

Symbolic Model Checking

Let's think about EG

$$EG\ \phi \ \equiv \ \phi \wedge \ EX\ EG\ \phi$$

Symbolic Model Checking

Let's think about EG

$$EG\ \phi\ \equiv\ \phi \wedge\ EX\ EG\ \phi$$

$$EG\ \phi\ \equiv\ \phi \wedge\ EX\ (\phi \wedge\ EX\ EG\ \phi)$$

Symbolic Model Checking

Let's think about EG

$$EG\ \phi \ \equiv\ \phi \wedge\ EX\ EG\ \phi$$

$$EG\ \phi \ \equiv\ \phi \wedge\ EX\ (\phi \wedge\ EX\ EG\ \phi)$$

??

$$EG\ \phi \ \equiv\ \phi \wedge\ EX\ \phi \wedge\ EX\ EX\ \phi \wedge EX\ EX\ EX\ \phi \wedge \ldots$$

Symbolic Model Checking

Let's think about EG

$$EG\ \phi \ \equiv \ \phi \wedge \ EX\ EG\ \phi$$

$$EG\ \phi \ \equiv \ \phi \wedge \ EX\ (\phi \wedge \ EX\ EG\ \phi)$$

?? Yes! Using lattices, fixed-points, μ-calculus

$$EG\ \phi \ \equiv \ \phi \wedge \ EX\ \phi \wedge \ EX\ EX\ \phi \wedge EX\ EX\ EX\ \phi \wedge \ldots$$

Huff & Ryan Logic for Computer Science Section 3.7, but you can just trust me :)

Symbolic Model Checking

Let's think about EG

$$EG\ \phi\ \equiv\ \phi \wedge\ EX\ EG\ \phi$$

$$EG\ \phi\ \equiv\ \phi \wedge\ EX\ (\phi \wedge\ EX\ EG\ \phi)$$

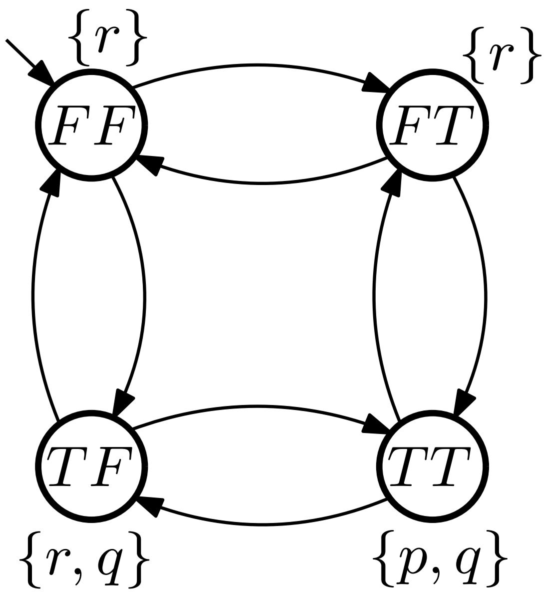
?? Yes! Using lattices, fixed-points, μ-calculus

$$EG\ \phi\ \equiv\ \phi \wedge\ EX\ \phi \wedge\ EX\ EX\ \phi \wedge EX\ EX\ EX\ \phi \wedge \ldots$$

Huff & Ryan Logic for Computer Science Section 3.7,
but you can just trust me :)

$$EG\ \phi\ \equiv\ \bigwedge_{i=0}^{\infty}\ EX^i\ \phi \qquad \text{(where } EX^0\phi\ =\ \phi\text{)}$$

Symbolic Model Checking



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

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

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

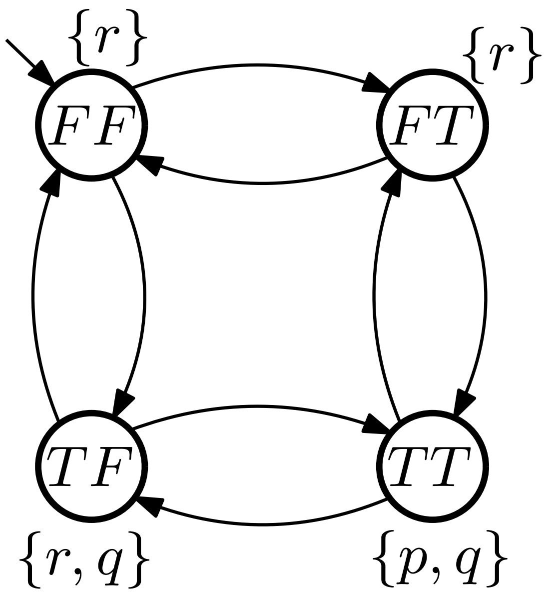
$$p \equiv x \wedge y \qquad q \equiv x \qquad r \equiv \neg(x \wedge y)$$

Transition Relation:

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

Let's compute $AF\ p$

Symbolic Model Checking



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

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

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

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

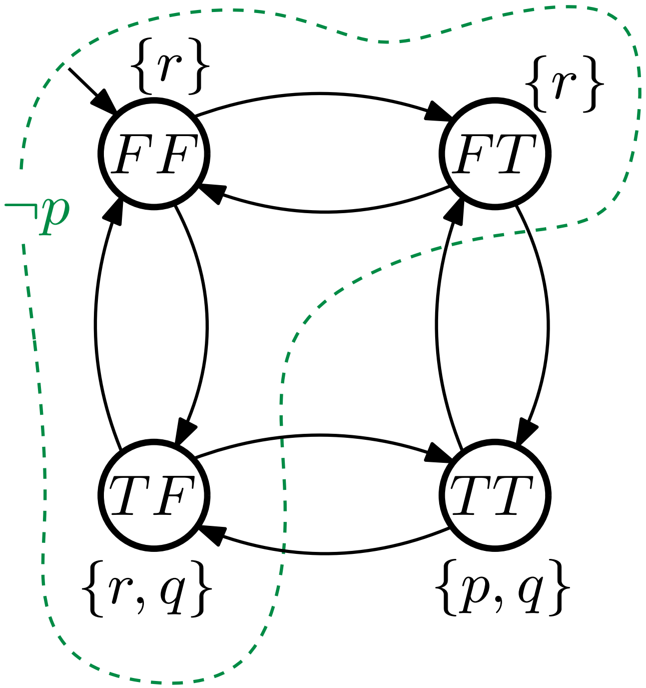
Transition Relation:

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

Let's compute $AF\ p$

$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$

Symbolic Model Checking



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

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

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

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

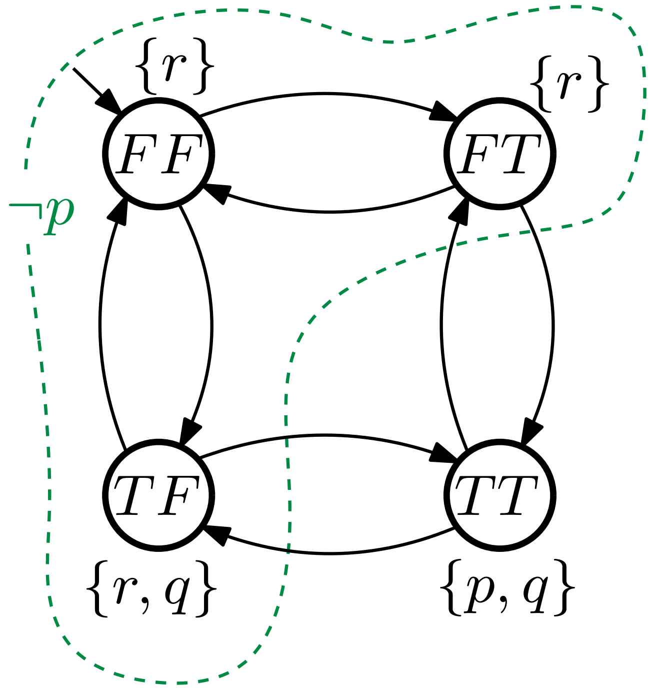
Transition Relation:

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

Let's compute $AF\ p$

$$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$$

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:

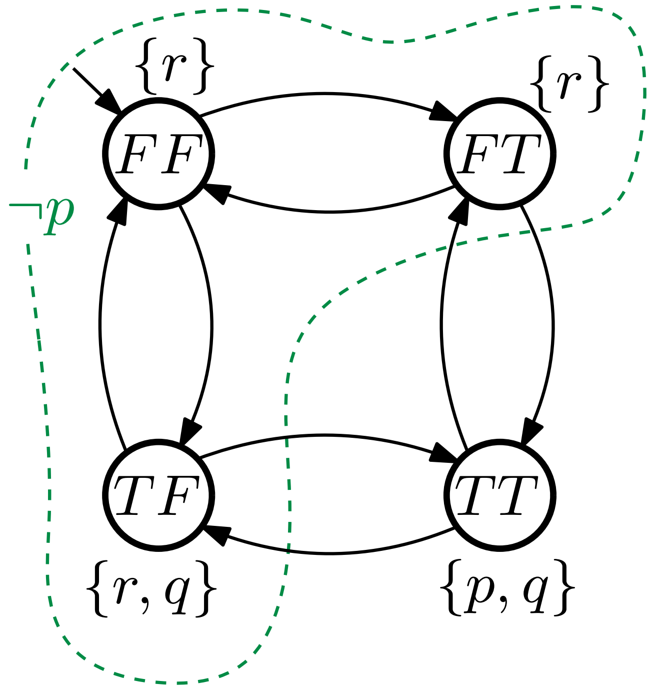
$$R \quad \equiv \quad (x' = x \wedge y' = \neg y) \vee (x' = \neg x \wedge y' = y)$$

Let's compute $AF\ p$

$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$

$EG(\neg x \vee \neg y)$

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:

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

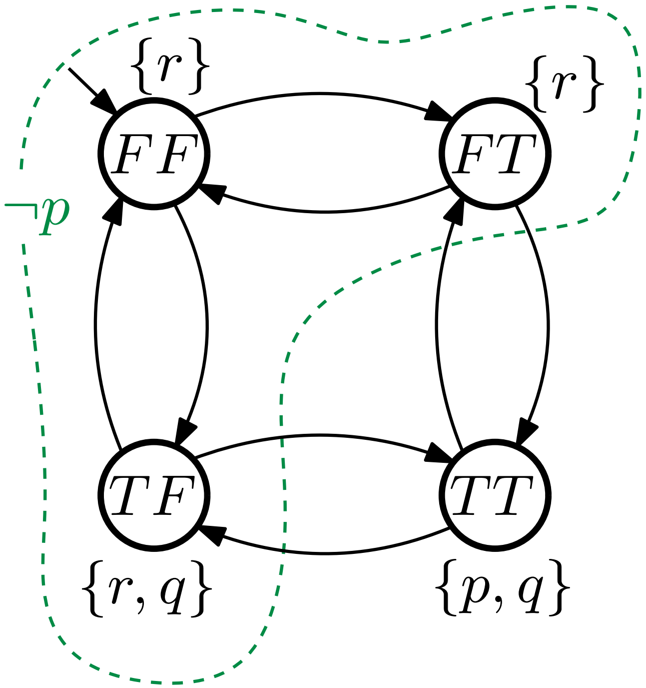
Let's compute $AF\ p$

$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$

$EG(\neg x \vee \neg y)$

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

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 $AF\ p$

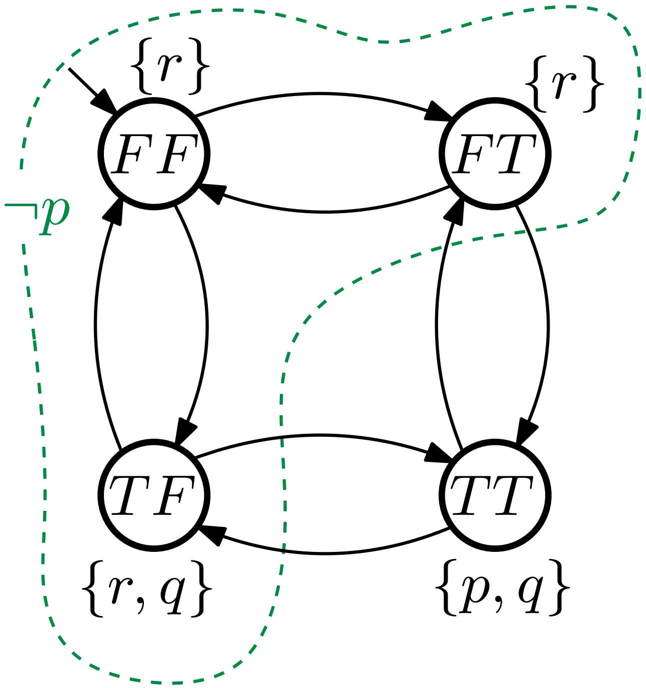
$$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$$

$$EG(\neg x \vee \neg y)$$

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

$$\equiv (\neg x \vee \neg y) \wedge T \wedge\ EX\ T \wedge\ EX\ EX\ T \ldots$$

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 $AF\ p$

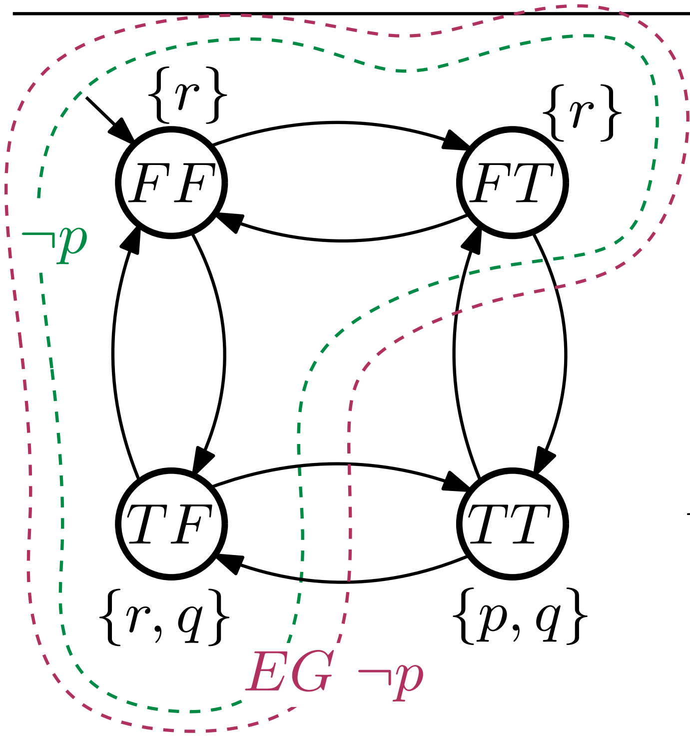
$$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$$

$$EG(\neg x \vee \neg y)$$

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

$$\equiv (\neg x \vee \neg y) \wedge T \wedge\ EX\ T \wedge\ EX\ EX\ T \ldots$$

Symbolic Model Checking



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

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

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

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

Transition Relation:

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

Let's compute $AF\ p$

$$AF\ p \equiv \neg EG \neg p \equiv \neg EG(\neg x \vee \neg y)$$

$$EG(\neg x \vee \neg y)$$

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

$$\equiv (\neg x \vee \neg y) \wedge T \wedge\ EX\ T \wedge\ EX\ EX\ T \ldots$$

$$\equiv (\neg x \vee \neg y) \wedge T \wedge T \wedge T \ldots$$

$$\equiv (\neg x \vee \neg y)$$

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.

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

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

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

```
(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)
```