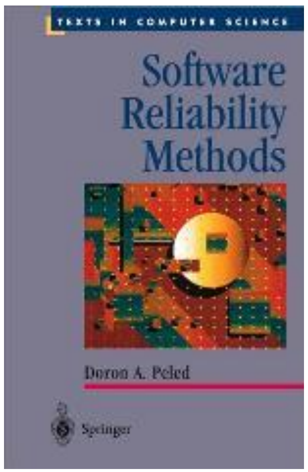
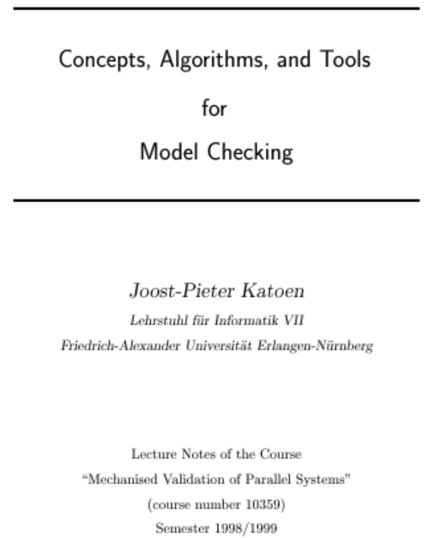
VPC 17-18
Computational tree logic (CTL)

Prof.ssa Susanna Donatelli
Universita' di Torino
www.di.unito.it
susi@di.unito.it

Reference material books:



Prof. Doron A. Peled
(University of Warwick, UK)



Concepts, Algorithms, and Tools

for

Model Checking

Joost-Pieter Katoen

Lehrstuhl für Informatik VII

Friedrich-Alexander Universität Erlangen-Nürnberg

Lecture Notes of the Course

"Mechanised Validation of Parallel Systems"

(course number 10359)

Semester 1998/1999

Prof. Jost-Pieter Katoen
(University of Aachen, D)

Acknowledgements

Some transparencies are adapted from the course notes and trasparencies of

- Prof. Doron A. Peled, University of Warwick (UK) and Bar Ilan University (Israel)
  http://www.dcs.warwick.ac.uk/~doron/srm.html

- Prof. Paul Gastin (MOVEP04 school)

Steps in the verification process

- Check the kind of system to analyze.
- Choose formalisms, methods and tools.
- Express system properties.
- Model the system.
- Apply methods.
- Obtain verification results.
- Analyze results.
- Identify errors.
- Suggest correction.

CTL main concepts

Computational Tree Logic, has been introduced by Clarke&Emerson in 1980

The *linear notion* of time (one single successor for each event) is substituted by a *branching notion of time* (each event has many successors, at each time instant there are many possible futures)

CTL is interpreted over a model in which R(s) is a set of states

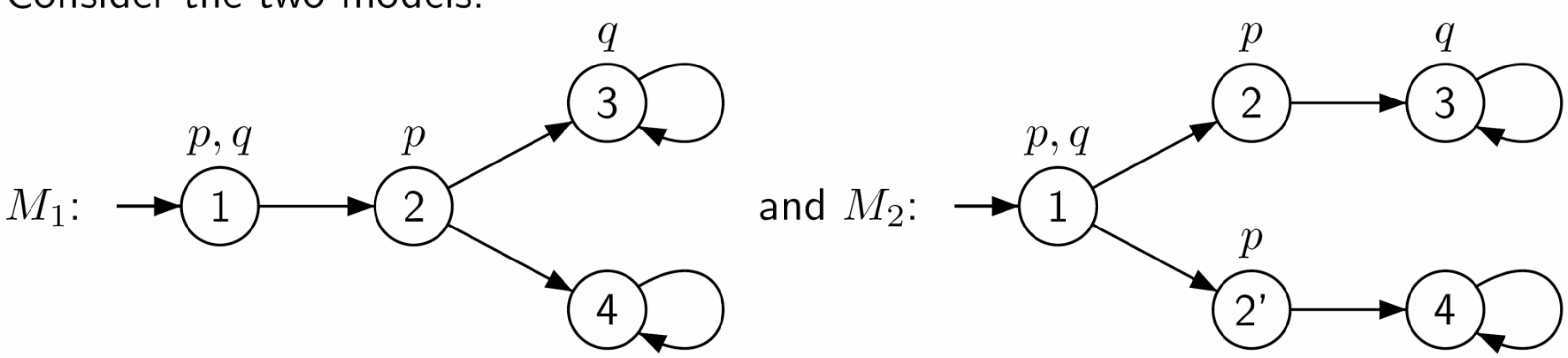
Possibility can't be expressed in LTL

Example

φ: Whenever p holds, it is possible to reach a state where q holds.

φ cannot be expressed in LTL.

Consider the two models:



$M_1 \models \varphi$ but $M_2 \not\models \varphi$

M_1 and M_2 satisfy the same LTL formulas.

CTL: Syntax

AP, set of atomic proposition. $p \in AP$.

CTL formulae:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid E[\varphi U \varphi] \mid A[\varphi U \varphi]$$

E: "for some path"

A: "for all paths"

EX: "for some path next"

U: until

Note: syntactically correct formulas quantifiers and temporal operators are in strict alternation

Derived operators

- $EF\varphi \equiv E[\text{true } U\ \varphi]$ “φ holds potentially” - “φ is possible”
- $AF\varphi \equiv A[\text{true } U\ \varphi]$ “φ is inevitable (unavoidable)”
- $EG\varphi \equiv \neg AF\neg\varphi$ “potentially always φ” – "globally along some path"
- $AG\varphi \equiv \neg EF\neg\varphi$ “invariantly φ”
- $AX\varphi \equiv \neg EX\neg\varphi$ “for all paths next”

CTL vs LTL

- LTL: statements about all paths starting in a state
- CTL: statements about all or some paths starting in a state
- Checking $E\varphi$ can be done in LTL using $A\neg\varphi$,
  (but it does not work for $AGEF\varphi$)
- Incomparable expressiveness
  - there are properties that can be expressed in LTL, but not in CTL
  - there are properties that can be expressed in CTL, but not in LTL
- Distinct model-checking algorithms, and their time complexities
- Distinct treatment of fairness

Semantic definition

CTL formulas are interpreted over Kripke structures

$$\mathcal{M}(S, R, L)$$

where

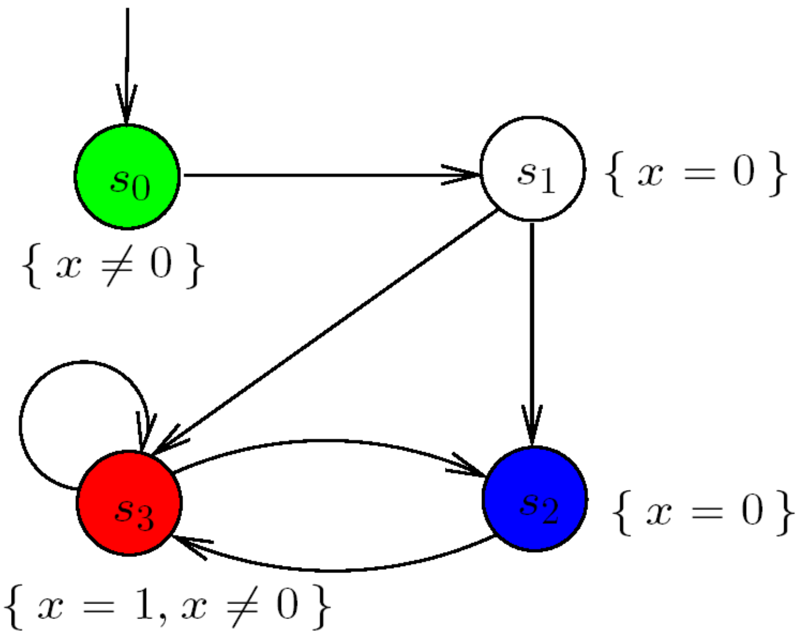
- S is a set of states
- R: S-->2^S is a successor function, assigning to s its set of successors R(s)
- L: S-->2^{AP}, is a labelling function

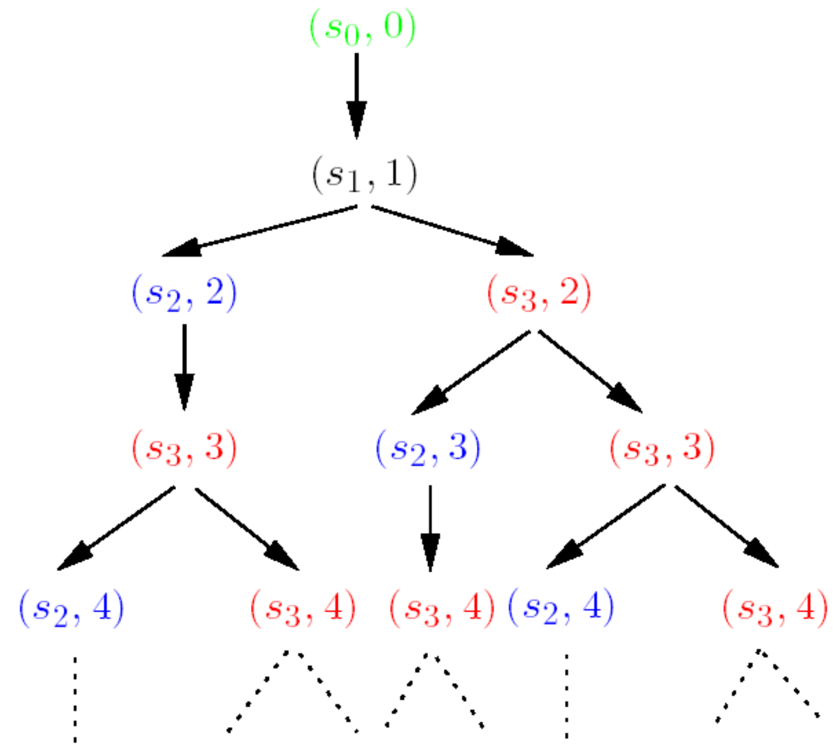
$\mathcal{M}$ can be seen as a tree of executions.

Given a model $\mathcal{M}$ and a formula φ, we define the satisfaction relation as $(\mathcal{M},s,\varphi) \in |=$, and we write $(\mathcal{M},s) |= \varphi$.

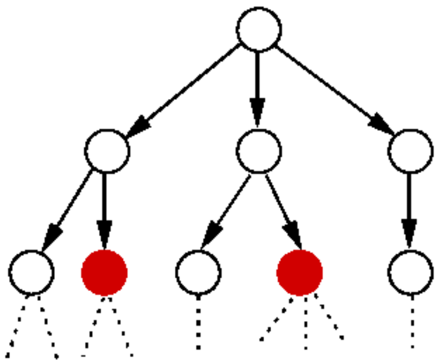
Semantic definition

A model M and its computation tree

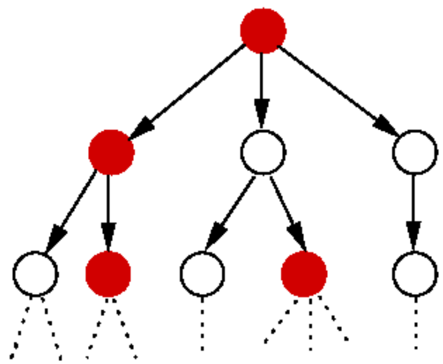




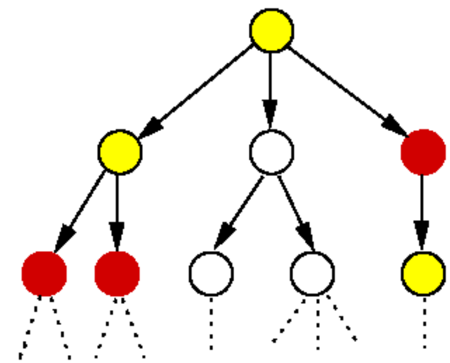
Semantic visualization



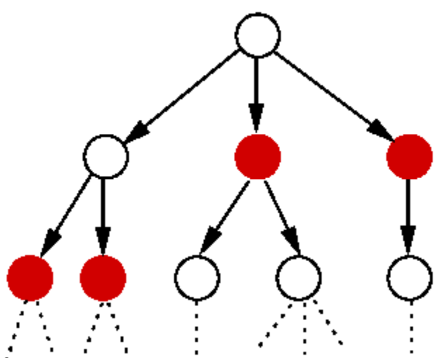
$\exists \Diamond\, \mathit{red}$ (EF *red*)



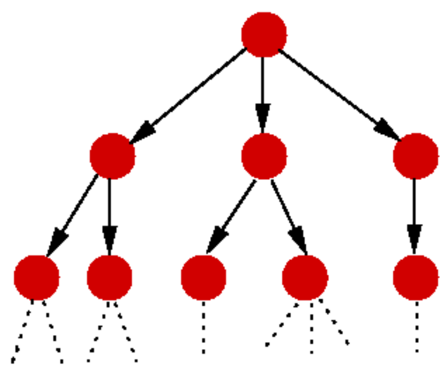
$\exists \Box\, \mathit{red}$



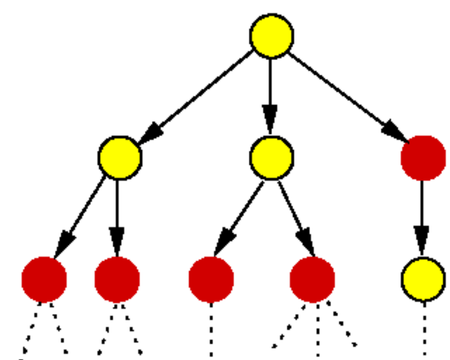
$\exists(\mathit{yellow}\ \mathsf{U}\ \mathit{red})$



$\forall \Diamond\, \mathit{red}$



$\forall \Box\, \mathit{red}$



$\forall(\mathit{yellow}\ \mathsf{U}\ \mathit{red})$

Formal semantics

Let $M(S, R, L)$ be a Kripke structure

Def: a path is an infinite sequence of states $s^0 s^1 s^2 \ldots\ldots$ such that $(s^i, s^{i+1}) \in R$

Def: if σ is a path, $\sigma[i]$ is the (i+1)-th element of the sequence

Def: $\mathcal{P}_M(s)$ is the set of all paths starting in s,

$$\mathcal{P}_M(s) = \{\sigma \in S^\omega \mid \sigma[0] = s\}$$

Def: s is a p-state if $p \in L(s)$

Def: σ is a p-path if it consists solely of p-states

Formal semantics

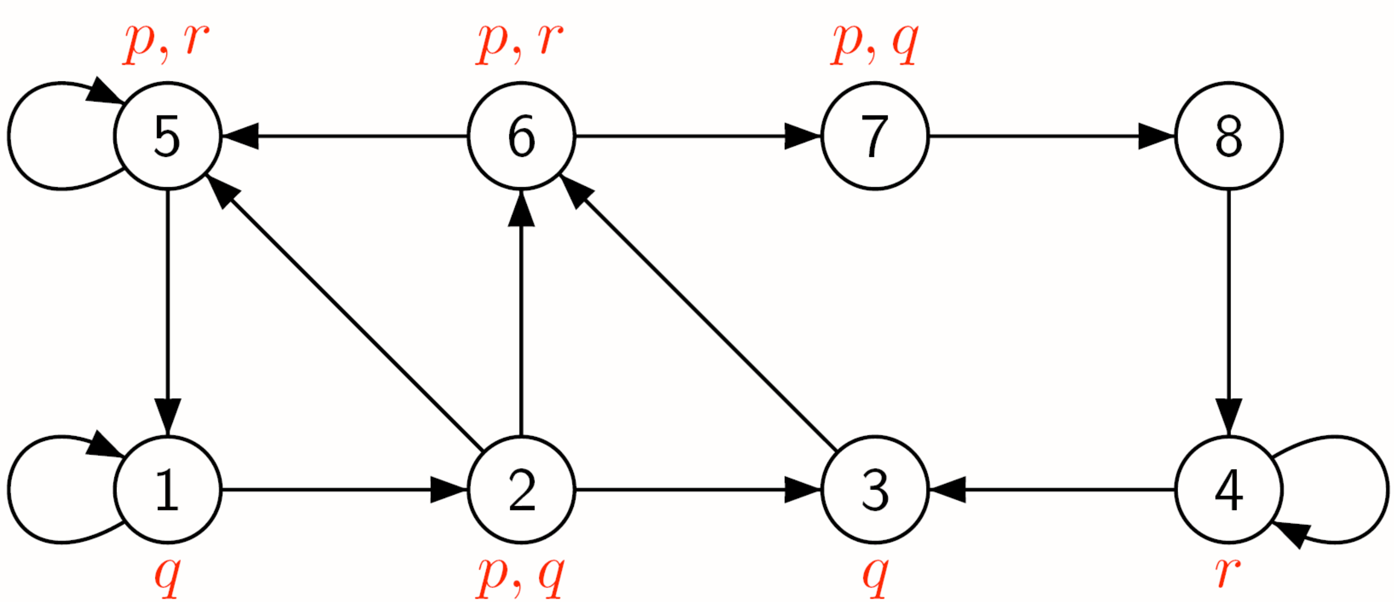
Given a Kripke structure *M*

- $s \models p$ iff $p \in L(s)$.
- $s \models \neg\varphi$ iff $\neg(s \models \varphi)$.
- $s \models \varphi\vee\psi$ iff $s \models \varphi \vee s \models \psi$.
- $s \models EX\varphi$ iff $\exists\sigma\in\mathcal{P}_M(s)$: $\sigma[1] \models\varphi$.
- $s \models E[\varphi\, U\, \psi]$ iff $\exists\sigma\in\mathcal{P}_M(s)$: $\exists j\geq 0$, $\sigma[j] \models\psi$
  $\wedge$ for each $0\leq k<j$, $\sigma[k] \models\varphi$.
- $s \models A[\varphi\, U\, \psi]$ iff $\forall\sigma\in\mathcal{P}_M(s)$: $\exists j\geq 0$, $\sigma[j] \models\psi$
  $\wedge$ for each $0\leq k<j$, $\sigma[k] \models\varphi$.

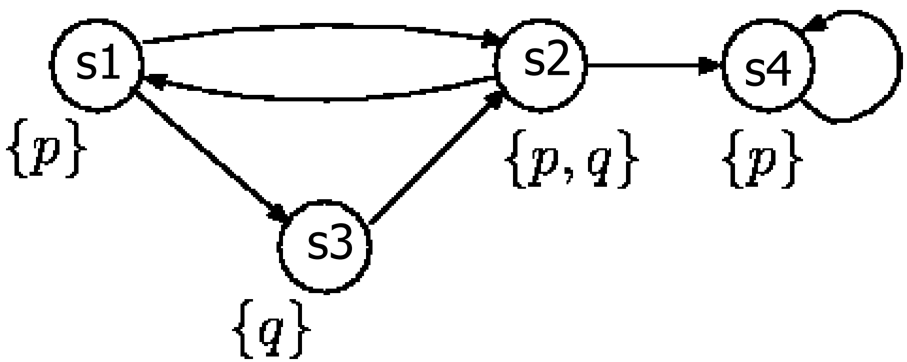
Examples

Sat(φ) = set of all states that satisfy φ. Compute Sat(φ) for:

- EX p
- AX p
- EF p
- AF p
- E q *U* r
- A q *U* r

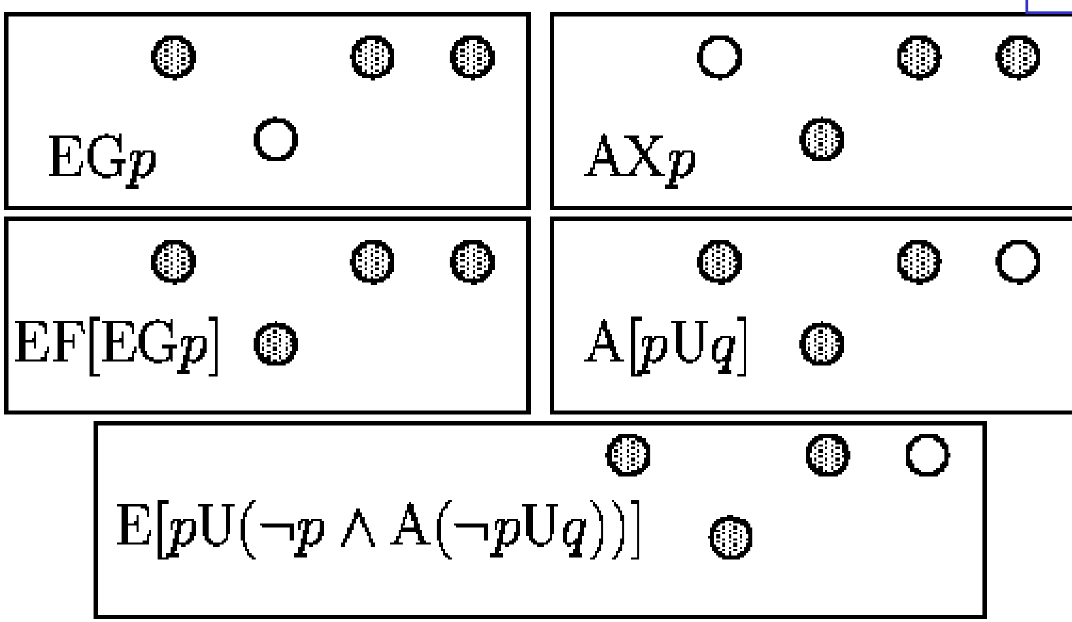


Examples

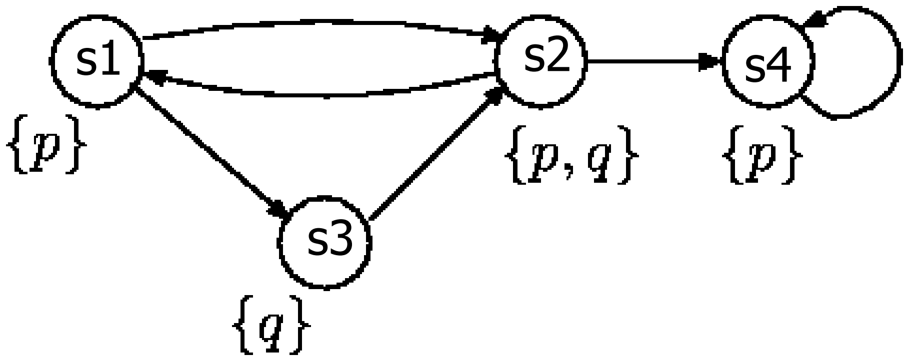


Color each state that satisfy the formula.

Sat(φ) = set of all states that satisfy φ.

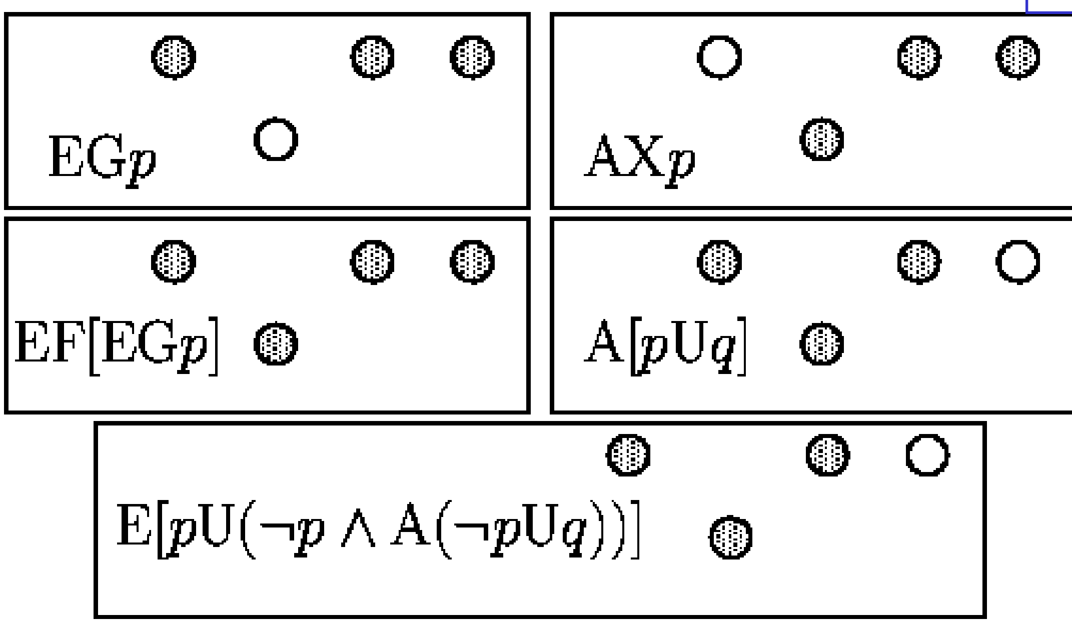


Examples



Color each state that satisfy the formula.

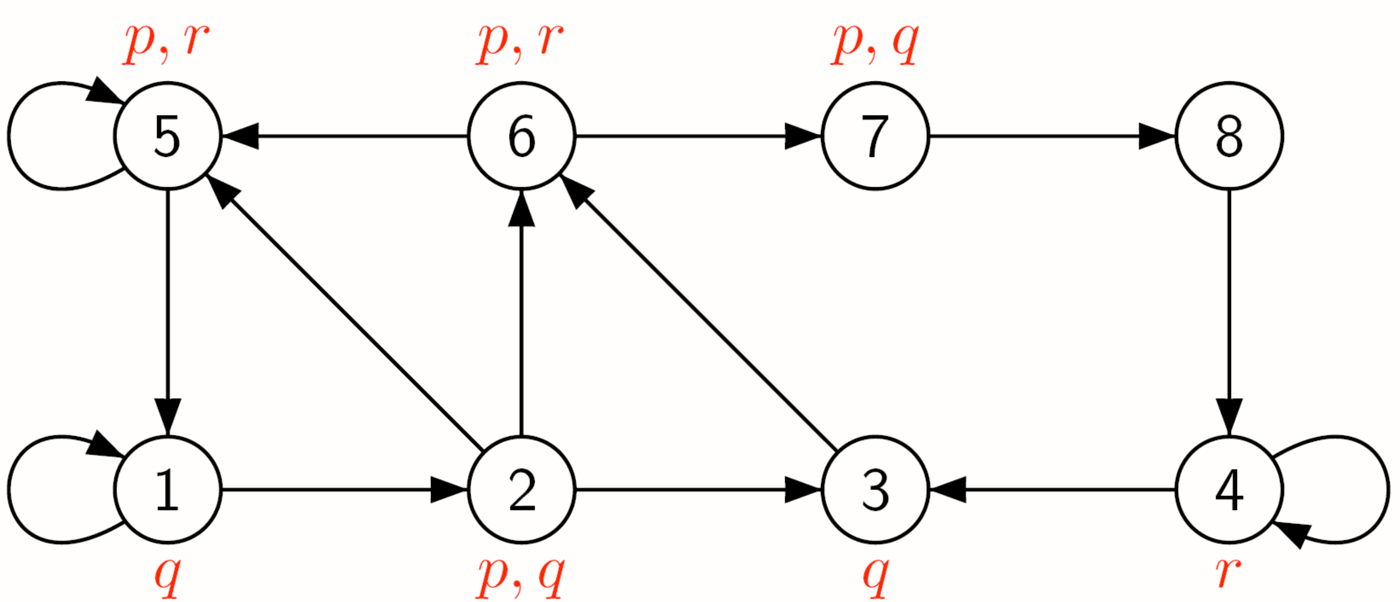
Sat(φ) = set of all states that satisfy φ.



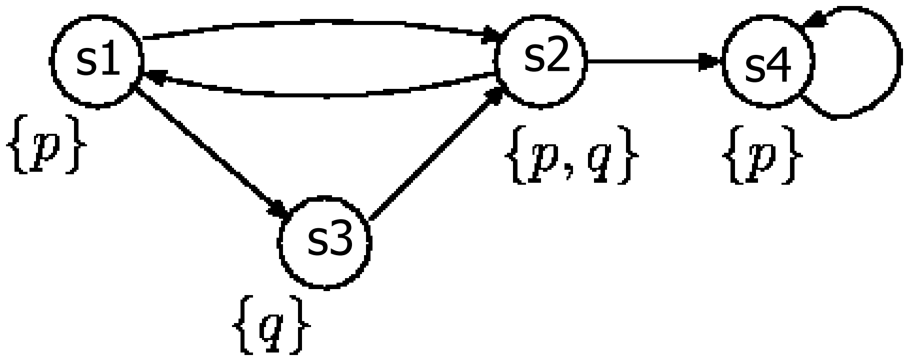
Examples

Sat(φ) = set of all states that satisfy φ. Compute Sat(φ) for:

- EX p
- AX p
- EF p
- AF p
- E q *U* r
- A q *U* r

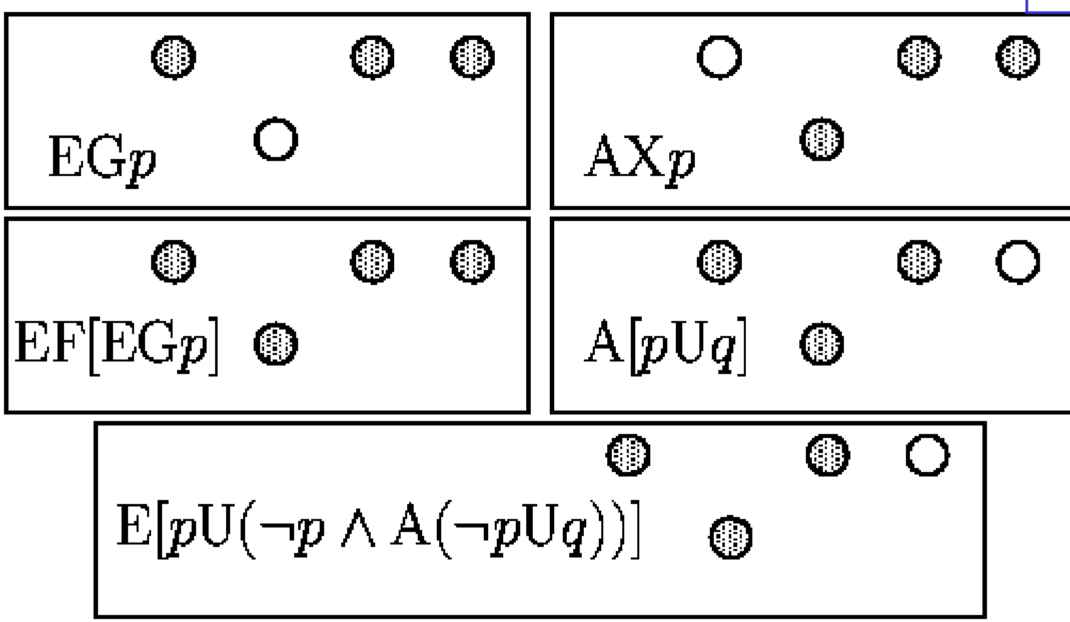


Examples



Color each state that satisfy the formula.

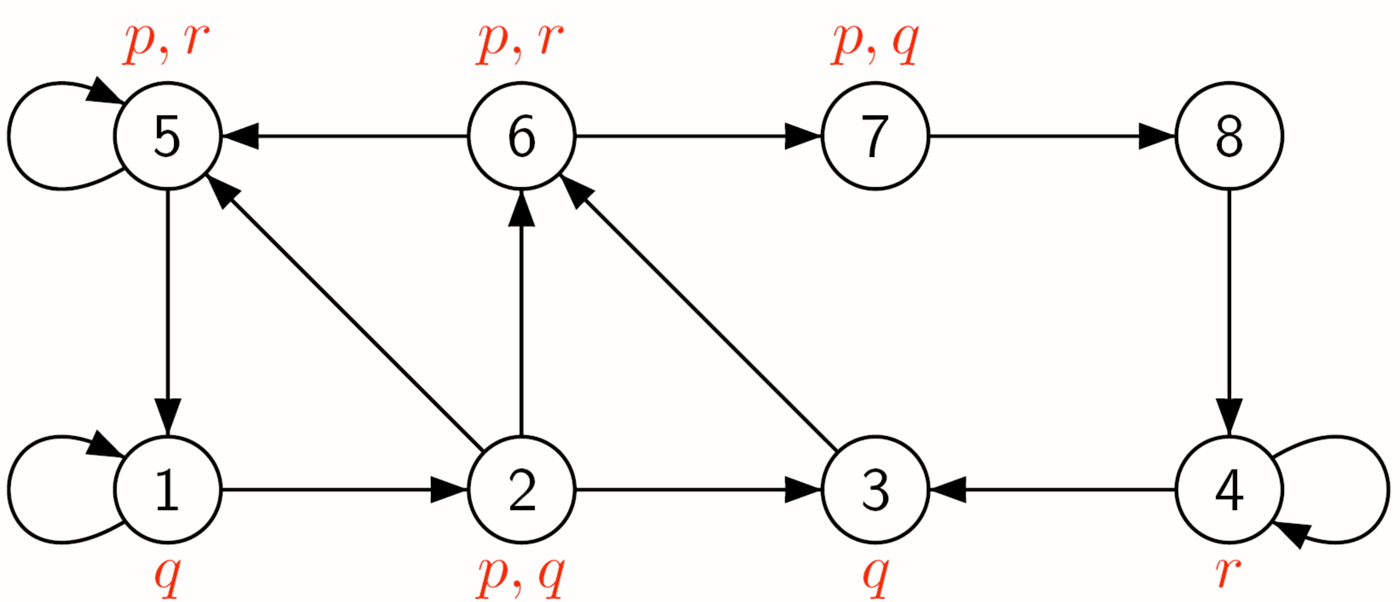
Sat(φ) = set of all states that satisfy φ.



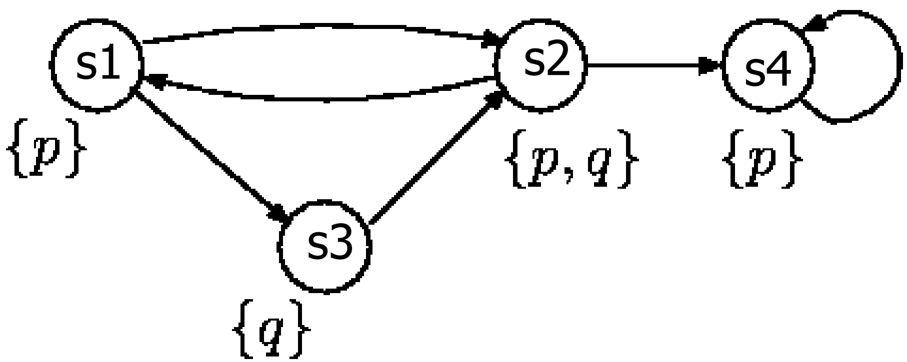
Examples

Sat(φ) = set of all states that satisfy φ. Compute Sat(φ) for:

- EX p
- AX p
- EF p
- AF p
- E q *U* r
- A q *U* r

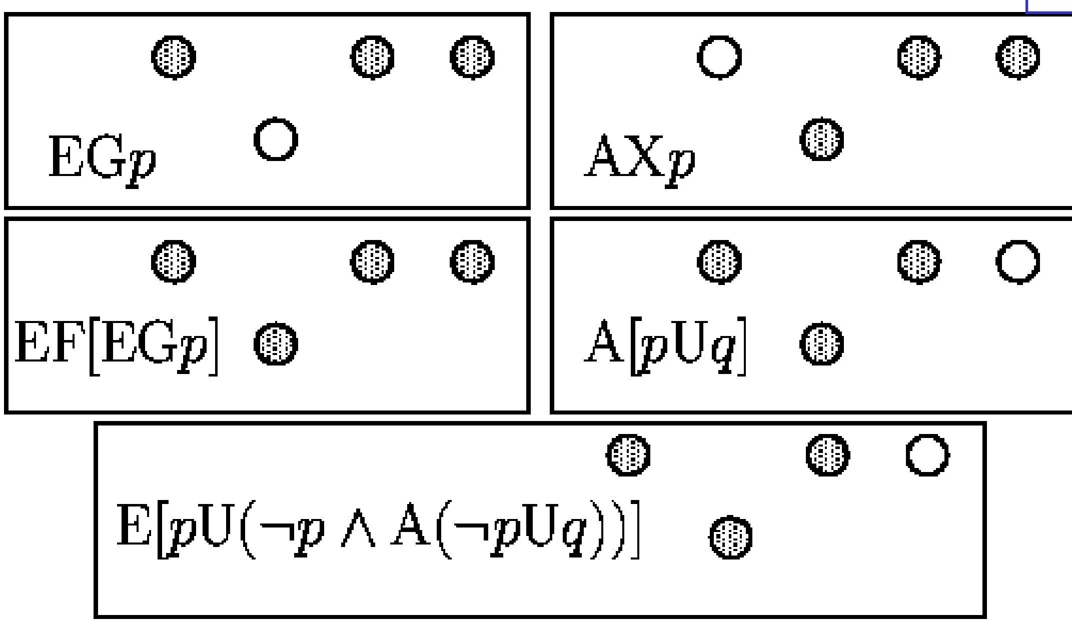


Examples

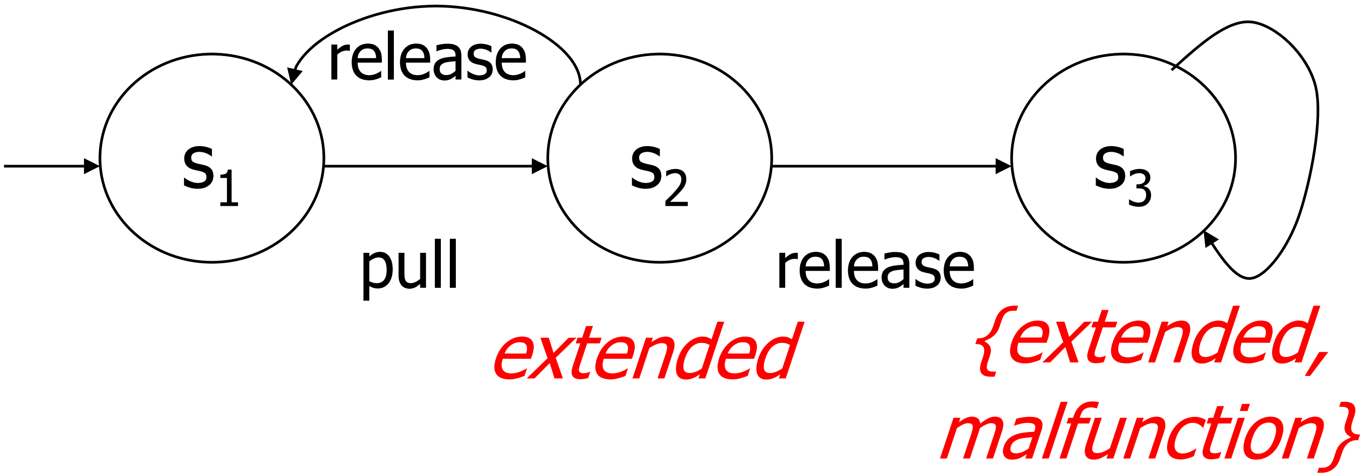


Color each state that satisfy the formula.

Sat(φ) = set of all states that satisfy φ.



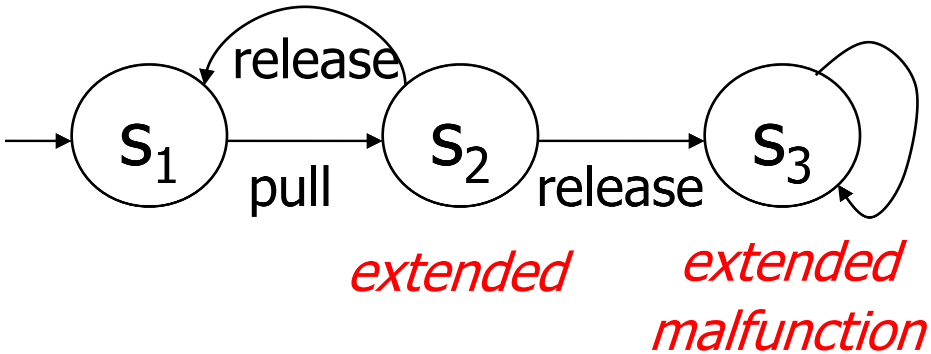
Spring Example



Computation tree?

…

CTL satisfaction examples



s_i |= EG extended ??

s_i |= AG extended ??

s_i |= AX extended ??

s_i |= AX EX extended ??

s_i |= AF extended ??

s_i |= AG extended ??

s_i |= AFEG extended ??

s_i |= AGEF extended ??

s_i |= A((¬extended) *U* malfunction)

s_i |= EG(¬extended->AX extended)

EG(extended ∨A X extended)

Some axioms (Peled's book notation)

Recall in LTL: $\varphi \, \mathsf{U} \, \psi \equiv \psi \vee (\varphi \wedge \bigcirc (\varphi \, \mathsf{U} \, \psi))$

Next

In CTL:

$$\forall(\Phi \, \mathsf{U} \, \Psi) \equiv \Psi \vee (\Phi \wedge \forall \bigcirc \forall(\Phi \, \mathsf{U} \, \Psi))$$

$$\forall \Diamond \Phi \equiv \Phi \vee \forall \bigcirc \forall \Diamond \Phi$$

$$\forall \Box \Phi \equiv \Phi \wedge \forall \bigcirc \forall \Box \Phi$$

$$\exists(\Phi \, \mathsf{U} \, \Psi) \equiv \Psi \vee (\Phi \wedge \exists \bigcirc \exists(\Phi \, \mathsf{U} \, \Psi))$$

$$\exists \Diamond \Phi \equiv \Phi \vee \exists \bigcirc \exists \Diamond \Phi$$

$$\exists \Box \Phi \equiv \Phi \wedge \exists \bigcirc \exists \Box \Phi$$

A

AF

AG

E

Some axioms

Recall in LTL: $\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$ and $\Diamond(\varphi \vee \psi) \equiv \Diamond\varphi \vee \Diamond\psi$

In CTL:

$$\forall\Box(\Phi \wedge \Psi) \equiv \forall\Box\Phi \wedge \forall\Box\Psi$$

$$\exists\Diamond(\Phi \vee \Psi) \equiv \exists\Diamond\Phi \vee \exists\Diamond\Psi$$

note that $\exists\Box(\Phi \wedge \Psi) \not\equiv \exists\Box\Phi \wedge \exists\Box\Psi$ and $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$

Some axioms

Recall in LTL: $\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$ and $\Diamond(\varphi \vee \psi) \equiv \Diamond\varphi \vee \Diamond\psi$

In CTL:

$$\forall\Box(\Phi \wedge \Psi) \equiv \forall\Box\Phi \wedge \forall\Box\Psi$$

$$\exists\Diamond(\Phi \vee \Psi) \equiv \exists\Diamond\Phi \vee \exists\Diamond\Psi$$

note that $\exists\Box(\Phi \wedge \Psi) \not\equiv \exists\Box\Phi \wedge \exists\Box\Psi$ and $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$

Comparing LTL and CTL

- Rewrite the syntax in state formulae and path formulae

- PLTL:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$$

- CTL (existential form)

state $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid E\psi$

path $\psi ::= \neg\psi \mid X\varphi \mid \varphi\, U\, \varphi$

Comparing LTL and CTL

Def: a CTL formula φ is equivalent to an LTL formula ψ ($\varphi \equiv \psi$) if, for any model M, we have

$$M \models \varphi \text{ iff } M \models \psi$$

Theorem: let φ be a CTL formula and ψ an LTL formula obtained from φ eliminating all paths quantifiers, then

- $\varphi \equiv \psi$ or
- an LTL formula equivalent to φ does not exists

LTL and CTL are incomparable

- There are LTL formula that *cannot be expressed* in CTL (an equivalent CTL formula does not exists)
  - FG p
  - F (p $\wedge$ X p)
  - G F p $\Rightarrow$ Fq if p holds infinitely often, then q will eventually hold

- There are CTL formula that *cannot be expressed* in LTL (an equivalent LTL formula does not exists)
  - AF AG p
  - AF (p $\wedge$ AX p)
  - AG EF p

LTL and CTL are incomparable

To show that they are incomparable we need to exhibit

- a formula LTL for which no corresponding equivalent CTL formula exists

AND

- a formula CTL for which no corresponding equivalent LTL formula exists

The proof relies on the "syntactical theorem" that limits the state space of the search for equivalent formulas of a given formula (remember that all LTL formula are implicitly quantified as "forall", as we are verifying the all model M, and not only an execution)

LTL and CTL are incomparable

Sketch of proof

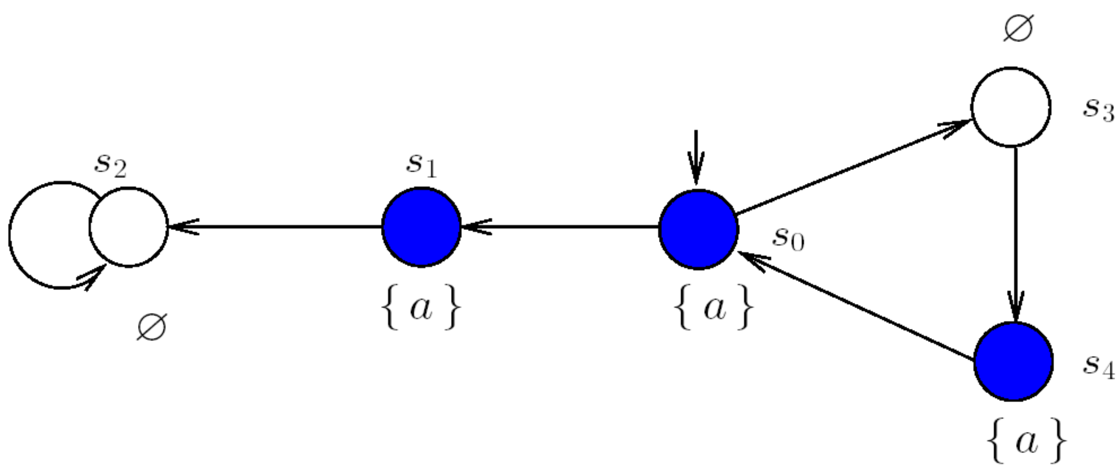
LTL does not imply CTL: given a formula LTL show that for all choices of quantifiers "addition" it is possible to exibit a model for which one formula is satisfied and the other is not

CTL does not imply LTL: remove all quantifiers and exibit a model for which one formula is satisfied and the other is not

LTL and CTL are incomparable

The LTL formula F(a ∧ X a) is not equivalent to the CTL formula AF(a ∧ AX a)

$\Diamond(a \wedge \bigcirc a)$ is not equivalent to $\forall\Diamond(a \wedge \forall\bigcirc a)$



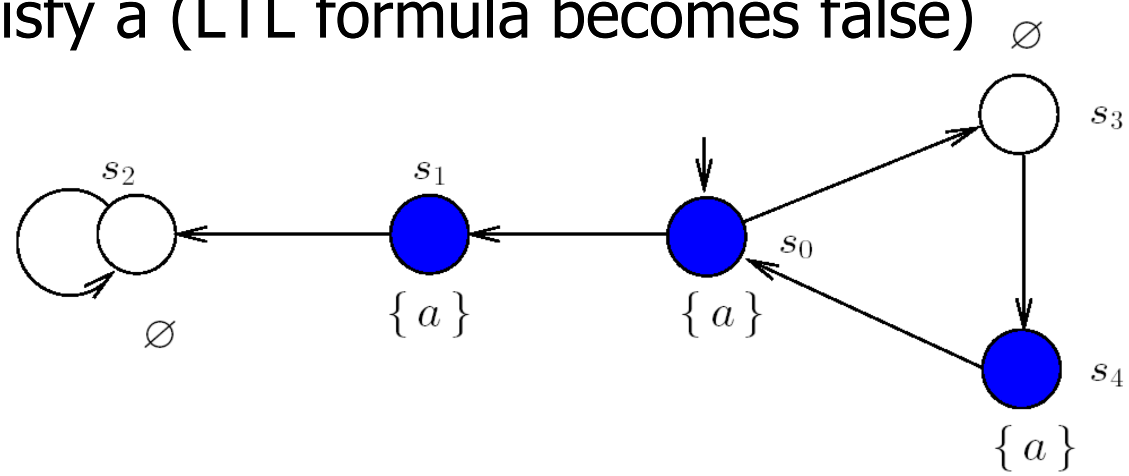
s_0|= F(a ∧ X a) but not s_0|=AF(a ∧ AX a)

path $s_0\, s_1\, (s_2)^\omega$ violates it

LTL and CTL are incomparable

The LTL formula F(a $\wedge$ X a) is not equivalent to the CTL formula AF(a $\wedge$ EX a)

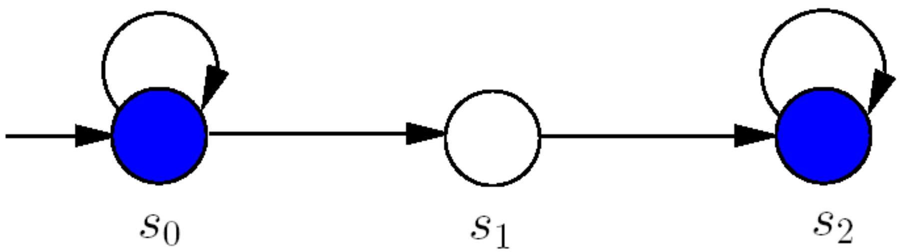
It is enough to take a model in which s_4 does not satisfy a (LTL formula becomes false)



Prop: the LTL formula F(a $\wedge$ X a) has no equivalent in CTL

LTL and CTL are incomparable

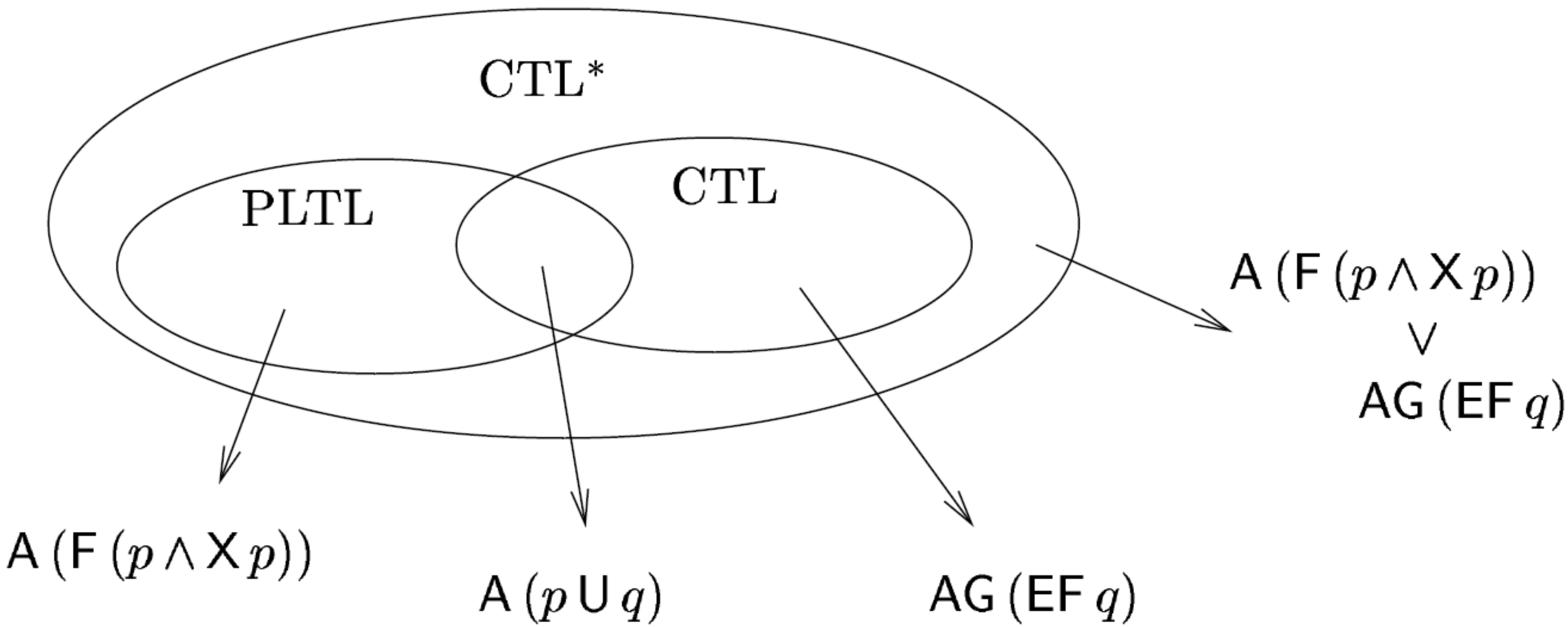
The CTL formula AF AG a is not equivalent to the LTL formula F G a



$s_0 \models$ F G a   but   not $s_0 \models$ AF AG a

path s_0^ω violates it

LTL and CTL are incomparable



CTL* (existential form)

state $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid E\psi$

path $\psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid X\,\psi \mid \psi\, U\, \psi$

Model checking CTL

Problem definition: given a model M, a state s, and a CTL formula φ, does $(M,s) \models \varphi$?

In practice the algorithm solves the problem: given a model M and a CTL formula φ, which are the states s, for which $(M,s) \models \varphi$?

As a by-product, at zero cost, the algorithm also computes all states that satisfy the subformulae of φ.

Model checking CTL

Definition of sub-formulae. Let p in AP, φ and ψ be CTL formulae, then the set of sub-formulae is defined as:

$$
\begin{aligned}
Sub(p) &= \{p\} \\
Sub(\neg\varphi) &= Sub(\varphi) \cup \{\neg\varphi\} \\
Sub(\varphi \vee \psi) &= Sub(\varphi) \cup Sub(\psi) \cup \{\varphi \vee \psi\} \\
Sub(EX\varphi) &= Sub(\varphi) \cup \{EX\,\varphi\} \\
Sub(E[\varphi\, U\, \psi]) &= Sub(\varphi) \cup Sub(\psi) \cup \{E[\varphi\, U\, \psi]\} \\
Sub(A[\varphi\, U\, \psi]) &= Sub(\varphi) \cup Sub(\psi) \cup \{A[\varphi\, U\, \psi]\}
\end{aligned}
$$

Model checking CTL

The algorithms starts with sub-formulae of length 1, and proceed by induction, until the formula of length $|\varphi|$ is computed

Usually S: *set of State,* is global

function Sat(φ: *CTL formula,* S: *set of State*): *set of* State

(* precondition: true*)

begin

if φ=true --> return S

[] φ=false --> return $\emptyset$

[] $\varphi \in AP$ --> return $\{s \mid \varphi \in L(s)\}$

Model checking CTL

[] $\varphi = \neg\varphi_1$ --> return S - Sat(φ_1)

[] $\varphi=\varphi_1 \lor \varphi_2$ --> return Sat(φ_1) $\cup$ Sat(φ_2)

[] φ=EXφ_1 --> return {s $\in$ S| $\exists$ (s,s') $\in$ R $\land$ s' $\in$ Sat(φ_1)}

[] φ=E[φ_1Uφ_2] --> return $Sat_{EU}(\varphi_1, \varphi_2)$

[] φ=A[φ_1Uφ_2] --> return $Sat_{AU}(\varphi_1, \varphi_2)$

(* postcondition: Sat(φ) = {s $\in$ S | (M,s) |= φ}

end

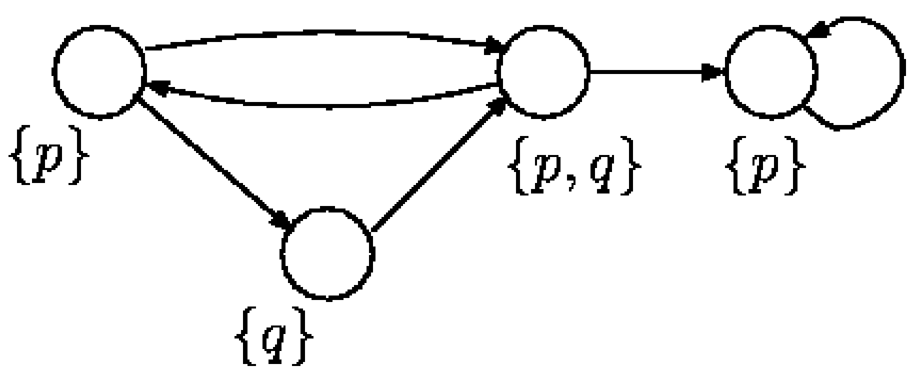
Model checking CTL

$Sat_{EU}(\varphi_1, \varphi_2)$ and $Sat_{AU}(\varphi_1, \varphi_2)$ are fixed point algorithms that use the axiom of the Until in terms of neXt and Until

Model checking CTL

function $Sat_{EU}\ (\phi, \psi : Formula)$: **set of** $State$;
(* precondition: true *)
begin var Q, Q' : **set of** $State$;
    $Q, Q' := Sat(\psi), \varnothing$;
    do $Q \neq Q' \longrightarrow$
        $Q' := Q$;
        $Q := Q \cup (\{ s \mid \exists\, s' \in Q.\, (s, s') \in R \} \cap Sat(\phi))$
    od;
    return Q
(* postcondition: $Sat_{EU}(\phi, \psi) = \{ s \in S \mid \mathcal{M}, s \models \mathsf{E}\,[\phi\, \mathsf{U}\, \psi] \}$ *)
end

```
function Sat_EU (φ, ψ : Formula) : set of State;
(* precondition: true *)
begin var Q, Q' : set of State;
      Q, Q' := Sat(ψ), ∅;
      do Q ≠ Q' ⟶
          Q' := Q;
          Q := Q ∪ ({ s | ∃ s' ∈ Q. (s, s') ∈ R } ∩ Sat(φ))
      od;
      return Q
(* postcondition: Sat_EU(φ, ψ) = { s ∈ S | M, s ⊨ E [φ U ψ] } *)
end
```



Model checking CTL

function $Sat_{AU}\,(\phi, \psi : Formula)$: **set of** $State$;

(* precondition: true *)

begin var Q, Q' : **set of** $State$;

$Q, Q' := Sat(\psi), \varnothing$;

do $Q \neq Q' \longrightarrow$

$Q' := Q$;

$Q := Q \cup (\{\, s \mid \forall s' \in Q.\,(s, s') \in R \,\} \cap Sat(\phi))$

{s| ∀ s': (s,s') ∈ R, s' ∈ Q}

od;

return Q

(* postcondition: $Sat_{AU}(\phi, \psi) = \{\, s \in S \mid \mathcal{M}, s \models \mathsf{A}\,[\phi \,\mathsf{U}\, \psi]\,\}$ *)

end

(* precondition: true *)

begin var Q, Q' : **set of** $State$;

  $Q, Q' := Sat(\psi), \varnothing$;

  do $Q \neq Q' \longrightarrow$

    $Q' := Q$;

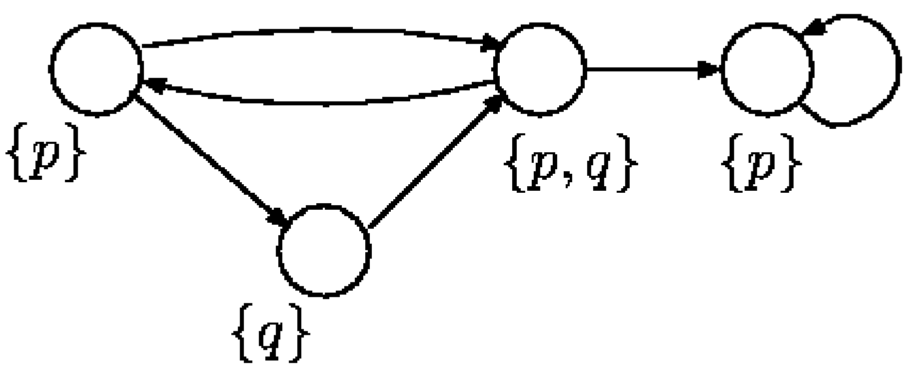
    $Q := Q \cup (\{\, s \mid \forall\, s' \in Q.\, (s, s') \in R \,\} \cap Sat(\phi))$

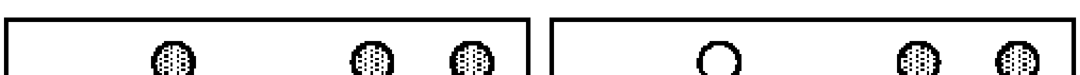
  od;

  return Q

(* postcondition: $Sat_{AU}(\phi, \psi) = \{\, s \in S \mid \mathcal{M}, s \models \mathsf{A}\,[\phi\, \mathsf{U}\, \psi] \,\}$ *)

{s| ∀ s': (s,s') ∈ R and s' ∈ Q}





Complexity of CTL model checking

- Sat(φ) is computed |Sub(φ)| times, and |Sub(φ)| is proportional to $|\varphi|$
- $Sat_{AU}(\varphi_1, \varphi_2)$ is proportional to $|Sys|^3$, since the iteration is traversed at most |Sys| and the "forall" inside depend on the pairs in R (at most $|Sys|^2$)

Total complexity amounts to $O(|\varphi| \times |Sys|^3)$

More efficient algorithms gets to $O(|\varphi| \times |Sys|^2)$

CTL and fairnes: motivations

Recall the following piece of code:

$$\textbf{process}\ \mathsf{Inc} \quad = \quad \textbf{while}\ \langle\, x \geqslant 0\ \textbf{do}\ x := x + 1\,\rangle\ \textbf{od}$$

$$\textbf{process}\ \mathsf{Reset} \quad = \quad x := -1$$

where $\langle .. \rangle$ means "atomic execution".

Does the program satisfies "F terminates"? No, since there is an execution in which only Inc is executed.

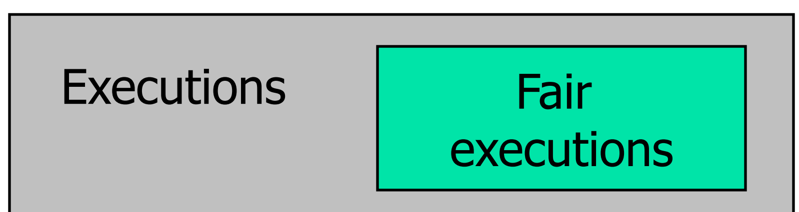
This situation is not possible if the OS schedule is fair, and we would like to rule-out from the model checking whose executions that are not fair

Fair executions: solutions

We want to consider only execution with fair behaviour.

Can be done:

- enforcing fairness in the formula: we should check whether fairness can be expressed in CTL
- modifying the MC algorithm as to consider only fair executions



Recall the LTL fairness definitions

- *Unconditional fairness:*
  - GF ψ also stated as true $\Rightarrow$ GF ψ
- *Weak fairness (justice):*
  - FG $\varphi \Rightarrow$ GF ψ (as in: FG enab(a) $\Rightarrow$ GF exec(a)
- *Strong transition fairness:*
  - GF $\varphi \Rightarrow$ GF ψ

Weak and strong cannot be expressed in CTL

Therefore: modify the model checking algorithm, defining a Fair-model for CTL

Fair executions: solutions

A fair CTL-model is a quadruple $\mathcal{M}$ = (S,R,L,$\mathcal{F}$), where (S,R,L) is a CTL-model and $\mathcal{F} \subseteq 2^S$ is a set of fairness constraints

$\mathcal{F} = \{F^1, F^2, \ldots\}$

A path $\sigma = s^0 s^1 s^2 \ldots\ldots$ is $\mathcal{F}$-fair if for every set of states $F^i \in \mathcal{F}$, there are infinitely many states in σ that belong to F^i

If lim(σ): set of states of σ visited infinitely often, then σ if $\mathcal{F}$-fair if $\lim(\sigma) \cap F^i \neq \emptyset$, for all i

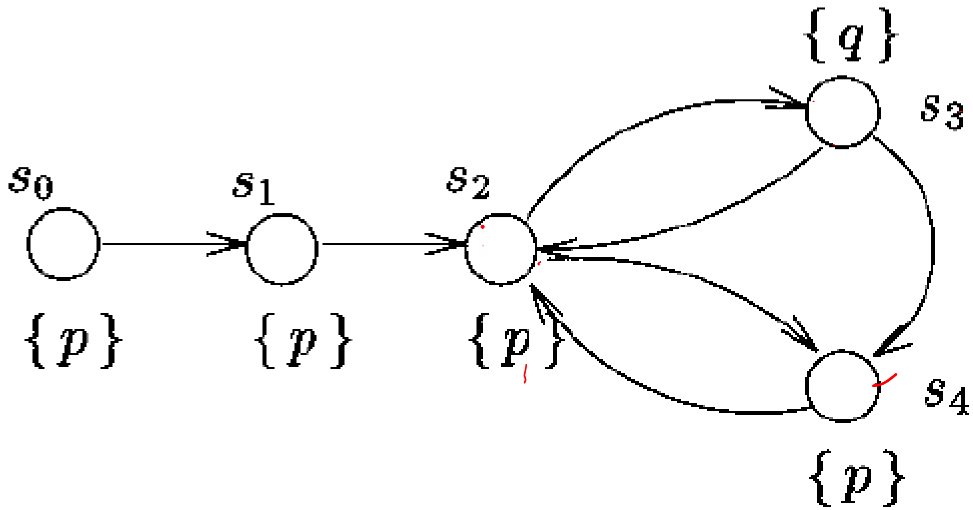
$\mathcal{P}^f_{\mathcal{M}}(s)$: set of $\mathcal{F}$-fair paths starting in s

Fair executions: modified semantics

Given a Kripke structure *M*

- $s \models_f p$ iff $p \in L(s)$.
- $s \models_f \neg\varphi$ iff $\neg(s \models_f \varphi)$.
- $s \models_f \varphi\vee\psi$ iff $s \models_f = \varphi \vee s \models_f \psi$.
- $s \models_f EX\varphi$ iff $\exists\sigma\in\mathcal{P}^f_M(s)$: $\sigma[1] \models_f \varphi$.
- $s \models_f E[\varphi\,U\,\psi]$ iff $\exists\sigma\in\mathcal{P}^f_M(s)$: $\exists j\geq 0$, $\sigma[j] \models_f \psi$
  $\wedge$ for each $0\leq k<j$, $\sigma[k] \models\varphi$.
- $s \models_f A[\varphi\,U\,\psi]$ iff $\forall\sigma\in\mathcal{P}^f_M(s)$: $\exists j\geq 0$, $\sigma[j] \models_f \psi$
  $\wedge$ for each $0\leq k<j$, $\sigma[k] \models\varphi$.

Fair executions: example



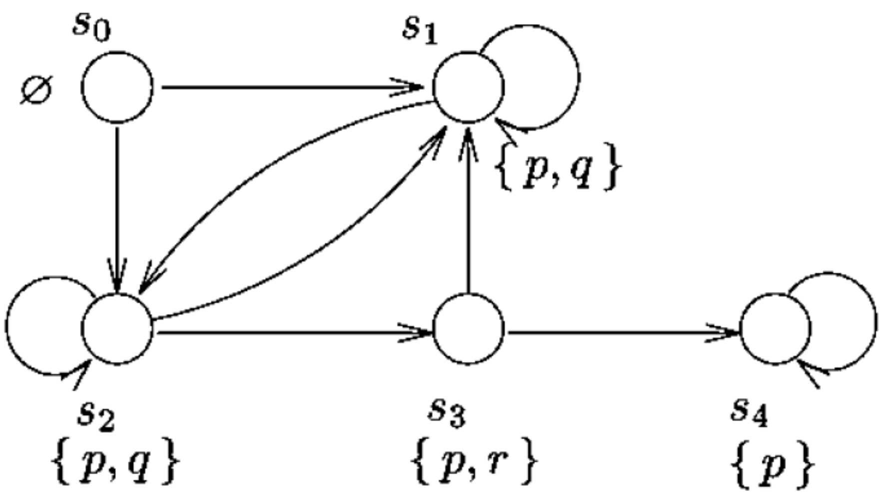
$(M, s_o) \models AG[p \rightarrow AF\ q]$ - false,

but with $F = \{F^1, F^2\}$, with $F^1 = \{s_3\}$ and $F^2 = \{s_4\}$

$(M, s_o) \models_f AG[p \rightarrow AF\ q]$

Posso togliere F_2 ?

Exercise on CTL



1. $\mathsf{EG}\, p$
2. $\mathsf{AG}\, p$
3. $\mathsf{EF}\, [\mathsf{AG}\, p]$
4. $\mathsf{AF}\, [p\, \mathsf{U}\, \mathsf{EG}\, (p \Rightarrow q)]$
5. $\mathsf{EG}\, [((p \wedge q) \vee r)\, \mathsf{U}\, (r\, \mathsf{U}\, \mathsf{AG}\, p)]$

Check the validity of the formulae in each state

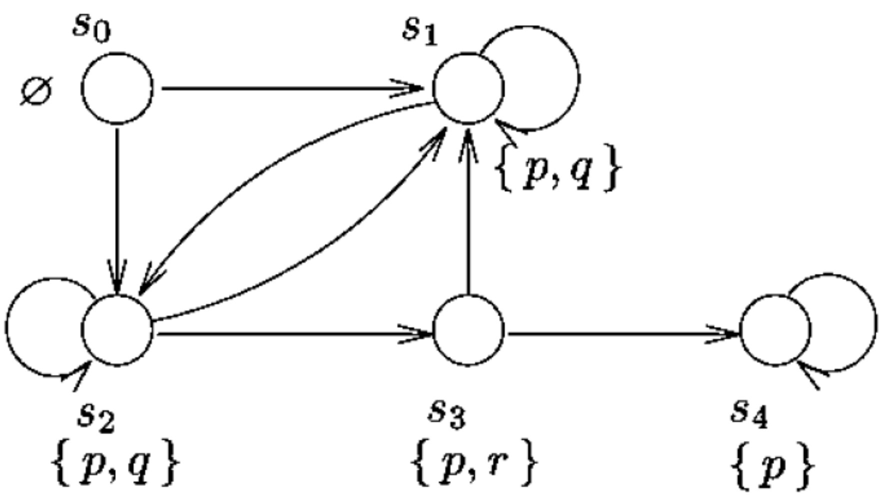
EFφ ≡ E[true U φ] "φ holds potentially"

AFφ ≡ A[true U φ] "φ is inevitable"

EGφ ≡ ¬AF¬φ "potentially always φ"

AGφ ≡ ¬EF¬φ "invariantly φ"

Exercise on CTL



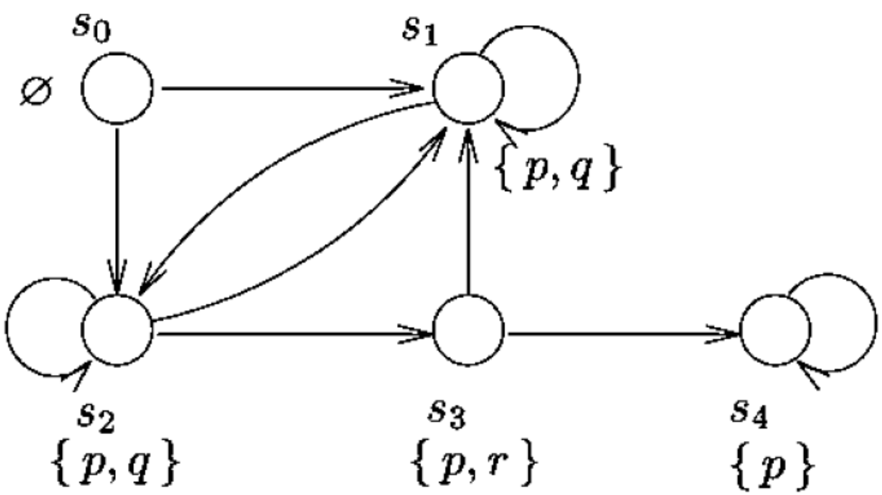
EFp ≡ E[true U p]

AFp ≡ A[true U p]

EFp: start with Q = {s1, s2, s3, s4} and in one step add s0, and at the next iteration the algorithm stops

AFp: start with Q = {s1, s2, s3, s4} and in the next step consider s0. S0 can be added only if all arcs out of s0 are in Q

Exercise on CTL



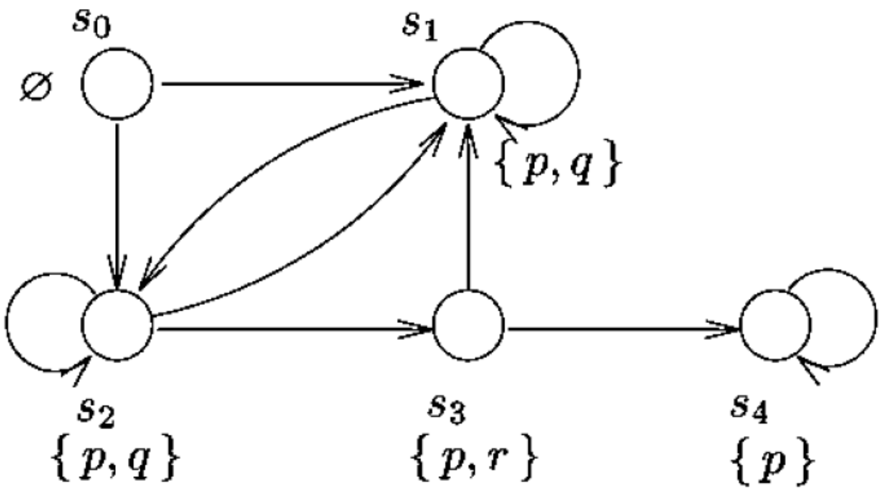
EGp ≡ ¬AF¬p ≡ ¬A[true U ¬p]

AGp ≡ ¬EF¬p ≡ ¬E[true U ¬p]

EGp: the result is the complement of the states that satisfy AF¬p that can be computed as before

AGp: the result is the complement of the states that satisfy EF¬p

Exercise on CTL



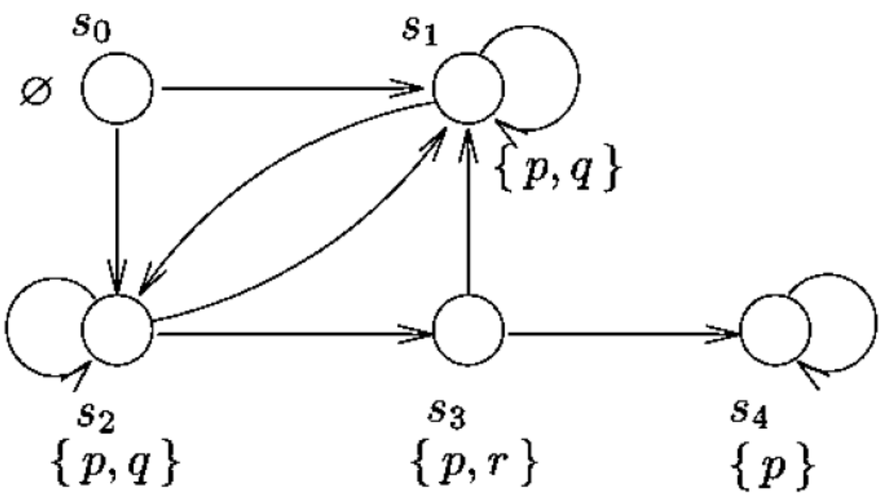
$EFq \equiv E[true\ U\ q]$

$AFq \equiv A[true\ U\ q]$

EFq: start with Q = {s1, s2} and in one step add s0, and s3, and at the next iteration the algorithm stops

AFq: start with Q = {s1, s2} and in the next step s0 is added. At the next iteration no new element is added and the algorithm stops.

Exercise on CTL



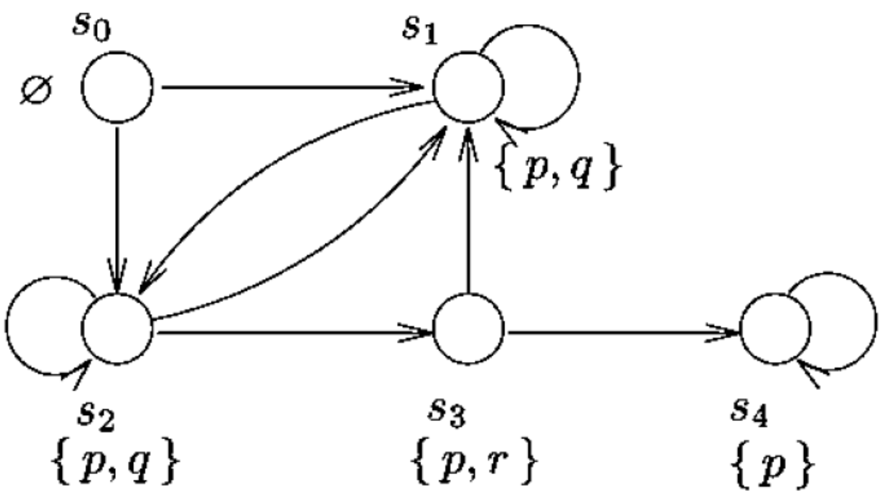
EGq ≡¬AF¬q ≡ E[true U q]

AGq ≡ ¬EF¬q ≡A[true U q]

EGq: the result is the complement of the states that satisfy AF¬q that can be computed as before

AGq: the result is the complement of the states that satisfy EF¬q

Exercise on CTL



1. $\mathsf{EG}\, p$
2. $\mathsf{AG}\, p$
3. $\mathsf{EF}\, [\mathsf{AG}\, p]$
4. $\mathsf{AF}\, [p \,\mathsf{U}\, \mathsf{EG}\, (p \Rightarrow q)]$
5. $\mathsf{EG}\, [((p \wedge q) \vee r) \,\mathsf{U}\, (r \,\mathsf{U}\, \mathsf{AG}\, p)]$

Check the validity of the formulae in each state

End of CTL