A Quick Tour on CTL Model Checking

Lecture #2 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

October 26, 2006

Linear and branching temporal logic

- *Linear* temporal logic:

  "statements about (all) paths starting in a state"

  - $s \models \Box(x \leqslant 20)$ iff for all possible paths starting in s always $x \leqslant 20$

- *Branching* temporal logic:

  "statements about all or some paths starting in a state"

  - $s \models \forall\Box(x \leqslant 20)$ iff for **all** paths starting in s always $x \leqslant 20$
  - $s \models \exists\Box(x \leqslant 20)$ iff for **some** path starting in s always $x \leqslant 20$
  - nesting of path quantifiers is allowed

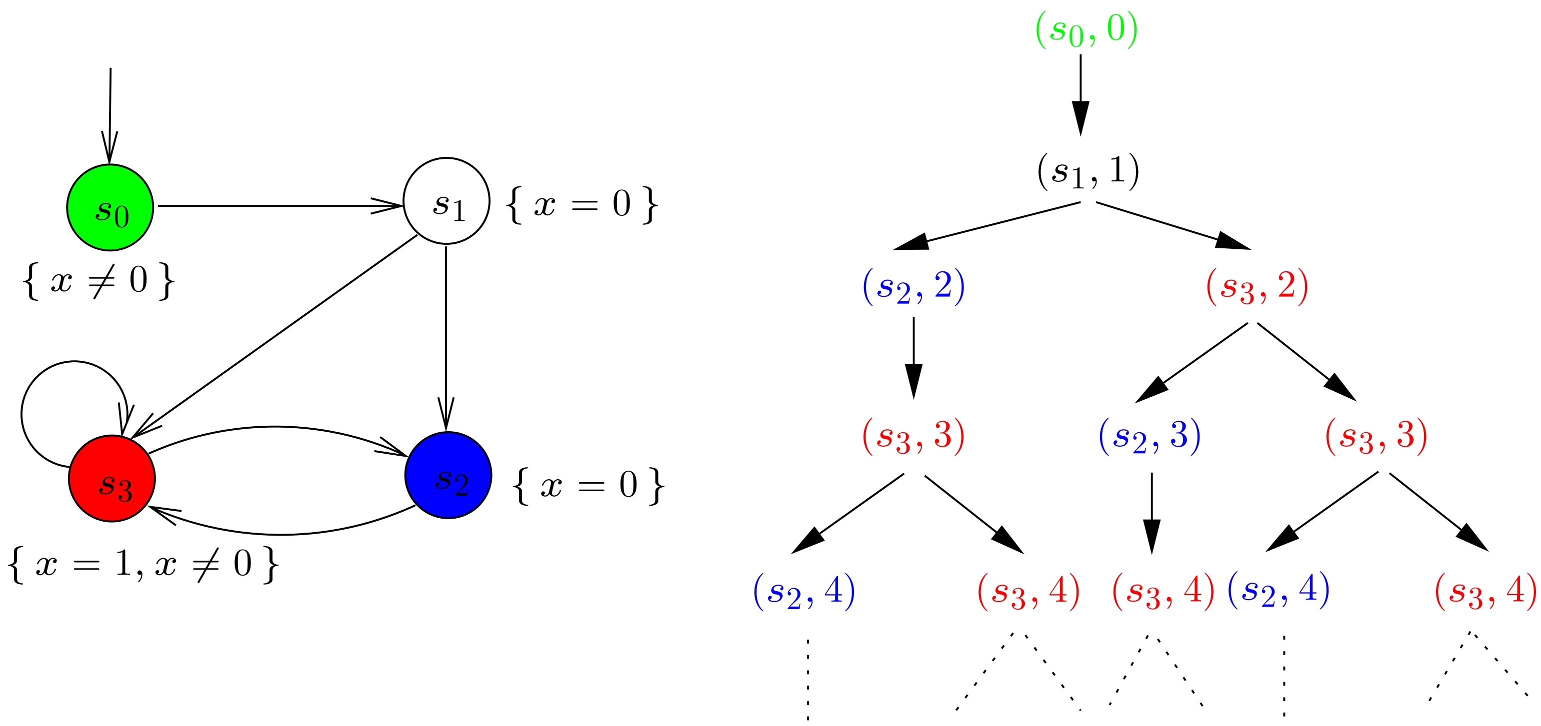
- Checking $\exists\varphi$ in LTL can be done using $\forall\neg\varphi$

  - . . . but this does not work for nested formulas such as $\forall\Box\exists\Diamond a$

Linear versus branching temporal logic

- Semantics is based on a branching notion of time
  - an infinite tree of states obtained by unfolding transition system
  - one “time instant” may have several possible successor “time instants”
- Incomparable expressiveness
  - there are properties that can be expressed in LTL, but not in CTL
  - there are properties that can be expressed in most branching, but not in LTL
- Distinct model-checking algorithms, and their time complexities
- Distinct treatment of fairness assumptions
- Distinct equivalences (pre-orders) on transition systems
  - that correspond to logical equivalence in LTL and branching temporal logics

Transition systems and trees



"behavior" in a state s	path-based: *trace*(s)	state-based: computation tree of s
temporal logic	LTL: path formulas φ $s \models \varphi$ iff $\forall \pi \in \mathit{Paths}(s).\ \pi \models \varphi$	CTL: state formulas existential path quantifi cation $\exists\varphi$ universal path quantifi cation: $\forall\varphi$
complexity of the model checking problems	PSPACE–complete $\mathcal{O}\left(\lvert TS\rvert \cdot 2^{\lvert\varphi\rvert}\right)$	PTIME $\mathcal{O}\left(\lvert TS\rvert \cdot \lvert\Phi\rvert\right)$
implementation-relation	trace inclusion and the like (proof is PSPACE-complete)	simulation and bisimulation (proof in polynomial time)
fairness	no special techniques	special techniques needed

Computation tree logic

modal logic over infinite trees [Clarke & Emerson 1981]

- Statements over states
  - $a \in AP$ — atomic proposition
  - $\neg \Phi$ and $\Phi \wedge \Psi$ — negation and conjunction
  - $\exists \varphi$ — there *exists* a path fulfilling φ
  - $\forall \varphi$ — *all* paths fulfill φ
- Statements over paths
  - $\bigcirc \Phi$ — the next state fulfills Φ
  - $\Phi \, \mathsf{U} \, \Psi$ — Φ holds until a Ψ-state is reached

$\Rightarrow$ note that $\bigcirc$ and U *alternate* with $\forall$ and $\exists$

Derived operators

potentially Φ: $\exists\Diamond\Phi = \exists(\text{true}\,\mathsf{U}\,\Phi)$

inevitably Φ: $\forall\Diamond\Phi = \forall(\text{true}\,\mathsf{U}\,\Phi)$

potentially always Φ: $\exists\Box\Phi := \neg\forall\Diamond\neg\Phi$

invariantly Φ: $\forall\Box\Phi = \neg\exists\Diamond\neg\Phi$

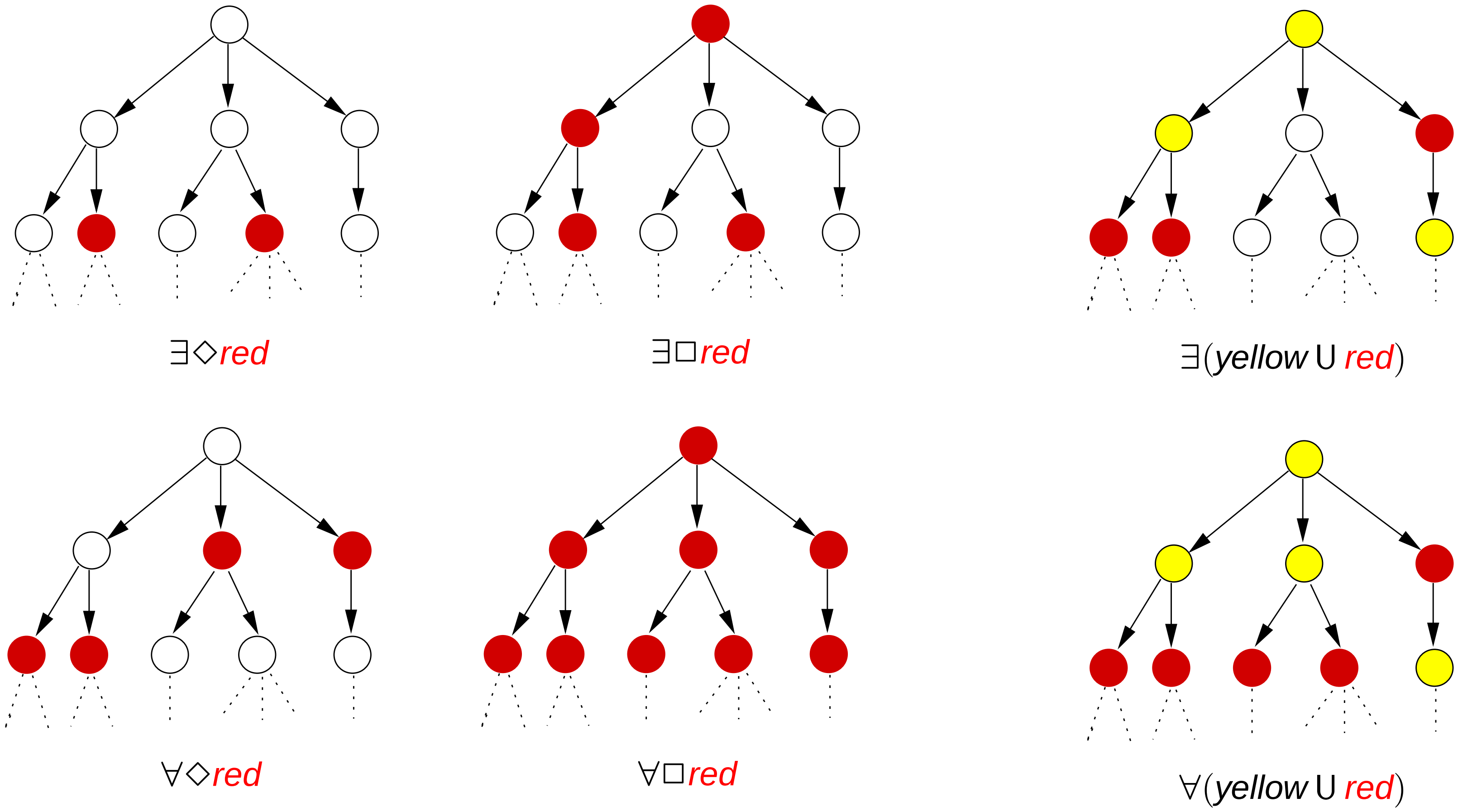
weak until:

$$\exists(\Phi\,\mathsf{W}\,\Psi) = \neg\forall\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$$

$$\forall(\Phi\,\mathsf{W}\,\Psi) = \neg\exists\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$$

the boolean connectives are derived as usual

Visualization of semantics



Semantics of CTL state-formulas

Defined by a relation $\models$ such that

$s \models \Phi$ if and only if formula Φ holds in state s

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg\,\Phi & \text{iff} & \neg\,(s \models \Phi) \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \wedge (s \models \Psi) \\
s \models \exists\varphi & \text{iff} & \pi \models \varphi \text{ for \textit{some} path } \pi \text{ that starts in } s \\
s \models \forall\varphi & \text{iff} & \pi \models \varphi \text{ for \textit{all} paths } \pi \text{ that start in } s
\end{array}
$$

Semantics of CTL path-formulas

Define a relation $\models$ such that

$\pi \models \varphi$ if and only if path π satisfies φ

$$\pi \models \bigcirc \Phi \qquad \text{iff } \pi[1] \models \Phi$$

$$\pi \models \Phi \,\mathsf{U}\, \Psi \qquad \text{iff } (\exists\, j \geqslant 0.\, \pi[j] \models \Psi \;\wedge\; (\forall\, 0 \leqslant k < j.\, \pi[k] \models \Phi))$$

where $\pi[i]$ denotes the state s_i in the path π

Transition system semantics

- For CTL-state-formula Φ, the *satisfaction set* *Sat*(Φ) is defined by:

$$\mathit{Sat}(\Phi) \; = \; \{ \, s \in S \mid s \models \Phi \, \}$$

- *TS* satisfies CTL-formula Φ iff Φ holds in all its initial states:

$$\mathit{TS} \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I. \, s_0 \models \Phi$$

- Point of attention: *TS* $\not\models \Phi$ and *TS* $\not\models \neg\Phi$ is possible!
  - because of several initial states, e.g. $s_0 \models \exists\Box\Phi$ and $s_0' \not\models \exists\Box\Phi$

CTL equivalence

CTL-formulas Φ and Ψ (over *AP*) are *equivalent*, denoted $\Phi \equiv \Psi$
if and only if $Sat(\Phi) = Sat(\Psi)$ for all transition systems *TS* over *AP*

$$\Phi \equiv \Psi \quad \text{iff} \quad (TS \models \Phi \quad \text{if and only if} \quad TS \models \Psi)$$

Expansion laws

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

In CTL:

$$
\begin{aligned}
\forall(\Phi \, \mathsf{U} \, \Psi) &\equiv \Psi \vee (\Phi \wedge \forall \bigcirc \forall(\Phi \, \mathsf{U} \, \Psi)) \\
\forall\Diamond\Phi &\equiv \Phi \vee \forall \bigcirc \forall\Diamond\Phi \\
\forall\Box\Phi &\equiv \Phi \wedge \forall \bigcirc \forall\Box\Phi
\end{aligned}
$$

$$
\begin{aligned}
\exists(\Phi \, \mathsf{U} \, \Psi) &\equiv \Psi \vee (\Phi \wedge \exists \bigcirc \exists(\Phi \, \mathsf{U} \, \Psi)) \\
\exists\Diamond\Phi &\equiv \Phi \vee \exists \bigcirc \exists\Diamond\Phi \\
\exists\Box\Phi &\equiv \Phi \wedge \exists \bigcirc \exists\Box\Phi
\end{aligned}
$$

Distributive laws

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 $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$

Equivalence of LTL and CTL formulas

- CTL-formula Φ and LTL-formula φ (both over *AP*) are *equivalent*, denoted $\Phi \equiv \varphi$, if for any transition system *TS* over *AP*:

$$TS \models \Phi \quad \text{if and only if} \quad TS \models \varphi$$

- Let Φ be a CTL-formula, and φ the LTL-formula that is obtained by eliminating all path quantifiers in Φ. Then:

$\Phi \equiv \varphi$ or there does not exist any LTL-formula that is equivalent to Φ

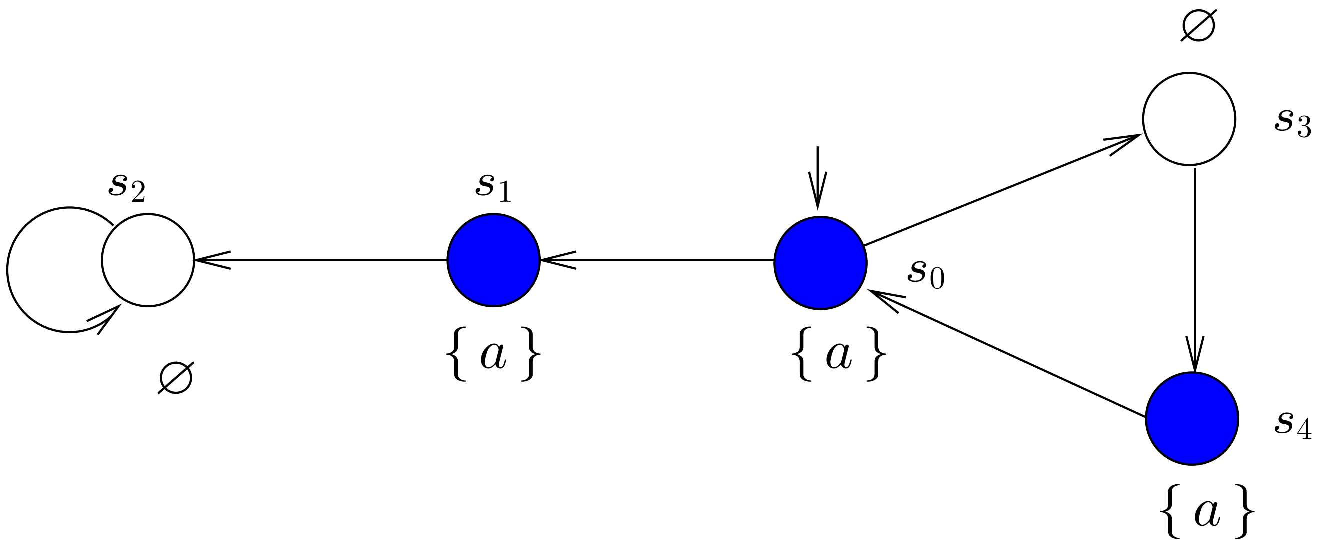
LTL and CTL are incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box a$
  - $\Diamond (a \;\wedge\; \bigcirc a)$
- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall \Diamond \forall \Box a$
  - $\forall \Diamond (a \wedge \forall \bigcirc a)$
  - $\forall \Box \exists \Diamond a$

$\Rightarrow$ Cannot be expressed = there does not exist an equivalent formula

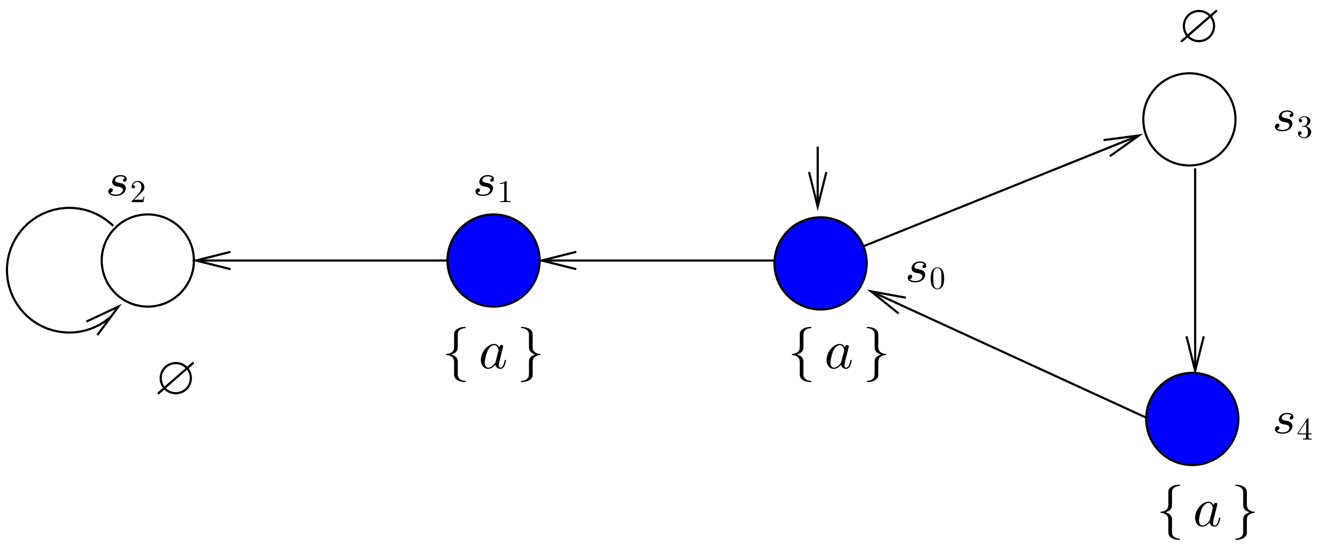
Comparing LTL and CTL (1)

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



Comparing LTL and CTL (1)

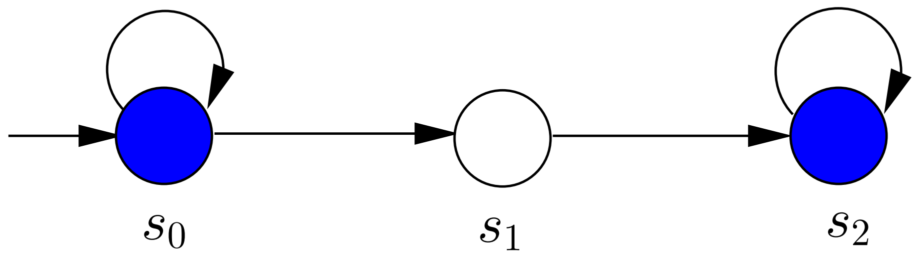
$\Diamond(a \wedge \bigcirc a)$ is not equivalent to $\forall\Diamond(a \wedge \forall\bigcirc a)$



$s_0 \models \Diamond(a \wedge \bigcirc a)$ but $\underbrace{s_0 \not\models \forall\Diamond(a \wedge \forall\bigcirc a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$

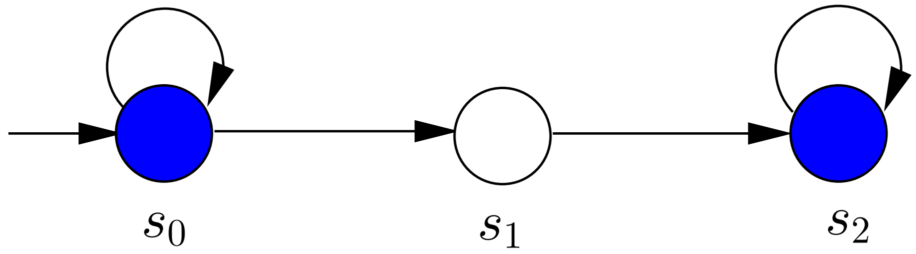
Comparing LTL and CTL (2)

$\forall\Diamond\forall\Box a$ is not equivalent to $\Diamond\Box a$



Comparing LTL and CTL (2)

$\forall\Diamond\forall\Box a$ is not equivalent to $\Diamond\Box a$



$s_0 \models \Diamond\Box a$ but $\underbrace{s_0 \not\models \forall\Diamond\forall\Box a}_{\text{path } s_0^\omega \text{ violates it}}$

Existential normal form (ENF)

The set of CTL formulas in *existential normal form* (ENF) is given by:

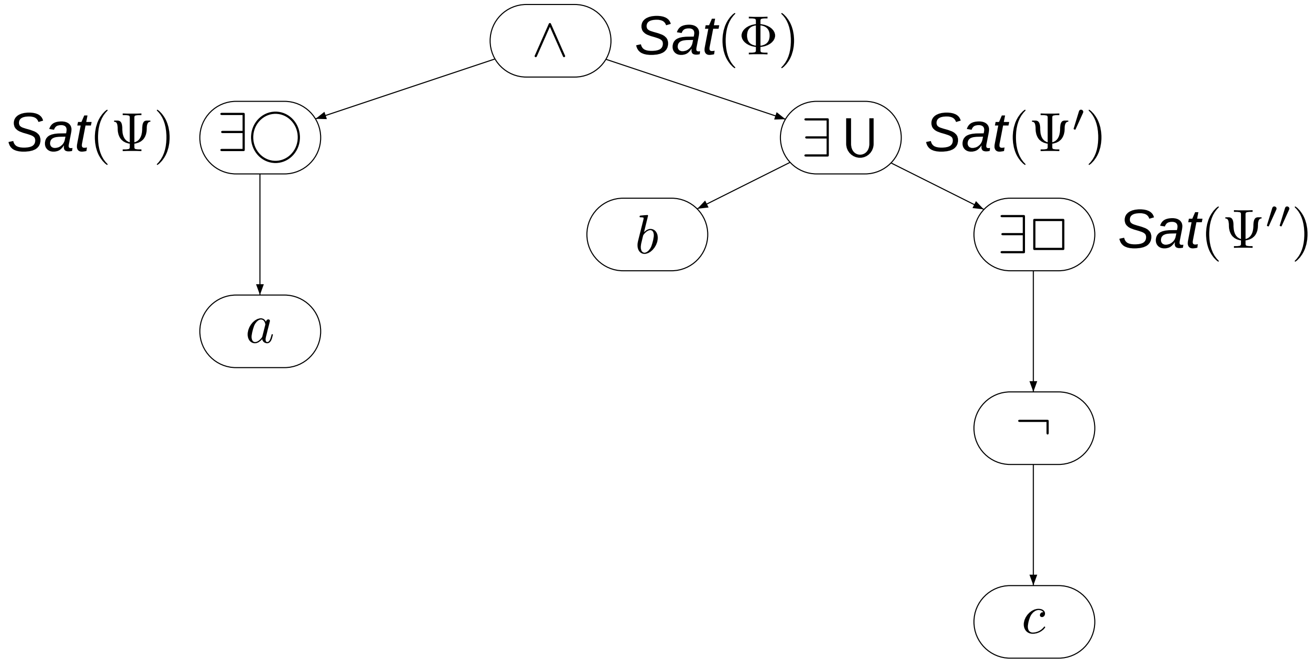
$$\Phi \quad ::= \quad \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\bigcirc\Phi \;\Big|\; \exists(\Phi_1 \,\mathsf{U}\, \Phi_2) \;\Big|\; \exists\Box\Phi$$

For each CTL formula, there exists an equivalent CTL formula in ENF

Model checking CTL

- Convert the formula Φ' into an equivalent Φ in ENF
- How to check whether state *TS* satisfies Φ?
  - compute *recursively* the set $\mathit{Sat}(\Phi)$ of states that satisfy Φ
  - check whether all initial states belong to $\mathit{Sat}(\Phi)$
- Recursive bottom-up computation:
  - consider the parse-tree of Φ
  - start to compute $\mathit{Sat}(a)$, for all leafs in the tree
  - then go one level up in the tree and check the formula of these nodes
  - then go one level up and check the formula of these nodes
  - and so on....... until the root of the tree (i.e., Φ) is checked

Example



$$\Phi = \underbrace{\exists\bigcirc a}_{\Psi} \wedge \underbrace{\exists(b\,\mathsf{U}\,\underbrace{\exists\Box\neg c}_{\Psi''})}_{\Psi'} .$$

Characterization of *Sat* (1)

For all CTL formulas Φ, Ψ over *AP* it holds:

$$
\begin{aligned}
\mathit{Sat}(\text{true}) &= S \\
\mathit{Sat}(a) &= \{\, s \in S \mid a \in L(s) \,\}, \text{ for any } a \in \mathit{AP} \\
\mathit{Sat}(\Phi \wedge \Psi) &= \mathit{Sat}(\Phi) \cap \mathit{Sat}(\Psi) \\
\mathit{Sat}(\neg\Phi) &= S \setminus \mathit{Sat}(\Phi) \\
\mathit{Sat}(\exists\bigcirc\Phi) &= \{\, s \in S \mid \mathit{Post}(s) \cap \mathit{Sat}(\Phi) \neq \varnothing \,\}
\end{aligned}
$$

where $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ is a transition system without terminal states

Characterization of *Sat* (2)

For all CTL formulas Φ, Ψ over *AP* it holds:

- *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest subset T of S, such that:

  (1) *Sat*$(\Psi) \subseteq T$ and

  (2) $s \in Sat(\Phi)$ and *Post*$(s) \cap T \neq \varnothing$ implies $s \in T$

- *Sat*$(\exists\Box\Phi)$ is the largest subset T of S, such that:

  (3) $T \subseteq$ *Sat*(Φ) and

  (4) $s \in T$ implies *Post*$(s) \cap T \neq \varnothing$

where *TS* $= (S,$ *Act*$, \rightarrow, I,$ *AP*$, L)$ is a transition system without terminal states

Computation of *Sat*

switch(Φ):

a	:	**return** $\{ s \in S \mid a \in L(s) \}$;
$\ldots$	:	$\ldots\ldots$
$\exists \bigcirc \Psi$	:	**return** $\{ s \in S \mid \mathit{Post}(s) \cap \mathit{Sat}(\Psi) \neq \varnothing \}$;
$\exists (\Phi_1 \, \mathsf{U} \, \Phi_2)$	:	$T := \mathit{Sat}(\Phi_2)$; (* compute the smallest fixed point *) **while** $\mathit{Sat}(\Phi_1) \setminus T \cap \mathit{Pre}(T) \neq \varnothing$ **do** **let** $s \in \mathit{Sat}(\Phi_1) \setminus T \cap \mathit{Pre}(T)$; $T := T \cup \{ s \}$; **od**; **return** T;
$\exists \Box \Psi$	:	$T := \mathit{Sat}(\Psi)$; (* compute the greatest fixed point *) **while** $\exists s \in T.\, \mathit{Post}(s) \cap T = \varnothing$ **do** **let** $s \in \{ s \in T \mid \mathit{Post}(s) \cap T = \varnothing \}$; $T := T \setminus \{ s \}$; **od**; **return** T;

end switch

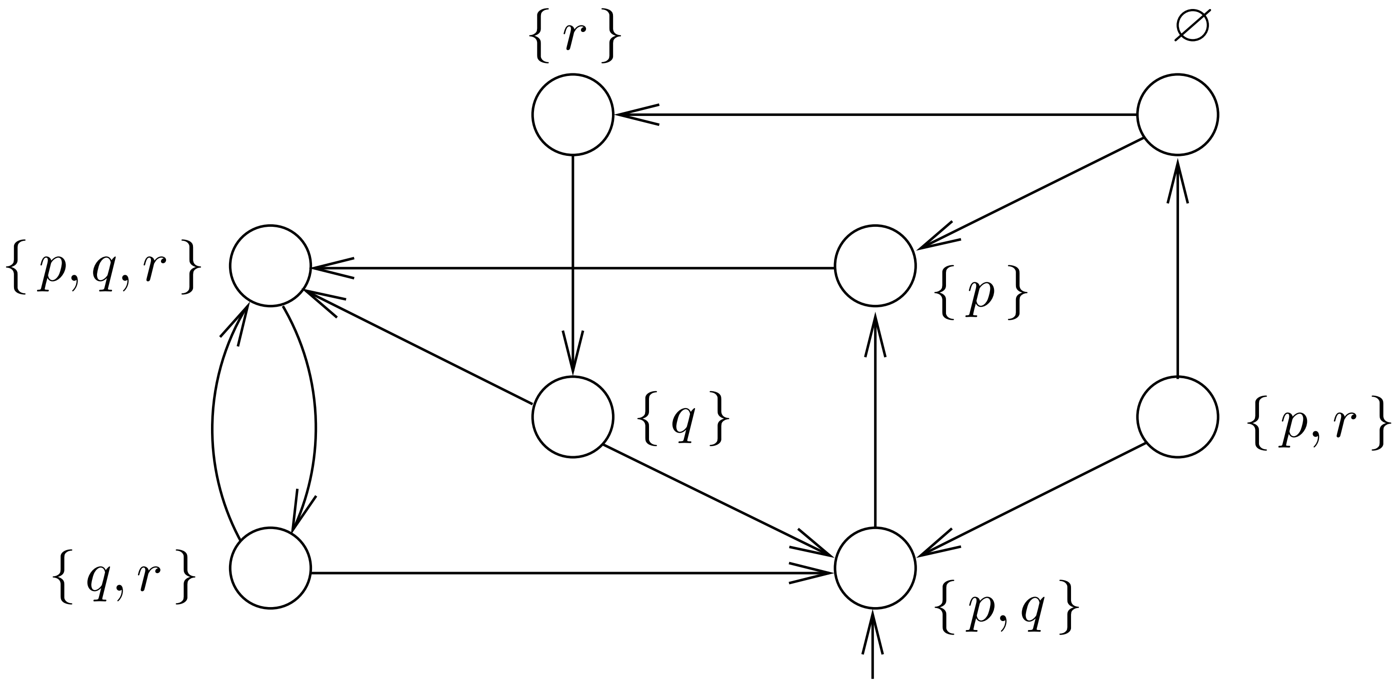
Computing $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$

Computing *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$

Input: finite transition system *TS* with state-set S and CTL-formula $\exists(\Phi \,\mathsf{U}\, \Psi)$
Output: *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$

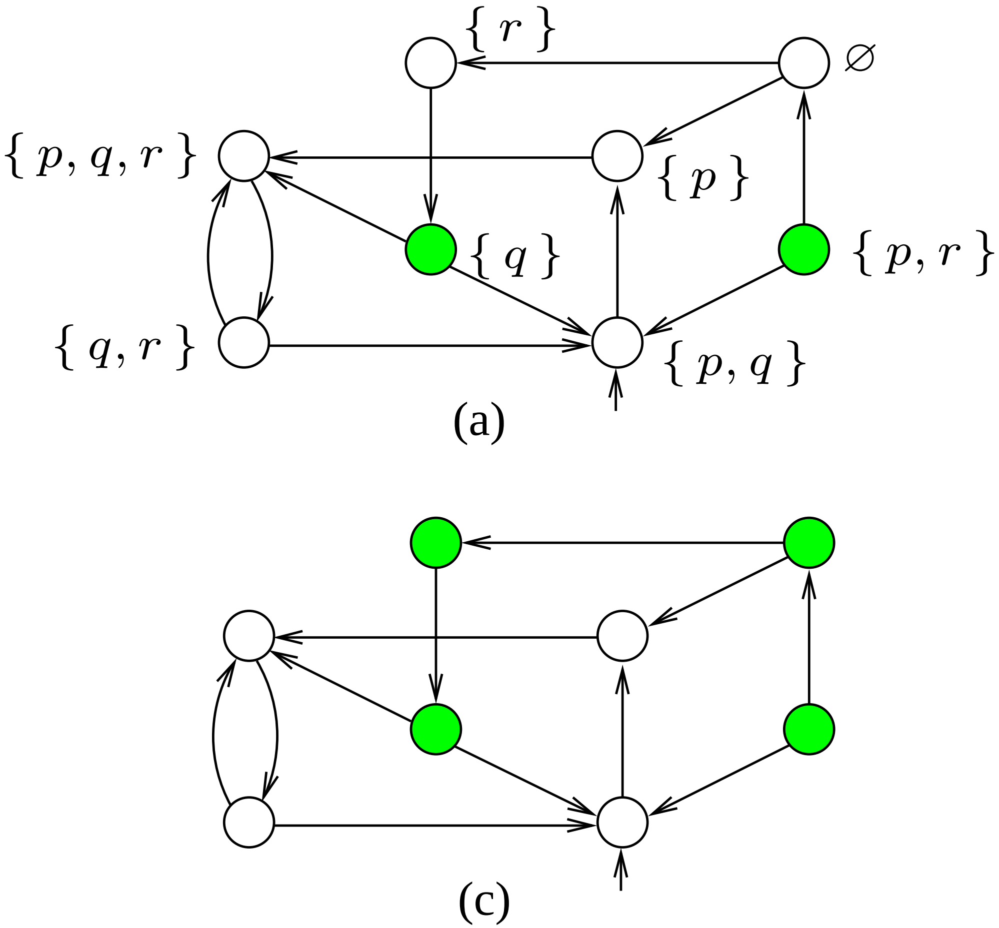
$E := $ *Sat*(Ψ); (* E administers the states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
$T := E$; (* T contains the already visited states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
while $E \neq \varnothing$ **do**
  let $s' \in E$;
  $E := E \setminus \{ s' \}$;
  for all $s \in$ *Pre*(s') **do**
    if $s \in$ *Sat*$(\Phi) \setminus T$ **then** $E := E \cup \{ s \}; T := T \cup \{ s \}$; **fi**
  od
od
return T

Example



let's check the CTL-formula $\exists\Diamond((p = r) \wedge (p \neq q))$

The computation in snapshots



Computing *Sat*$(\exists\Box\Phi)$

$E := S \setminus$ *Sat*(Φ); (* E contains any not visited s' with $s' \not\models \exists\Box\Phi$ *)

$T :=$ *Sat*(Φ); (* T contains any s for which $s \models \exists\Box\Phi$ has not yet been disproven *)

for all $s \in$ *Sat*(Φ) **do** $c[s] := |$ *Post*$(s)\,|$; **od** (* initialize array c *)

while $E \neq \varnothing$ **do**
(* loop invariant: $c[s] = |$ *Post*$(s) \cap (T \cup E)\,|$ *)
  let $s' \in E$; (* $s' \not\models \Phi$ *)
  $E := E \setminus \{ s' \}$; (* s' has been considered *)
  for all $s \in$ *Pre*(s') **do**
    if $s \in T$ **then**
      $c[s] := c[s] - 1$; (* update counter $c[s]$ for predecessor s of s' *)
      if $c[s] = 0$ **then**
        $T := T \setminus \{ s \}$; $E := E \cup \{ s \}$; (* s does not have any successor in T *)
      fi
    fi
  od
od
return T

Alternative algorithm

1. Consider only state s if $s \models \Phi$, otherwise *eliminate* s
   - change *TS* into *TS*$[\Phi] = (S', \mathit{Act}, \rightarrow', I', \mathit{AP}, L')$ with $S' = \mathit{Sat}(\Phi)$,
   - $\rightarrow' = \rightarrow \cap (S' \times \mathit{Act} \times S')$, $I' = I \cap S'$, and $L'(s) = L(s)$ for $s \in S'$

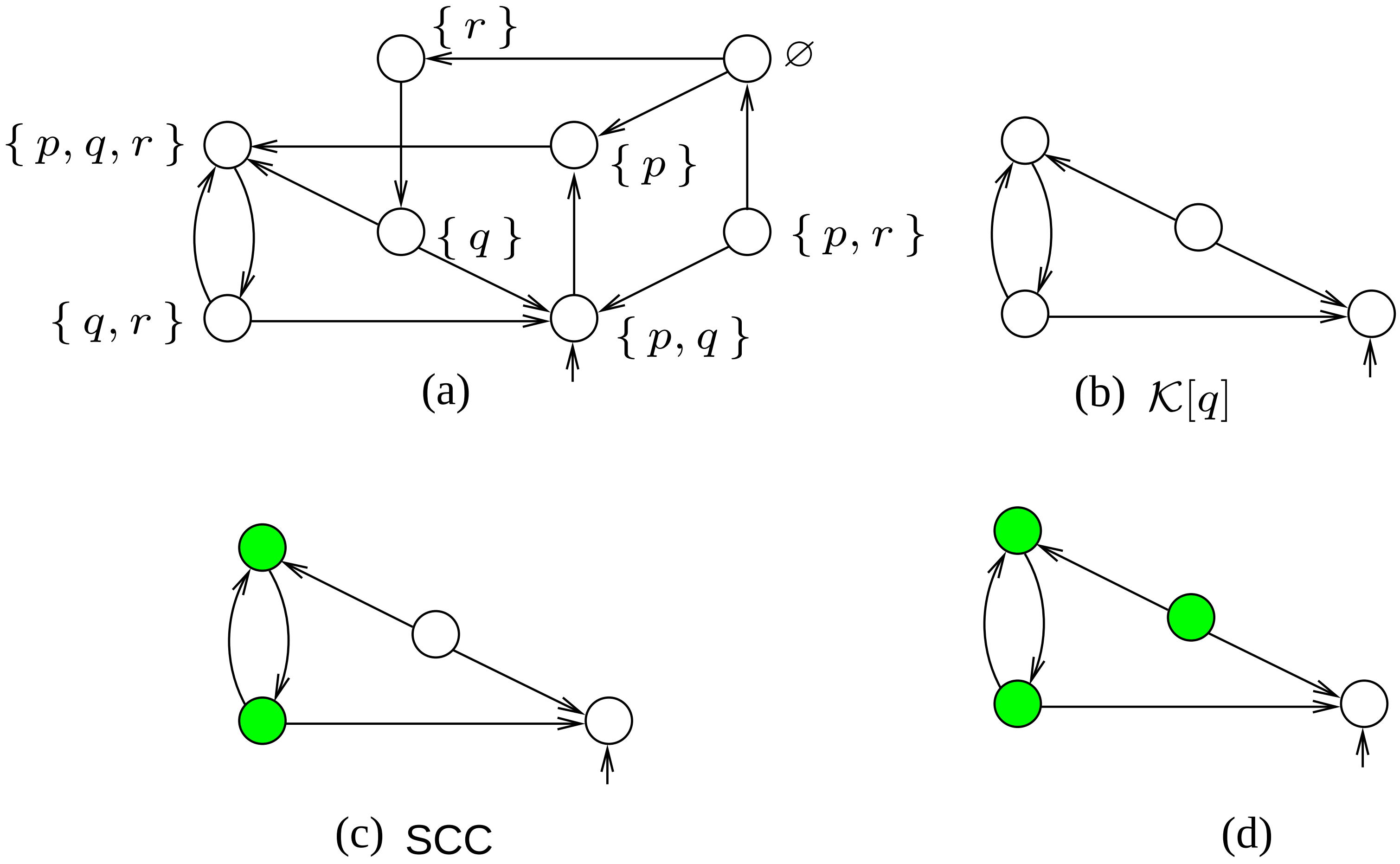
   $\Rightarrow$ all removed states will not satisfy $\exists \Box \Phi$, and thus can be safely removed

2. Determine all *non-trivial strongly connected components* in *TS*$[\Phi]$
   - non-trivial SCC = maximal, connected subgraph with at least one transition

   $\Rightarrow$ any state in such SCC satisfies $\exists \Box \Phi$

3. $s \models \exists \Box \Phi$ is equivalent to “some *SCC is reachable* from s”
   - this search can be done in a backward manner

Example



Time complexity

For transition system *TS* with N states and K transitions,

and CTL formula Φ, the CTL model-checking problem *TS* $\models \Phi$

can be determined in time $\mathcal{O}(|\,\Phi\,| \cdot (N + M))$

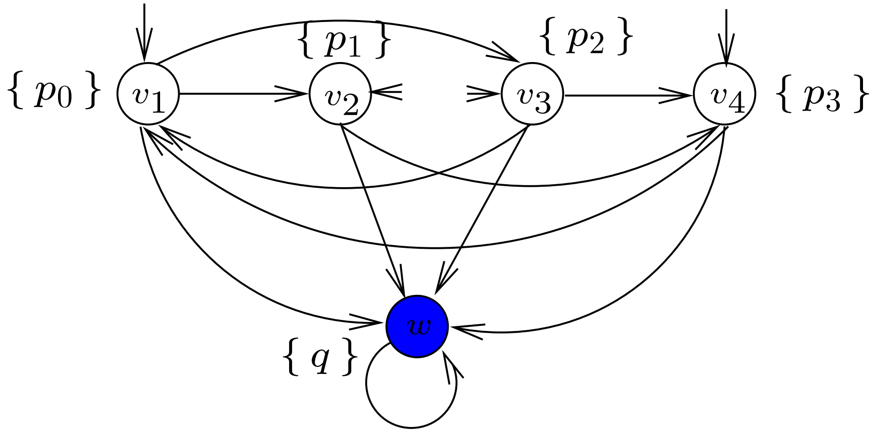
this applies to both algorithm for existential until-formulas

Model-checking LTL versus CTL

- Let *TS* be a transition system with N states and M transitions
- Model-checking LTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot 2^{|\,\Phi\,|})$
  - linear in the state space of the system model
  - exponential in the length of the formula
- Model-checking CTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot |\,\Phi\,|)$
  - linear in the state space of the system model and the formula
- Is model-checking CTL more efficient? No!

Model-checking LTL versus CTL

⇒ LTL-formulae can be *exponentially shorter* than their equivalent in CTL



- Existence of Hamiltonian path in LTL: $\neg \left((\Diamond p_0 \wedge \ldots \wedge \Diamond p_3) \; \wedge \; \bigcirc^4 q \right)$

- In CTL, all possible (= 4!) routes need to be encoded