CTL Model Checking

Lecture #4 of Principles of Model Checking

Joost-Pieter Katoen

Software Modeling and Verification Group

affiliated to University of Twente, Formal Methods and Tools

University of Twente, September 29, 2009

Content of this lecture

- Computation tree logic
  - syntax, semantics, equational laws
- CTL model checking
  - recursive descent, backward reachability, complexity
- Comparing LTL and CTL
  - what can be expressed in CTL? what in LTL?, efficiency
- Fairness
  - fair CTL semantics, model checking

Content of this lecture

$\Rightarrow$ Computation tree logic

- syntax, semantics, equational laws

- CTL model checking
  - recursive descent, backward reachability, complexity
- Comparing LTL and CTL
  - what can be expressed in CTL? what in LTL?, efficiency
- Fairness
  - fair CTL semantics, model checking

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

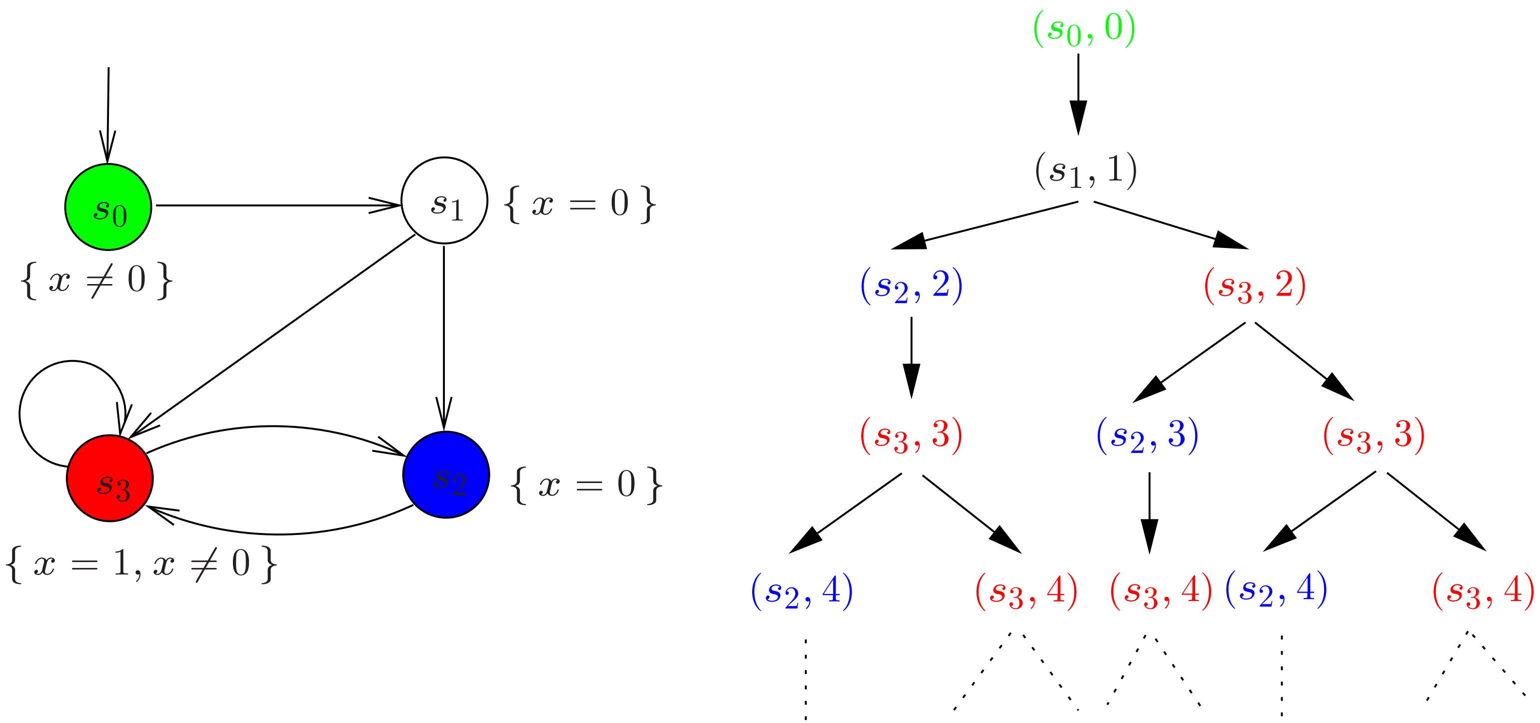
- Checking $\exists\varphi$ in LTL can be done using $\forall\neg\varphi$

  – ... 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 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 quantification $\exists\varphi$ universal path quantification: $\forall\varphi$
complexity of the model checking problems	PSPACE–complete $\mathcal{O}\left(\lvert \mathit{TS} \rvert \cdot 2^{\lvert\varphi\rvert}\right)$	PTIME $\mathcal{O}\left(\lvert \mathit{TS} \rvert \cdot \lvert\Phi\rvert\right)$
implementation- relation	trace inclusion and the like (proof is PSPACE-complete)	simulation and bisimulation (proof in polynomial time)

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

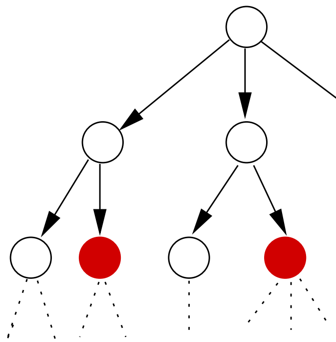
$\Rightarrow$ note that $\bigcirc$ and U *alternate* with $\forall$ and $\exists$

Derived operators

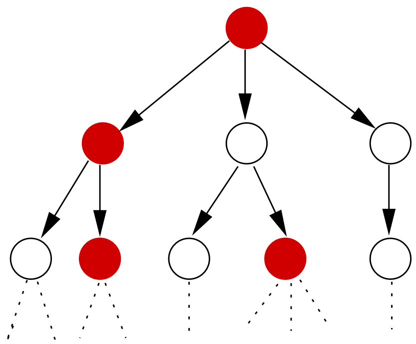
$$
\begin{array}{llll}
\text{potentially } \Phi: & \exists\Diamond\Phi & = & \exists(\mathsf{true}\,\mathsf{U}\,\Phi) \\
\text{inevitably } \Phi: & \forall\Diamond\Phi & = & \forall(\mathsf{true}\,\mathsf{U}\,\Phi) \\
\\
\text{potentially always } \Phi: & \exists\Box\Phi & := & \neg\forall\Diamond\neg\Phi \\
\text{invariantly } \Phi: & \forall\Box\Phi & = & \neg\exists\Diamond\neg\Phi \\
\\
\text{weak until:} & \exists(\Phi\,\mathsf{W}\,\Psi) & = & \neg\forall\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big) \\
 & \forall(\Phi\,\mathsf{W}\,\Psi) & = & \neg\exists\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)
\end{array}
$$

the boolean connectives are derived as usual

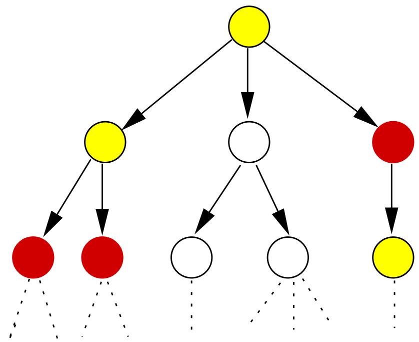
Visualization of semantics



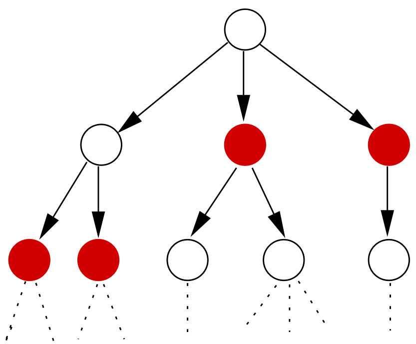
$\exists \Diamond \mathit{red}$



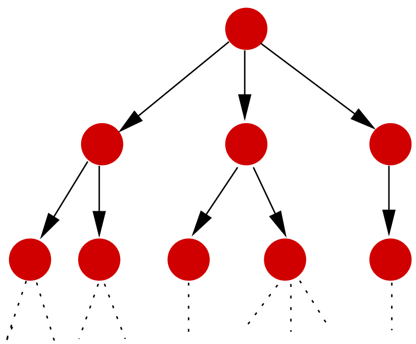
$\exists \Box \mathit{red}$



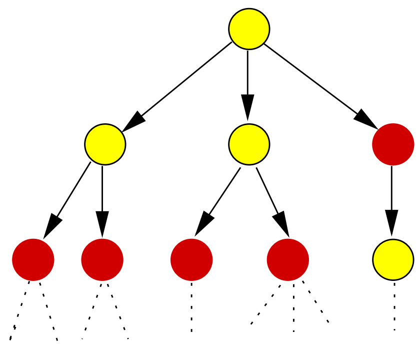
$\exists (\mathit{yellow} \,\mathsf{U}\, \mathit{red})$



$\forall \Diamond \mathit{red}$



$\forall \Box \mathit{red}$



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

Semantics of CTL state-formulas

Defined by a relation $\models$ such that

$\boxed{s \models \Phi \text{ if and only if formula } \Phi \text{ holds in state } s}$

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

Semantics of CTL path-formulas

Define a relation $\models$ such that

$\boxed{\pi \models \varphi \text{ if and only if path } \pi \text{ satisfies } \varphi}$

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

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

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

Transition system semantics

- For CTL-state-formula Φ, the *satisfaction set* $\mathit{Sat}(\Phi)$ 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: $\mathit{TS} \not\models \Phi$ and $\mathit{TS} \not\models \neg\Phi$ 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 $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$

Content of this lecture

- Computation tree logic
  - syntax, semantics, equational laws

⇒ CTL model checking
  - recursive descent, backward reachability, complexity

- Comparing LTL and CTL
  - what can be expressed in CTL? what in LTL?, efficiency

- Fairness
  - fair CTL semantics, model checking

Existential normal form (ENF)

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

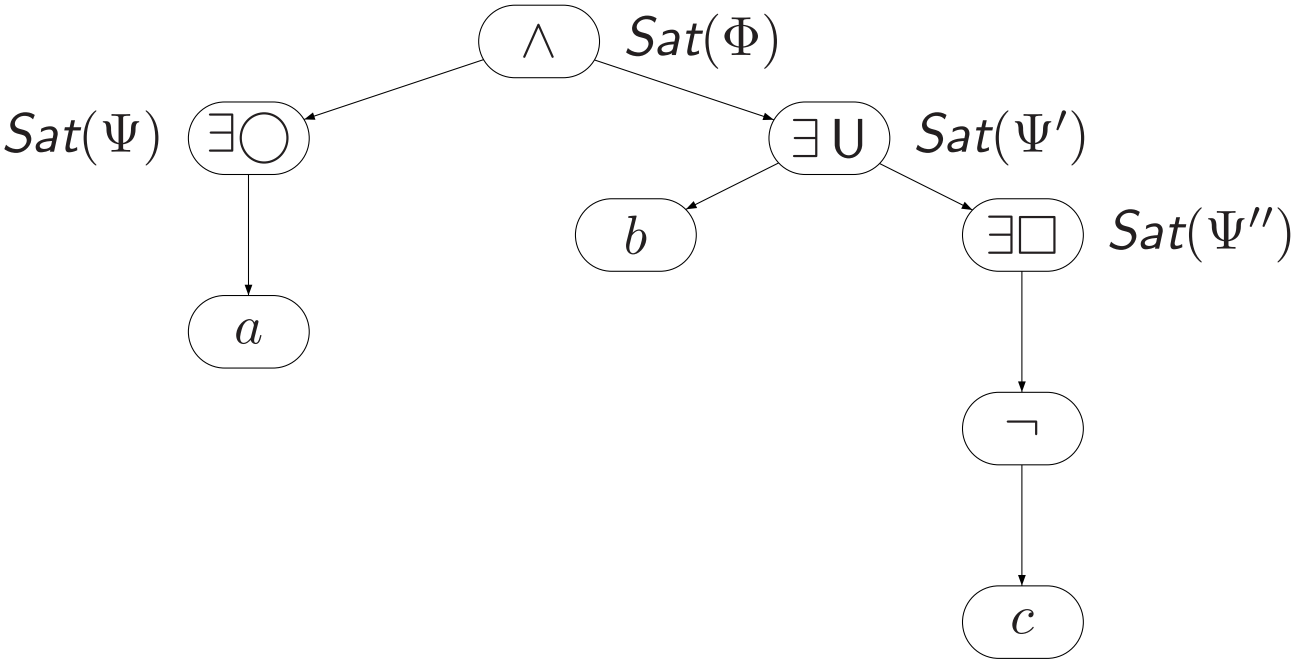
$$\Phi \quad ::= \quad \mathsf{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\bigcirc\Phi \;\Big|\; \exists(\Phi_1 \,\mathsf{U}\, \Phi_2) \;\Big|\; \exists\Box\Phi$$

$$\boxed{\text{For each CTL formula, there exists an equivalent CTL formula in ENF}}$$

Model checking CTL

- Convert the formula Φ' into an equivalent Φ in ENF
- How to check whether *TS* satisfies Φ?
  - compute *recursively* the set $Sat(\Phi)$ of states that satisfy Φ
  - check whether all initial states belong to $Sat(\Phi)$
- Recursive bottom-up computation:
  - consider the parse-tree of Φ
  - start to compute $Sat(a)$, 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



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

Characterization of *Sat* (1)

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

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

where $TS = (S, Act, \longrightarrow, I, AP, L)$ 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) $T \subseteq Sat(\Phi)$ and

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

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

Computation of *Sat*

switch(Φ):

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

end switch

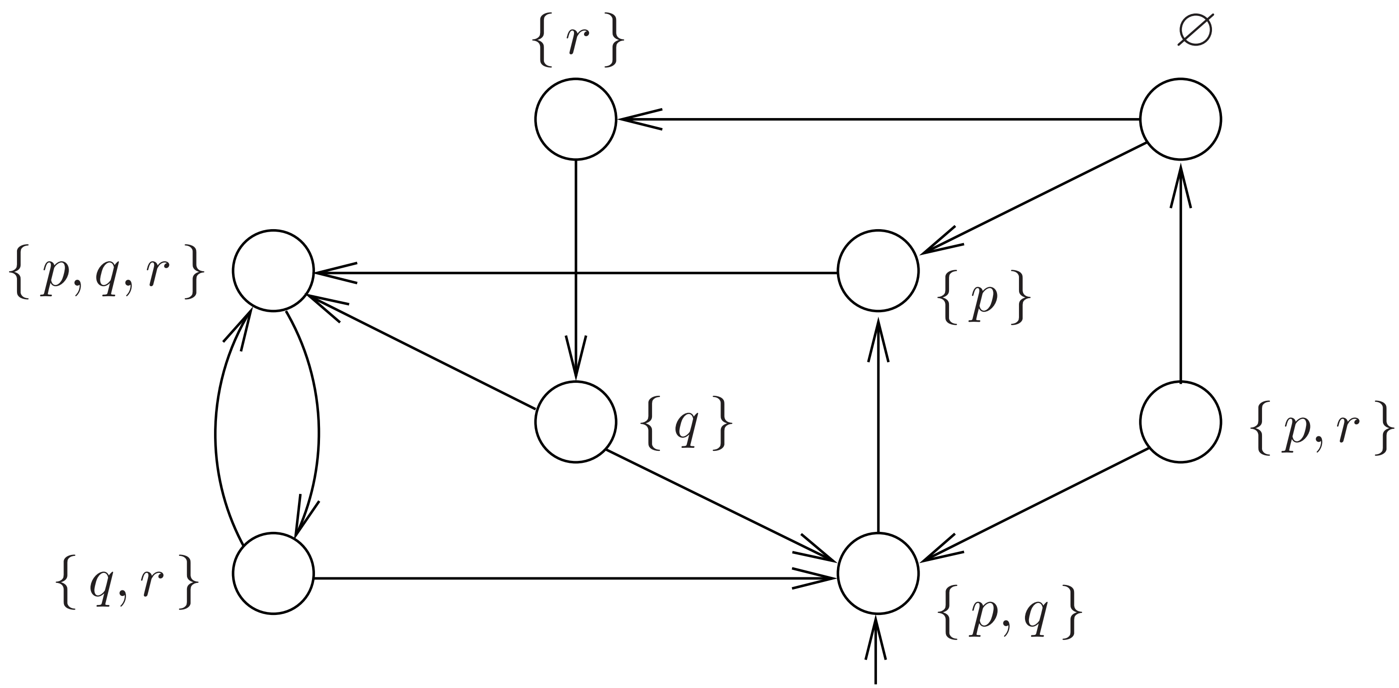
Computing $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$

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

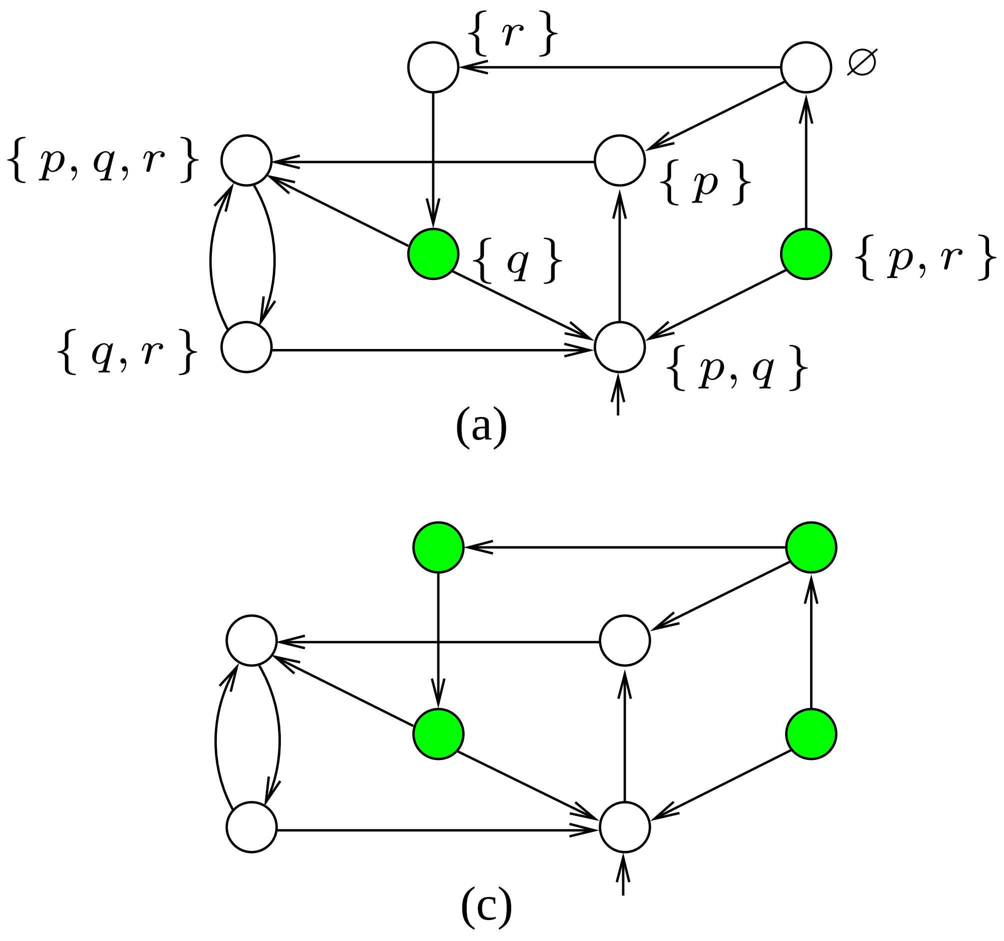
$E := Sat(\Psi)$; (* E administers the states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
$T := E$; (* T contains the already visited states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
while $E \neq \varnothing$ **do**
  let $s' \in E$;
  $E := E \setminus \{ s' \}$;
  for all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

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

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

for all $s \in Sat(\Phi)$ **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 $TS[\Phi] = (S', Act, \rightarrow', I', AP, L')$ with $S' = Sat(\Phi)$,
   - $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $I' = I \cap S'$, 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 $TS[\Phi]$

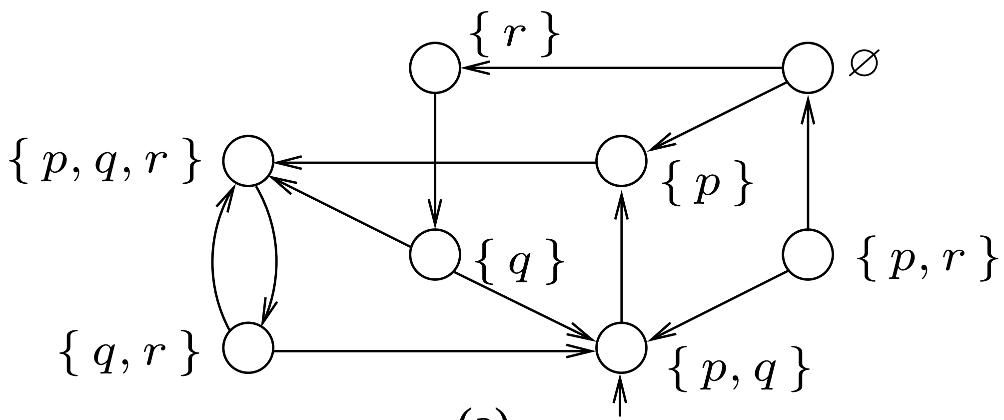
   - non-trivial SCC = maximal, connected subgraph with at least one transition

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

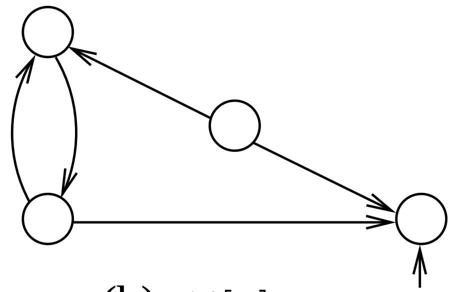
3. $s \models \exists\Box\Phi$ is equivalent to "some *SCC is reachable* from s"

   - this search can be done in a backward manner

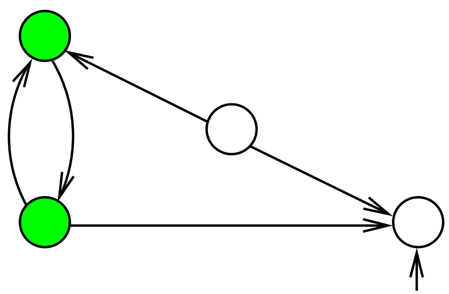
Example



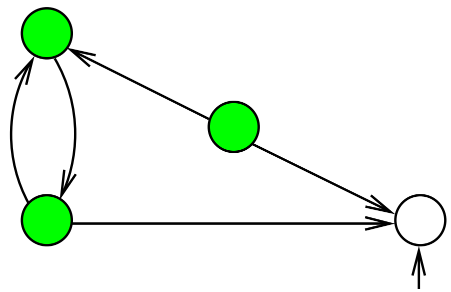
(a)



(b) $\mathcal{K}[q]$



(c) SCC



(d)

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

Content of this lecture

- Computation tree logic
  - syntax, semantics, equational laws
- CTL model checking
  - recursive descent, backward reachability, complexity

$\Rightarrow$ Comparing LTL and CTL
  - what can be expressed in CTL?, what in LTL?, efficiency

- Fairness
  - fair CTL semantics, model checking

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:

$$\boxed{\Phi \;\equiv\; \varphi \text{ or there does not exist any LTL-formula that is equivalent to } \Phi}$$

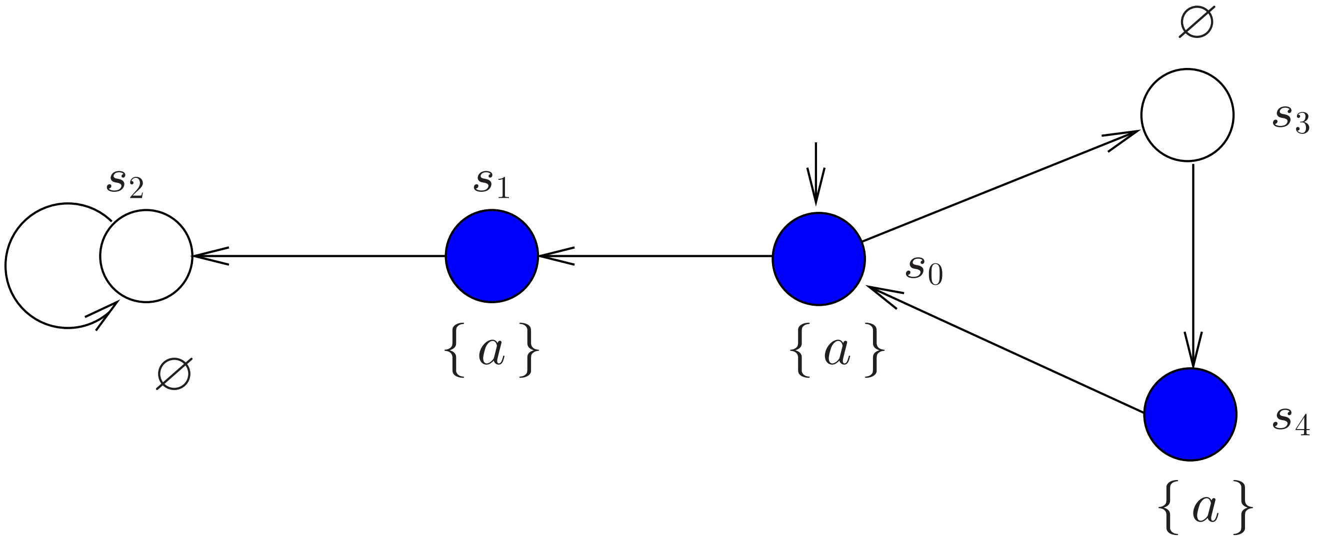
LTL and CTL are incomparable

- 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

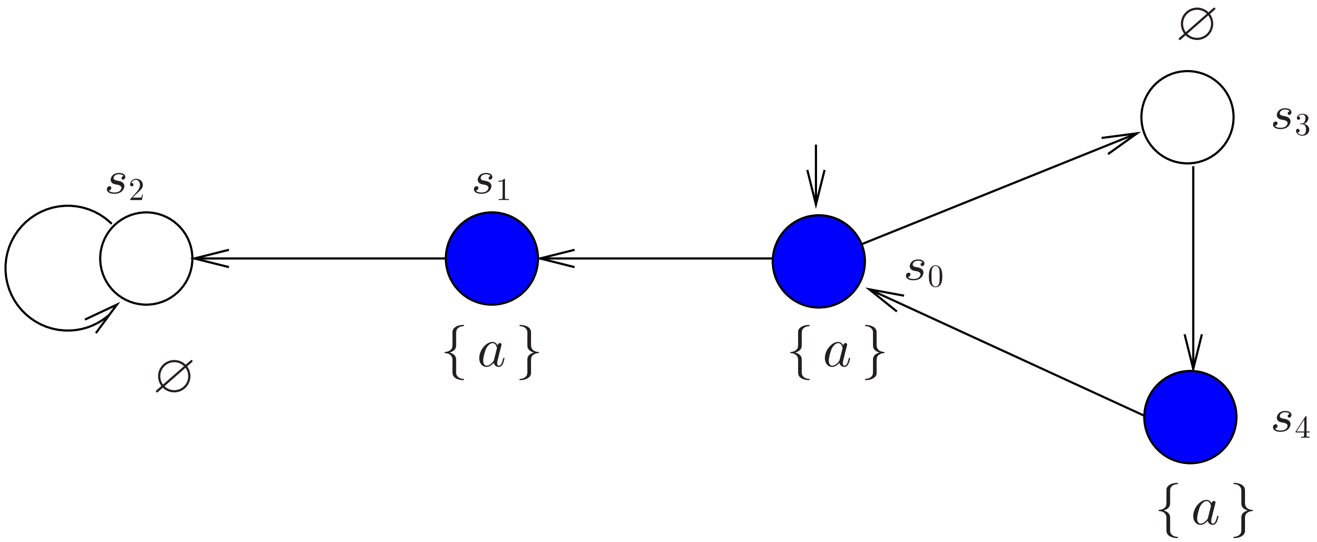
Comparing LTL and CTL (1)

$\Diamond(a \wedge \bigcirc a)$ 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 $\forall\Diamond(a \wedge \forall\bigcirc a)$

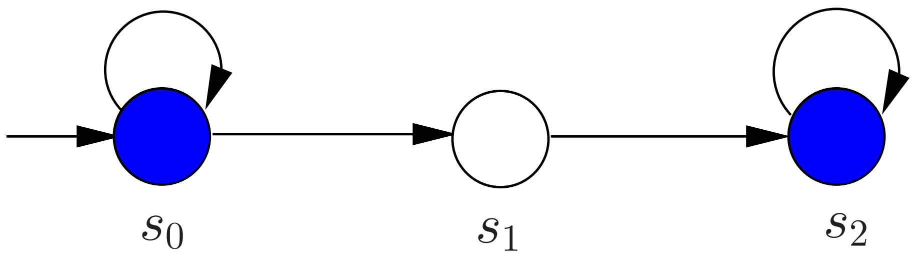


$s_0 \models \Diamond(a \wedge \bigcirc a)$ but $s_0 \not\models \forall\Diamond(a \wedge \forall\bigcirc a)$

since path $s_0\, s_1\, (s_2)^\omega$ violates $\Diamond(a \wedge \forall\bigcirc a)$

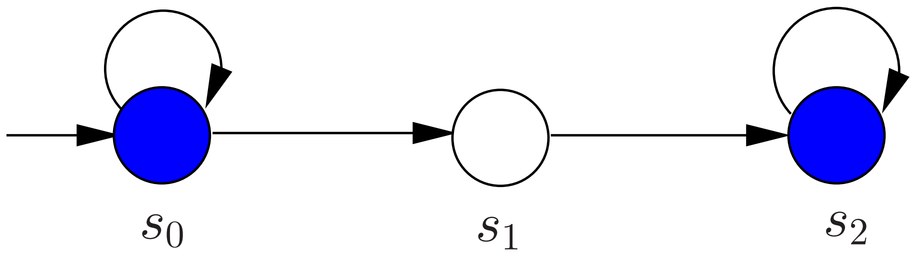
Comparing LTL and CTL (2)

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



Comparing LTL and CTL (2)

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

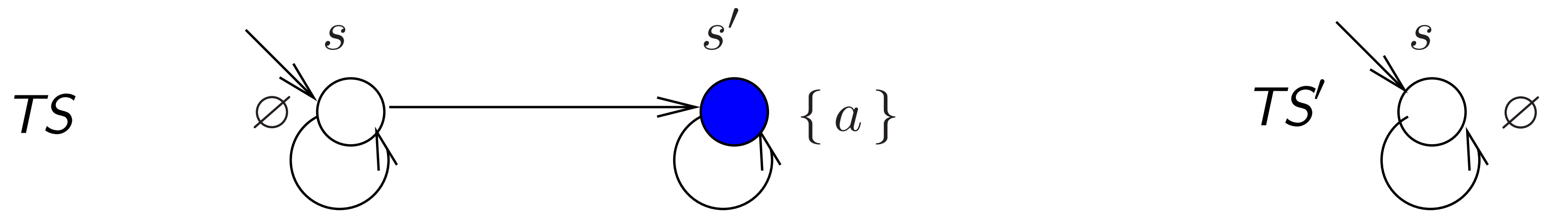


$s_0 \models \Diamond\Box a$ but $s_0 \not\models \forall\Diamond\forall\Box a$

since path s_0^ω violates $\Diamond\forall\Box a$

Comparing LTL and CTL (3)

- No LTL-formula φ is equivalent to $\forall\Box\exists\Diamond a$
- This is shown by contradiction: assume $\varphi \equiv \forall\Box\exists\Diamond a$; let:



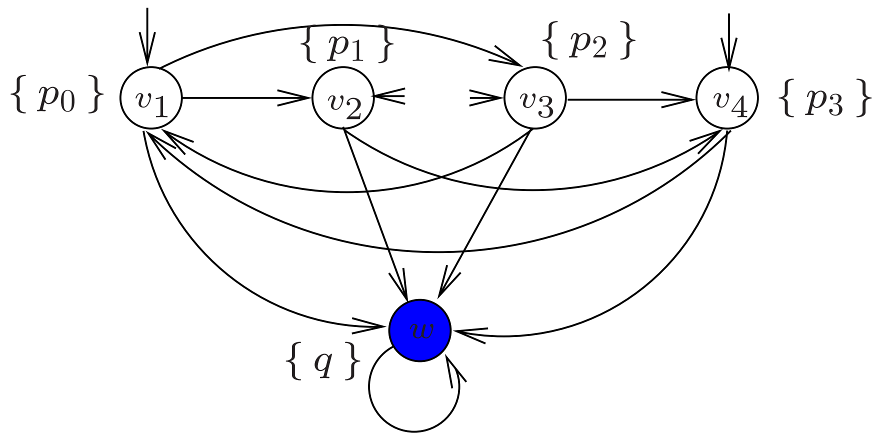
- $TS \models \forall\Box\exists\Diamond a$, and thus—by assumption—$TS \models \varphi$
- $Paths(TS') \subseteq Paths(TS)$, thus $TS' \models \varphi$
- But $TS' \not\models \forall\Box\exists\Diamond a$ as path $s^\omega \not\models \Box\exists\Diamond a$

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

$\Rightarrow$ 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

Content of this lecture

- Computation tree logic
  - syntax, semantics, equational laws
- CTL model checking
  - recursive descent, backward reachability, complexity
- Comparing LTL and CTL
  - what can be expressed in CTL?, what in LTL?, efficiency

$\Rightarrow$ Fairness

- fair CTL semantics, model checking

Fairness constraints in CTL

- For LTL it holds: $TS \models_{fair} \varphi$ if and only if $TS \models (fair \rightarrow \varphi)$
- An analogous approach for CTL is not possible!
- Formulas form $\forall(fair \rightarrow \varphi)$ and $\exists(fair \wedge \varphi)$ needed
- But: boolean combinations of path formulae are not allowed in CTL
- and: e.g., strong fairness constraints $\Box\Diamond b \rightarrow \Box\Diamond c \equiv \Diamond\Box\neg b \;\vee\; \Diamond\Box c$
  - cannot be expressed in CTL since persistence properties cannot
- Solution: change the semantics of CTL by ignoring unfair paths

CTL fairness constraints

- A *strong CTL fairness constraint* is a formula of the form:

$$sfair = \bigwedge_{0<i\leqslant k} (\Box\Diamond\Phi_i \rightarrow \Box\Diamond\Psi_i)$$

  - where Φ_i and Ψ_i (for $0 < i \leqslant k$) are CTL-formulas over *AP*
  - weak and unconditional CTL fairness constraints are defined analogously, e.g.

$$ufair = \bigwedge_{0<i\leqslant k} \Box\Diamond\Psi_i \quad \text{and} \quad wfair = \bigwedge_{0<i\leqslant k} (\Diamond\Box\Phi_i \rightarrow \Box\Diamond\Psi_i)$$

  - a CTL fairness assumption $fair$ is a combination of $ufair$, $sfair$ and $wfair$

⇒ a CTL fairness constraint is an LTL formula over CTL state formulas!

  - note that $s \models \Phi_i$ and $s \models \Psi_i$ refer to standard (unfair!) CTL semantics

Semantics of fair CTL

For CTL fairness assumption $fair$, relation $\models_{fair}$ is defined by:

$$
\begin{array}{lll}
s \models_{fair} a & \text{iff} & a \in \mathit{Label}(s) \\
s \models_{fair} \neg \Phi & \text{iff} & \neg (s \models_{fair} \Phi) \\
s \models_{fair} \Phi \vee \Psi & \text{iff} & (s \models_{fair} \Phi) \vee (s \models_{fair} \Psi) \\
s \models_{fair} \exists \varphi & \text{iff} & \pi \models_{fair} \varphi \text{ for } \textit{some fair} \text{ path } \pi \text{ that starts in } s \\
s \models_{fair} \forall \varphi & \text{iff} & \pi \models_{fair} \varphi \text{ for } \textit{all fair} \text{ paths } \pi \text{ that start in } s
\end{array}
$$

$$
\begin{array}{ll}
\pi \models_{fair} \bigcirc \Phi & \text{iff } \pi[1] \models_{fair} \Phi \\
\pi \models_{fair} \Phi \, \mathsf{U} \, \Psi & \text{iff } (\exists\, j \geqslant 0.\, \pi[j] \models_{fair} \Psi \;\wedge\; (\forall\, 0 \leqslant k < j.\, \pi[k] \models_{fair} \Phi))
\end{array}
$$

π is a fair path iff $\pi \models_{LTL} fair$ for CTL fairness assumption $fair$

Transition system semantics

- For CTL-state-formula Φ, and fairness assumption $fair$:

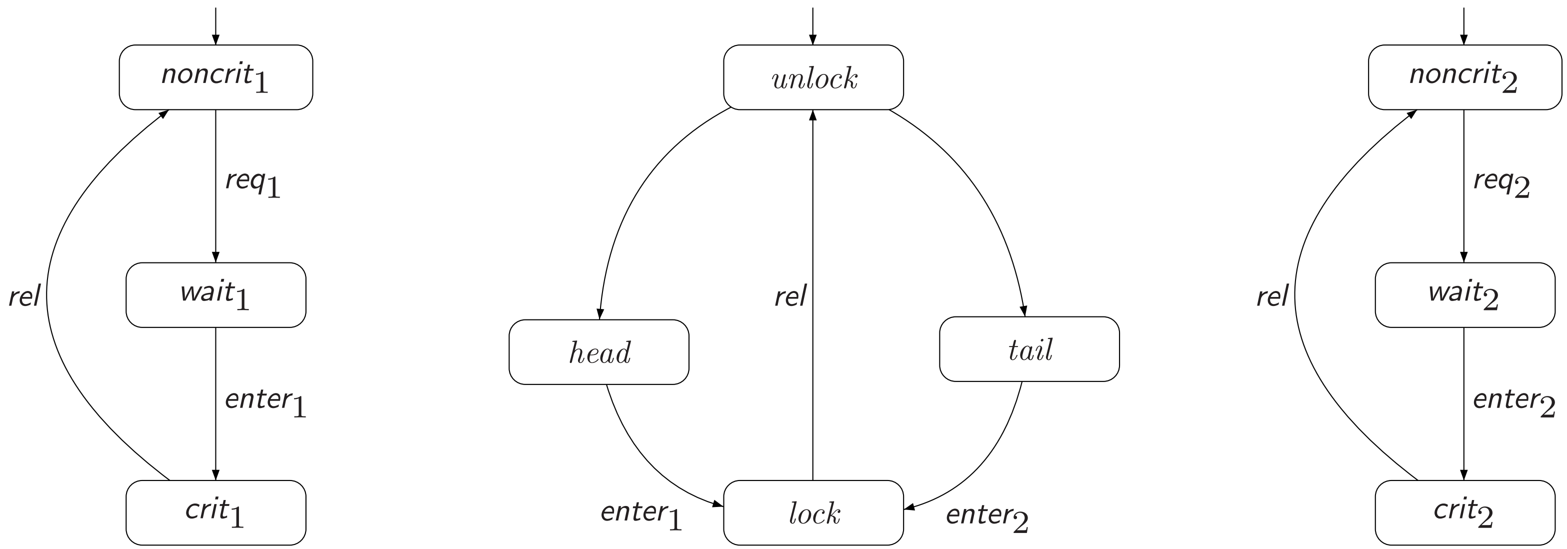
$$\mathit{Sat}_{fair}(\Phi) \;=\; \{\, s \in S \mid s \models_{fair} \Phi \,\}$$

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

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

  – this is equivalent to $I \subseteq \mathit{Sat}_{fair}(\Phi)$

Randomized arbiter



$$TS_1 \parallel Arbiter \parallel TS_2 \not\models (\forall\Box\forall\Diamond\, crit_1) \;\wedge\; (\forall\Box\forall\Diamond\, crit_2)$$

But: $TS_1 \parallel Arbiter \parallel TS_2 \models_{fair} \forall\Box\forall\Diamond crit_1 \;\wedge\; \forall\Box\forall\Diamond crit_2$ with

$$fair = \Box\Diamond head \;\wedge\; \Box\Diamond tail$$

Fair CTL model-checking problem

For:

- finite transition system *TS* without terminal states
- CTL formula Φ in ENF, and
- CTL fairness assumption $fair$

establish whether or not:

$$TS \models_{fair} \Phi$$

use bottom-up procedure à la CTL to determine $Sat_{fair}(\Phi)$
using as much as possible standard CTL model-checking algorithms

CTL fairness constraints

- A *strong CTL fairness constraint*: $sfair = \bigwedge_{0<i\leqslant k} (\Box\Diamond\Phi_i \rightarrow \Box\Diamond\Psi_i)$
  - where Φ_i and Ψ_i (for $0 < i \leqslant k$) are CTL-formulas over *AP*

- Replace the CTL state-formulas in $sfair$ by fresh atomic propositions:

$$sfair := \bigwedge_{0<i\leqslant k} (\Box\Diamond a_i \rightarrow \Box\Diamond b_i)$$

  - where $a_i \in L(s)$ if and only if $s \in Sat(\Phi_i)$ (not $Sat_{fair}(\Phi_i)$!)
  - $\ldots$ $b_i \in L(s)$ if and only if $s \in Sat(\Psi_i)$ (not $Sat_{fair}(\Psi_i)$!)
  - (for unconditional and weak fairness this goes similarly)

- Note: $\pi \models fair$ iff $\pi[j..] \models fair$ for some $j \geqslant 0$ iff $\pi[j..] \models fair$ for all $j \geqslant 0$

Results for $\models_{fair}$ (1)

$s \models_{fair} \exists \bigcirc a$ if and only if $\exists s' \in \mathit{Post}(s)$ with $s' \models a$ and $\mathit{FairPaths}(s') \neq \varnothing$

$s \models_{fair} \exists (a \,\mathsf{U}\, a')$ if and only if there exists a finite path fragment

$$s_0\, s_1\, s_2 \ldots s_{n-1} s_n \in \mathit{Paths}_{\mathit{fin}}(s) \quad \text{with } n \geqslant 0$$

such that $s_i \models a$ for $0 \leqslant i < n$, $s_n \models a'$, and $\mathit{FairPaths}(s_n) \neq \varnothing$

Results for $\models_{fair}$ (2)

$s \models_{fair} \exists \bigcirc a$ if and only if $\exists s' \in \mathit{Post}(s)$ with $s' \models a$ and $\underbrace{\mathit{FairPaths}(s') \neq \varnothing}_{s' \models_{fair} \exists \Box \mathsf{true}}$

$s \models_{fair} \exists (a \,\mathsf{U}\, a')$ if and only if there exists a finite path fragment

$$s_0\, s_1\, s_2 \ldots s_{n-1} s_n \in \mathit{Paths}_{\mathit{fin}}(s) \quad \text{with } n \geqslant 0$$

such that $s_i \models a$ for $0 \leqslant i < n$, $s_n \models a'$, and $\underbrace{\mathit{FairPaths}(s_n) \neq \varnothing}_{s_n \models_{fair} \exists \Box \mathsf{true}}$

Basic algorithm

- Determine $\mathit{Sat}_{\mathit{fair}}(\exists\Box\mathsf{true}) \;=\; \{\, s \in S \mid \mathit{FairPaths}(s) \neq \varnothing \,\}$
- Introduce an atomic proposition a_{fair} and adjust labeling where:
  - $a_{\mathit{fair}} \in L(s)$ if and only if $s \in \mathit{Sat}_{\mathit{fair}}(\exists\Box\mathsf{true})$
- Compute the sets $\mathit{Sat}_{\mathit{fair}}(\Psi)$ for all subformulas Ψ of Φ (in ENF) by:

$$
\begin{array}{rcl}
\mathit{Sat}_{\mathit{fair}}(a) & = & \{\, s \in S \mid a \in L(s) \,\} \\
\mathit{Sat}_{\mathit{fair}}(\neg a) & = & S \setminus \mathit{Sat}_{\mathit{fair}}(a) \\
\mathit{Sat}_{\mathit{fair}}(a \wedge a') & = & \mathit{Sat}_{\mathit{fair}}(a) \,\cap\, \mathit{Sat}_{\mathit{fair}}(a') \\
\mathit{Sat}_{\mathit{fair}}(\exists\bigcirc a) & = & \mathit{Sat}\,(\exists\bigcirc(a \wedge a_{\mathit{fair}})) \\
\mathit{Sat}_{\mathit{fair}}(\exists(a \,\mathsf{U}\, a')) & = & \mathit{Sat}\,\big(\exists(a \,\mathsf{U}\, (a' \wedge a_{\mathit{fair}}))\big) \\
\mathit{Sat}_{\mathit{fair}}(\exists\Box a) & = & \ldots\ldots
\end{array}
$$

- Thus: model checking CTL under fairness constraints is
  - CTL model checking + algorithm for computing $\mathit{Sat}_{\mathit{fair}}(\exists\Box a)$!

Model checking CTL with fairness

The model-checking problem for CTL with fairness can be reduced to:

(1) the model-checking problem for CTL (without fairness), and

(2) the problem of computing $Sat_{fair}(\exists\Box a)$ for $a \in AP$

note that $\exists\Box$true is a special case of $\exists\Box a$

thus a single algorithm suffices for $Sat_{fair}(\exists\Box a)$ and $Sat_{fair}(\exists\Box\text{true})$

Core model-checking algorithm

(* states are assumed to be labeled with a_i and b_i *)

compute $\mathit{Sat}_{\mathit{fair}}(\exists\Box\text{true}) = \{ s \in S \mid \mathit{FairPaths}(s) \neq \varnothing \}$
forall $s \in \mathit{Sat}_{\mathit{fair}}(\exists\Box\text{true})$ **do** $L(s) := L(s) \cup \{ a_{\mathit{fair}} \}$ **od**

(* compute $\mathit{Sat}_{\mathit{fair}}(\Phi)$ *)

for all $0 < i \leqslant |\,\Phi\,|$ **do**
  for all $\Psi \in \mathit{Sub}(\Phi)$ with $|\,\Psi\,| = i$ **do**
    switch(Ψ):

true	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := S$;
a	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := \{ s \in S \mid a \in L(s) \}$;
$a \wedge a'$	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := \{ s \in S \mid a, a' \in L(s) \}$;
$\neg a$	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := \{ s \in S \mid a \notin L(s) \}$;
$\exists\bigcirc a$	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := \mathit{Sat}(\exists\bigcirc(a \wedge a_{\mathit{fair}}))$;
$\exists(a \,\mathsf{U}\, a')$	:	$\mathit{Sat}_{\mathit{fair}}(\Psi) := \mathit{Sat}(\exists(a \,\mathsf{U}\, (a' \wedge a_{\mathit{fair}})))$;
$\exists\Box a$	:	compute $\mathit{Sat}_{\mathit{fair}}(\exists\Box a)$

    end switch
    replace all occurrences of Ψ (in Φ) by the fresh atomic proposition a_Ψ
    forall $s \in \mathit{Sat}_{\mathit{fair}}(\Psi)$ **do** $L(s) := L(s) \cup \{ a_\Psi \}$ **od**
  od
od
return $I \subseteq \mathit{Sat}_{\mathit{fair}}(\Phi)$

Characterization of $\mathit{Sat}_{fair}(\exists\Box a)$

$$s \models_{sfair} \exists\Box a \quad \text{where} \quad sfair = \bigwedge_{0<i\leqslant k} (\Box\Diamond a_i \rightarrow \Box\Diamond b_i)$$

iff there exists a finite path fragment $s_0 \ldots s_n$ and a cycle $s'_0 \ldots s'_r$ with:

1. $s_0 = s \quad$ and $\quad s_n = s'_0 = s'_r$
2. $s_i \models a$, for any $0 \leqslant i \leqslant n$, and $s'_j \models a$, for any $0 \leqslant j \leqslant r$, and
3. $\mathit{Sat}(a_i) \cap \{ s'_1, \ldots, s'_r \} = \varnothing$ or $\mathit{Sat}(b_i) \cap \{ s'_1, \ldots, s'_r \} \neq \varnothing$ for $0 < i \leqslant k$

Computing $Sat_{fair}(\exists\Box a)$

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

  $\Rightarrow$ each infinite path fragment in $TS[a]$ satisfies $\Box a$
- $s \models_{fair} \exists\Box a$ iff there is a non-trivial SCC D in $TS[a]$ reachable from s:

$$D \cap Sat(a_i) = \varnothing \quad \text{or} \quad D \cap Sat(b_i) \neq \varnothing \quad \text{for} \quad 0 < i \leqslant k \qquad (*)$$

- $Sat_{sfair}(\exists\Box a) = \{ s \in S \mid Reach_{TS[a]}(s) \cap T \neq \varnothing \}$
  - T is the union of all non-trivial SCCs C that contain D satisfying (*)

how to compute the set T of SCCs?

Unconditional fairness

$$ufair \;\equiv\; \bigwedge_{0<i\leqslant k} \Box\Diamond b_i$$

Let T be the set union of all non-trivial SCCs C of $TS[a]$ satisfying

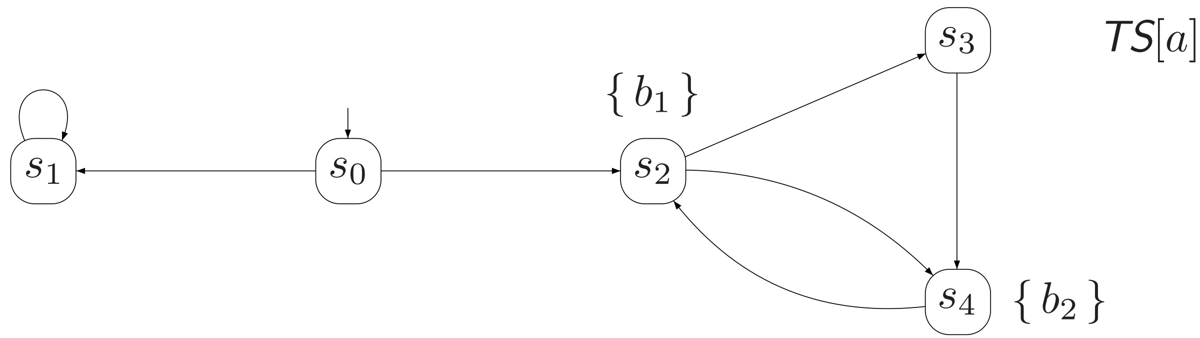
$$C \cap \mathit{Sat}(b_i) \;\neq\; \varnothing \quad \text{for all } 0 < i \leqslant k$$

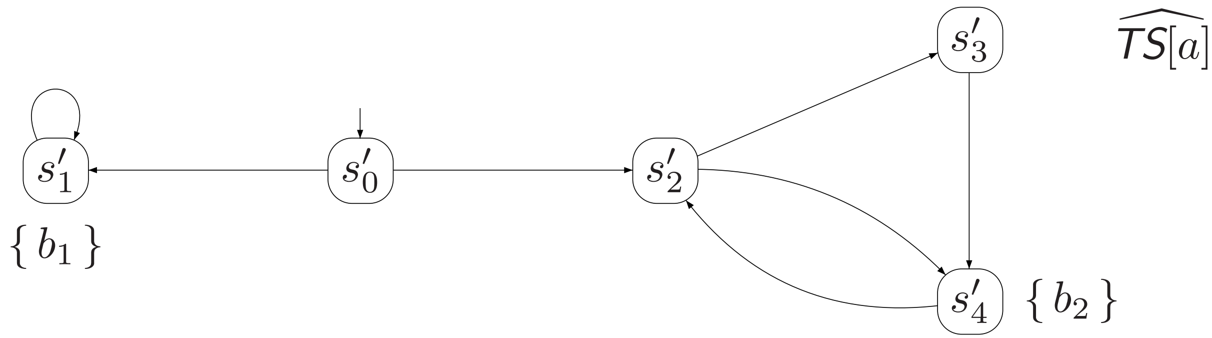
It now follows:

$$s \models_{ufair} \exists\Box a \quad \text{if and only if} \quad \mathit{Reach}_{TS[a]}(s) \cap T \neq \varnothing$$

$\Rightarrow$ T can be determined by a simple graph analysis (DFS)

Example





$TS[a] \models_{ufair} \exists \Box a$ but $\widehat{TS[a]} \not\models_{ufair} \exists \Box a$ with $ufair = \Box\Diamond b_1 \;\wedge\; \Box\Diamond b_2$

Strong fairness

- $sfair \;=\; \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$, i.e., $k{=}1$
- $s \models_{sfair} \exists\Box a$ iff C is a non-trivial SCC in $TS[a]$ reachable from s with:

  (1) $C \cap Sat(b_1) \neq \varnothing$, or

  (2) $D \cap Sat(a_1) = \varnothing$, for some non-trivial SCC D in C
- D is a non-trivial SCC in the graph that is obtained from $C[\neg a_1]$
- For T the union of non-trivial SCCs in satisfying (1) and (2):

$$s \models_{sfair} \exists\Box a \quad \text{if and only if} \quad Reach_{TS[a]}(s) \cap T \neq \varnothing$$

for several strong fairness constraints ($k > 1$), this is applied recursively

T is determined by standard graph analysis (DFS)

Time complexity

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

CTL formula Φ, and CTL fairness constraint $fair$ with k conjuncts,

the CTL model-checking problem *TS* $\models_{fair} \Phi$

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