Verification – Lecture 16
CTL vs. LTL

Bernd Finkbeiner – Sven Schewe
Rayna Dimitrova – Lars Kuhtz – Anne Proetzsch

Wintersemester 2007/2008

REVIEW

CTL

modal logic over infinite trees [Clarke & Emerson 1981]

- State formulas: Φ ::= true $\mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$
  - $a \in AP$ — atomic proposition
  - $\neg\,\Phi$ and $\Phi_1 \wedge \Phi_2$ — negation and conjunction
  - $\exists\varphi$ — there *exists* a path fulfilling φ
  - $\forall\varphi$ — *all* paths fulfill φ
- Path formulas: $\varphi :: \bigcirc\Phi \mid \Phi_1\,\mathsf{U}\,\Phi_2$
  - $\bigcirc\Phi$ — the next state fulfills Φ
  - $\Phi_1\,\mathsf{U}\,\Phi_2$ — Φ_1 holds until a Φ_2-state is reached

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

- $\forall\bigcirc\bigcirc\,\Phi$ and $\forall\exists\bigcirc\,\Phi \notin$ CTL, but $\forall\bigcirc\forall\bigcirc\,\Phi$ and $\forall\bigcirc\exists\bigcirc\,\Phi \in$ CTL

Derived operators

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

Transition system semantics

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

$$\mathit{Sat}(\Phi) \ = \ \{\, q \in Q \mid q \models \Phi \,\}$$

- State graph *S* satisfies CTL-formula Φ iff Φ holds in all its initial states:

$$S \models \Phi \quad \text{if and only if} \quad \forall q_0 \in Q_0.\, q_0 \models \Phi$$

  - this is equivalent to $Q_0 \subseteq \mathit{Sat}(\Phi)$

- Point of attention: $S \not\models \Phi$ and $S \not\models \neg\Phi$ is possible!
  - because of several initial states, e.g. $q_0 \models \exists\Box\,\Phi$ and $q_0' \not\models \exists\Box\,\Phi$

Model checking CTL

- How to check whether state graph *S* satisfies CTL formula $\widehat{\Phi}$?
  - convert the formula $\widehat{\Phi}$ into the equivalent Φ in ENF
  - compute *recursively* the set *Sat*$(\Phi) = \{ q \in S \mid q \models \Phi \}$
  - $S \models \Phi$ if and only if each initial state of *S* belongs to *Sat*(Φ)
- Recursive bottom-up computation of *Sat*(Φ):
  - consider the parse-tree of Φ
  - start to compute *Sat*(a_i), for all leafs in the tree
  - then go one level up in the tree and determine *Sat*$(\cdot)$ for these nodes

$$\text{e.g.,: } \mathit{Sat}(\underbrace{\Psi_1 \;\wedge\; \Psi_2}_{\text{node at level } i}) \;=\; \mathit{Sat}(\underbrace{\Psi_1}_{\substack{\text{node at}\\ \text{level } i-1}}) \cap \mathit{Sat}(\underbrace{\Psi_2}_{\substack{\text{node at}\\ \text{level } i-1}})$$

  - then go one level up and determine *Sat*$(\cdot)$ of these nodes
  - and so on....... until the root is treated, i.e., *Sat*(Φ) is computed

Existential normal form (ENF)

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

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

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

$$\forall\bigcirc\Phi \;\equiv\; \neg\exists\bigcirc\neg\Phi$$

$$\forall(\Phi \,\mathsf{U}\, \Psi) \;\equiv\; \neg\exists(\neg\Psi \,\mathsf{U}\, (\neg\Phi \wedge \neg\Psi)) \;\wedge\; \neg\exists\Box\,\neg\Psi$$

Characterization of *Sat* (1)

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

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

where $S = (Q, Q_0, E, L)$ is a finite state graph without terminal states

Characterization of *Sat* (2)

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

  (1) $Sat(\Psi) \subseteq T$ and (2) $(q \in Sat(\Phi)$ and $Successors(q) \cap T \neq \varnothing) \;\Rightarrow\; q \in T$

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

  (3) $T \subseteq Sat(\Phi)$ and (4) $q \in T$ implies $Successors(q) \cap T \neq \varnothing$

where $S = (Q, Q_0, E, L)$ is a state graph without terminal states

Computing *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$ (3)

Input: finite state graph *S* with state-set Q and CTL-formula $\exists(\Phi \,\mathsf{U}\, \Psi)$
Output: *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi)) = \{ q \in Q \mid q \models \exists(\Phi \,\mathsf{U}\, \Psi) \}$

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

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

$V := Q \setminus \mathit{Sat}(\Phi);$ (* V contains any not visited q' with $q' \not\models \exists\Box\Phi$ *)

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

for all $q \in \mathit{Sat}(\Phi)$ **do** $c[q] := |\, \mathit{Successors}(q) \,|;$ **od** (* initialize array c *)

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

Alternative algorithm for *Sat*$(\exists\Box\,\Phi)$

1. Consider only state q if $q \models \Phi$, otherwise *eliminate* q
   - change *S* into *S*$[\Phi] = (Q', Q'_0, E', L')$ with $Q' =$ *Sat*(Φ),
   - $E' = E \cap (Q' \times Q')$, $Q'_0 = Q_0 \cap Q'$, and $L'(q) = L(q)$ for $q \in Q'$

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

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

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

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

Time complexity

For state graph *S* with N states and K edges,
and CTL formula Φ, the CTL model-checking problem *S* $\models \Phi$
can be determined in time $\mathcal{O}(|\,\Phi\,| \cdot (N + M))$

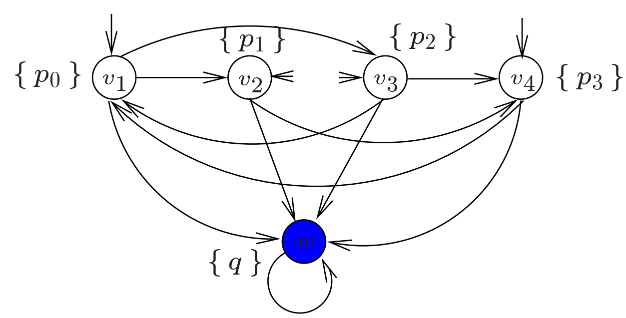
this applies to both algorithms for $\exists\Box\,\Phi$

Model-checking LTL versus CTL

- Let S be a state graph with N states and M edges
- 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?

Hamiltonian path problem

$\Rightarrow$ LTL-formulae can be *exponentially shorter* than their CTL-equivalent



- Existence of Hamiltonian path in LTL: $\bigwedge_i \Big(\Diamond p_i \;\wedge\; \Box(p_i \rightarrow \bigcirc\Box\neg p_i) \Big)$
- In CTL, all possible (= 4!) routes need to be encoded

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 state graph S (over *AP*):

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

- Let Φ be a CTL-formula, and φ the LTL-formula obtained by eliminating all path quantifiers in Φ. Then: [Clarke & Draghicescu]

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

Infinitely often

Example:

CTL-formula $\forall\Box\forall\Diamond a$ and LTL-formula $\Box\Diamond a$ are equivalent.

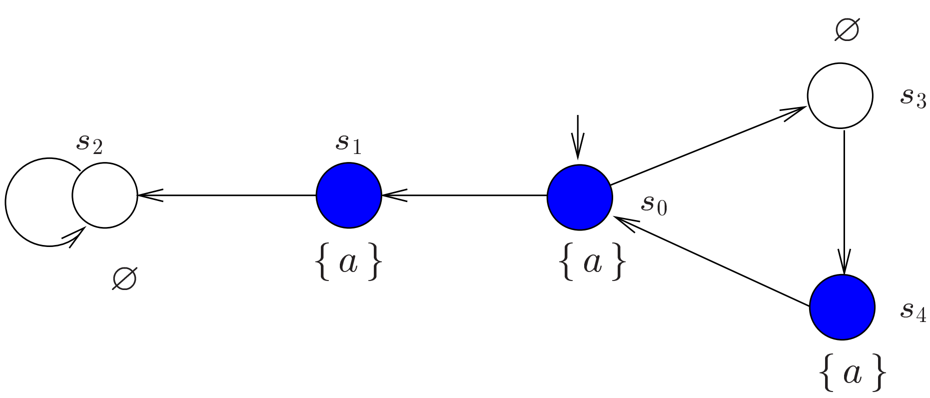
LTL and CTL are incomparable

- 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

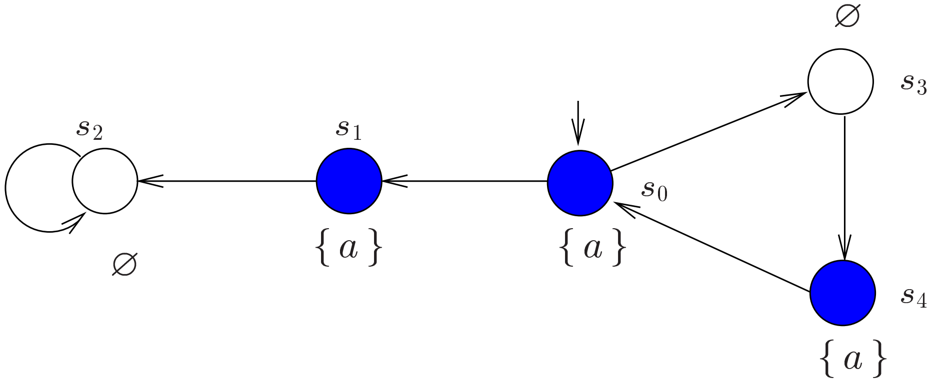
Comparing LTL and CTL (1)

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



Comparing LTL and CTL (1)

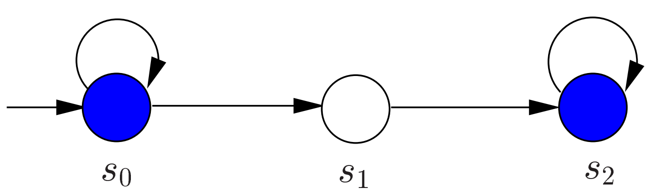
$\Diamond\,(a \;\wedge\; \bigcirc a)$ is not equivalent to $\forall\Diamond\,(a \;\wedge\; \forall\bigcirc a)$



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

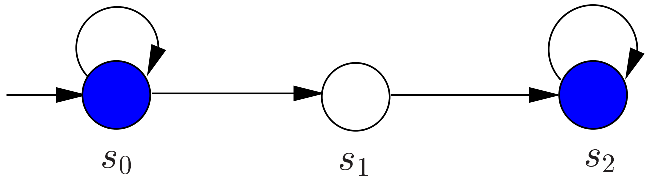
Comparing LTL and CTL (2)

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



Comparing LTL and CTL (2)

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

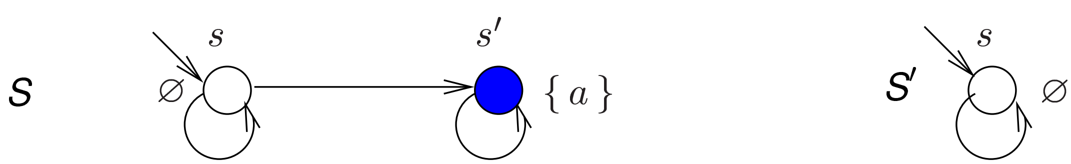


$s_0 \models \Diamond\Box\, a$ **but** $\underbrace{s_0 \not\models \forall\Diamond\,\forall\Box\, a}_{\text{path } s_0^\omega \text{ violates it}}$

Comparing LTL and CTL (3)

The CTL-formula $\forall\Box\,\exists\Diamond\, a$ cannot be expressed in LTL

- This is shown by contradiction: assume $\varphi \equiv \forall\Box\,\exists\Diamond\, a$; let:



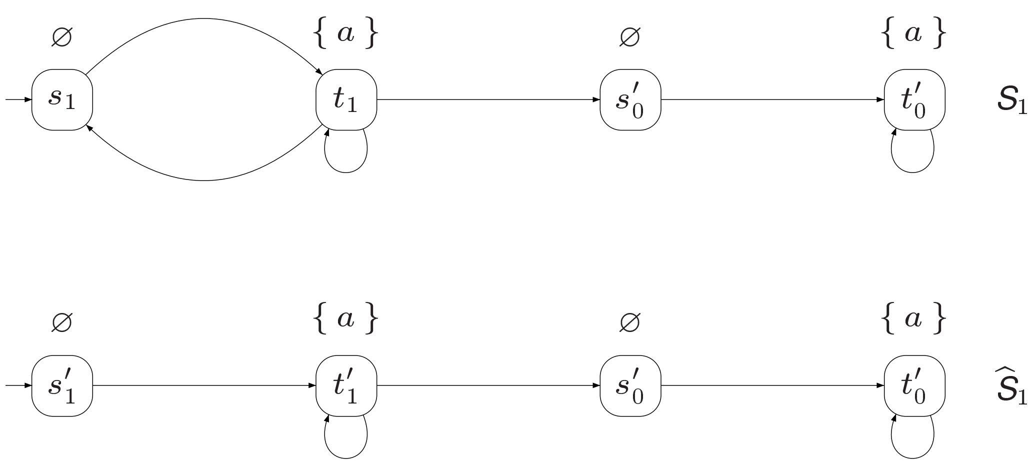
- $S \models \forall\Box\,\exists\Diamond\, a$, and thus—by assumption—$S \models \varphi$
- *Paths*$(S') \subseteq$ *Paths*(S), thus $S' \models \varphi$
- But $S' \not\models \forall\Box\,\exists\Diamond\, a$ as path $s^\omega \not\models \Box\,\exists\Diamond\, a$

Comparing LTL and CTL (4)

The LTL-formula $\Diamond\Box\, a$ cannot be expressed in CTL

- Provide two series of state graphs S_n and $\widehat{S}_n$
- such that $S_n \not\models \Diamond\Box\, a$ and $\widehat{S}_n \models \Diamond\Box\, a$ (*), and
- for any $\forall$CTL-formula Φ with $|\Phi| \leqslant n$: $S_n \models \Phi$ iff $\widehat{S}_n \models \Phi$ (**)
  - proof is by induction on n (omitted here)
- Assume there is a CTL-formula $\Phi \equiv \Diamond\Box\, a$ with $|\Phi| = n$
  - by (*), it follows $S_n \not\models \Phi$ and $\widehat{S}_n \models \Phi$
  - but this contradicts (**): $S_n \models \Phi$ if and only if $\widehat{S}_n \models \Phi$

The state graphs S_n and $\widehat{S}_n$ ($n = 1$)



only difference: S_n includes $t_n \to s_n$, while $\widehat{S}_n$ does not

LTL Fairness constraints

Let Φ and Ψ be propositional logic formulas over *AP*.

1. An *unconditional LTL fairness constraint* is of the form:

$$ufair \;=\; \Box\Diamond\Psi$$

2. A *strong LTL fairness condition (compassion)* is of the form:

$$sfair \;=\; \Box\Diamond\Phi \;\longrightarrow\; \Box\Diamond\Psi$$

3. A *weak LTL fairness constraint (justice)* is of the form:

$$wfair \;=\; \Diamond\Box\Phi \;\longrightarrow\; \Box\Diamond\Psi$$

A LTL fairness assumption $fair$ is a conjunction of LTL fairness constraints.

Fair satisfaction

For state q in state graph *S* (over *AP*) without terminal states, let

$$\begin{aligned} \mathit{FairPaths}_{fair}(q) &= \{\, \pi \in \mathit{Paths}(q) \mid \pi \models fair \,\} \\ \mathit{FairTraces}_{fair}(q) &= \{\, \mathit{trace}(\pi) \mid \pi \in \mathit{FairPaths}_{fair}(q) \,\} \end{aligned}$$

For LTL-formula φ, and fairness assumption $fair$:

$$\begin{aligned} q \models_{fair} \varphi \quad &\text{if and only if} \quad \forall \pi \in \mathit{FairPaths}_{fair}(q).\, \pi \models \varphi \quad \text{and} \\ S \models_{fair} \varphi \quad &\text{if and only if} \quad \forall q_0 \in Q_0.\, q_0 \models_{fair} \varphi \end{aligned}$$

$\models_{fair}$ is the *fair satisfaction relation* for LTL; $\models$ the standard one for LTL

Reducing $\models_{fair}$ to $\models$

For:

- state graph *S* without terminal states
- LTL formula φ, and
- LTL fairness assumption *fair*

it holds:

$$\boxed{S \models_{fair} \varphi \quad \text{if and only if} \quad S \models (fair \rightarrow \varphi)}$$

verifying an LTL-formula under a fairness assumption can be done
using standard LTL model-checking algorithms

Fairness constraints in CTL

- For LTL it holds: $S \models_{fair} \varphi$ if and only if $S \models (fair \rightarrow \varphi)$
- An analogous approach for CTL is not possible!
- Formulas of form $\forall(fair \rightarrow \varphi)$ and $\exists(fair \wedge \varphi)$ needed
- But: boolean combinations of path formulae are not allowed in CTL
- and: strong fairness constraints

$$\Box \Diamond b \rightarrow \Box \Diamond c \equiv \Diamond \Box \neg b \ \vee \ \Diamond \Box c$$

  cannot be expressed, since persistence properties are not in CTL

- 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 conjunction of CTL fairness constraints.

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

Semantics of fair CTL

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

$s \models_{fair} a$	iff	$a \in Label(s)$
$s \models_{fair} \neg\Phi$	iff	$\neg(s \models_{fair} \Phi)$
$s \models_{fair} \Phi \vee \Psi$	iff	$(s \models_{fair} \Phi) \vee (s \models_{fair} \Psi)$
$s \models_{fair} \exists\varphi$	iff	$\pi \models_{fair} \varphi$ for *some fair* path π that starts in s
$s \models_{fair} \forall\varphi$	iff	$\pi \models_{fair} \varphi$ for *all fair* paths π that start in s

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

π is a fair path iff $\pi \models fair$ for CTL fairness assumption $fair$

Transition system semantics

- For CTL-state-formula Φ, and fairness assumption *fair*, the *satisfaction set* $\mathit{Sat}_{\mathit{fair}}(\Phi)$ is defined by:

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

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

$$S \models_{\mathit{fair}} \Phi \quad \text{if and only if} \quad \forall q_0 \in Q_0 .\, q_0 \models_{\mathit{fair}} \Phi$$

  – this is equivalent to $Q_0 \subseteq \mathit{Sat}_{\mathit{fair}}(\Phi)$

Fair CTL model-checking problem

For:

- finite state graph S without terminal states
- CTL formula Φ in ENF, and
- CTL fairness assumption *fair*

establish whether or not:

$$S \models_{\mathit{fair}} \Phi$$

use bottom-up procedure a la CTL to determine $\mathit{Sat}_{\mathit{fair}}(\Phi)$
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*(Φ_i) (not *Sat*$_{\mathit{fair}}(\Phi_i)$!)
  - $\ldots$ $b_i \in L(s)$ if and only if $s \in$ *Sat*(Ψ_i) (not *Sat*$_{\mathit{fair}}(\Psi_i)$!)
  - (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 $\exists s' \in$ *Successors*(s) with $s' \models a$ and *FairPaths*$(s') \neq \varnothing$

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

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

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

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

$s \models_{fair} \exists\bigcirc a$ if and only if $\exists s' \in$ *Successors*(s) with $s' \models a$ and $\underbrace{\textit{FairPaths}(s') \neq \varnothing}_{s' \models_{fair} \exists\Box \text{true}}$

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

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

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

Basic algorithm

- Determine $\textit{Sat}_{fair}(\exists\Box \text{true}) = \{ q \in Q \mid \textit{FairPaths}(q) \neq \varnothing \}$
- Introduce an atomic proposition a_{fair} such that:
  - $a_{fair} \in L(q)$ if and only if $q \in \textit{Sat}_{fair}(\exists\Box \text{true})$
- Compute the sets $\textit{Sat}_{fair}(\Psi)$ for all subformulas Ψ of Φ (in ENF) by:

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

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

Core model-checking algorithm

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

compute $Sat_{fair}(\exists\Box\,\text{true}) = \{ q \in Q \mid FairPaths(q) \neq \varnothing \}$
forall $q \in Sat_{fair}(\exists\Box\,\text{true})$ **do** $L(q) := L(q) \cup \{ a_{fair} \}$ **od**

(* compute $Sat_{fair}(\Phi)$ *)

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

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

    end switch
    replace all occurrences of Ψ (in Φ) by the fresh atomic proposition a_Ψ
    forall $q \in Sat_{fair}(\Psi)$ **do** $L(q) := L(q) \cup \{ a_\Psi \}$ **od**
  od
od
return $Q_0 \subseteq Sat_{fair}(\Phi)$

Characterization of $Sat_{fair}(\exists\Box\, a)$

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

iff there exists a finite path fragment $q_0 \ldots q_n$ and a cycle $q'_0 \ldots q'_r$ with:

1. $q_0 = q$ and $q_n = q'_0 = q'_r$
2. $q_i \models a$, for any $0 \leqslant i \leqslant n$, and $q'_j \models a$, for any $0 \leqslant j \leqslant r$, and
3. $Sat(a_i) \cap \{ q'_1, \ldots, q'_r \} = \varnothing$ or $Sat(b_i) \cap \{ q'_1, \ldots, q'_r \} \neq \varnothing$ for $0 < i \leqslant k$