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

⇒ 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”

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

- **Branching** temporal logic:
  
  “statements about all or some paths starting in a state”

  - \( s \models \forall \Box(x \leq 20) \) iff for all paths starting in \( s \) always \( x \leq 20 \)
  - \( s \models \exists \Box(x \leq 20) \) iff for some path starting in \( s \) always \( x \leq 20 \)
  - 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

\[(s_0, 0) \rightarrow (s_1, 1) \rightarrow (s_2, 2) \rightarrow (s_3, 3) \rightarrow (s_2, 4) \rightarrow \ldots \]

\[(s_0, 0) \rightarrow (s_1, 1) \rightarrow (s_2, 2) \rightarrow (s_3, 3) \rightarrow (s_3, 4) \rightarrow (s_2, 4) \rightarrow (s_3, 4) \rightarrow \ldots \]
<table>
<thead>
<tr>
<th>“behavior” in a state $s$</th>
<th>path-based: $\text{trace}(s)$</th>
<th>state-based: computation tree of $s$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>temporal logic</strong></td>
<td>LTL: path formulas $\varphi$ $s \models \varphi$ iff $\forall \pi \in \text{Paths}(s). \pi \models \varphi$</td>
<td>CTL: state formulas existential path quantification $\exists \varphi$ universal path quantification: $\forall \varphi$</td>
</tr>
<tr>
<td><strong>complexity of the model checking problems</strong></td>
<td>PSPACE–complete $O(</td>
<td>TS</td>
</tr>
<tr>
<td><strong>implementation-relation</strong></td>
<td>trace inclusion and the like (proof is PSPACE-complete)</td>
<td>simulation and bisimulation (proof in polynomial time)</td>
</tr>
</tbody>
</table>
Computation tree logic

modal logic over infinite trees [Clarke & Emerson 1981]

- **Statements over states**
  - $a \in AP$  
    atomic proposition
  - $\neg \Phi$ and $\Phi \land \Psi$  
    negation and conjunction
  - $\exists \varphi$  
    there exists a path fulfilling $\varphi$
  - $\forall \varphi$  
    all paths fulfill $\varphi$

- **Statements over paths**
  - $\Box \Phi$  
    the next state fulfills $\Phi$
  - $\Phi U \Psi$  
    $\Phi$ holds until a $\Psi$-state is reached

$\Rightarrow$ note that $\Box$ and $U$ *alternate* with $\forall$ and $\exists$
Derived operators

potentially $\Phi$: $\exists \diamondsuit \Phi = \exists (\text{true } \cup \Phi)$

inevitably $\Phi$: $\forall \diamondsuit \Phi = \forall (\text{true } \cup \Phi)$

potentially always $\Phi$: $\exists \Box \Phi := \neg \forall \diamondsuit \neg \Phi$

invariantly $\Phi$: $\forall \Box \Phi = \neg \exists \diamondsuit \neg \Phi$

weak until: $\exists (\Phi \mathcal{W} \Psi) = \neg \forall ((\Phi \land \neg \Psi) \cup (\neg \Phi \land \neg \Psi))$

$\forall (\Phi \mathcal{W} \Psi) = \neg \exists ((\Phi \land \neg \Psi) \cup (\neg \Phi \land \neg \Psi))$

the boolean connectives are derived as usual
Visualization of semantics

∀ ♦ red

∃ ♦ red

∃ (yellow ∪ red)

∀ ♦ red

∀ ♦ red

∀ (yellow ∪ red)
Semantics of CTL state-formulas

Defined by a relation $|=\phi$ such that

- $s |= \phi$ if and only if formula $\phi$ holds in state $s$

- $s |= a$ iff $a \in L(s)$
- $s |= \neg \phi$ iff $\neg (s |= \phi)$
- $s |= \phi \land \psi$ iff $(s |= \phi) \land (s |= \psi)$
- $s |= \exists \phi$ iff $\pi |= \phi$ for some path $\pi$ that starts in $s$
- $s |= \forall \phi$ iff $\pi |= \phi$ for all paths $\pi$ that start in $s$
Semantics of CTL path-formulas

Define a relation $\models$ such that

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

$\pi \models \Box \Phi$ iff $\pi[1] \models \Phi$

$\pi \models \Phi \cup \Psi$ iff $(\exists j \geq 0. \pi[j] \models \Psi) \land (\forall 0 \leq k < j. \pi[k] \models \Phi))$

where $\pi[i]$ denotes the state $s_i$ in the path $\pi$
Transition system semantics

• For CTL-state-formula $\Phi$, the *satisfaction set* $\text{Sat}(\Phi)$ is defined by:

$$\text{Sat}(\Phi) = \{ s \in S \mid s \models \Phi \}$$

• $TS$ satisfies CTL-formula $\Phi$ iff $\Phi$ holds in all its initial states:

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

• **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 $\Phi$ and $\Psi$ (over $AP$) are \textit{equivalent}, denoted $\Phi \equiv \Psi$ if and only if $\text{Sat}(\Phi) = \text{Sat}(\Psi)$ for all transition systems $TS$ over $AP$

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

Recall in LTL: $\varphi U \psi \equiv \psi \lor (\varphi \land \Box (\varphi U \psi))$

In CTL:

$$\forall(\Phi U \Psi) \equiv \Psi \lor (\Phi \land \forall \Box \forall(\Phi U \Psi))$$

$$\forall \Diamond \Phi \equiv \Phi \lor \forall \Box \forall \Diamond \Phi$$

$$\forall \Box \Phi \equiv \Phi \land \forall \Box \forall \Box \Phi$$

$$\exists(\Phi U \Psi) \equiv \Psi \lor (\Phi \land \exists \Box \exists(\Phi U \Psi))$$

$$\exists \Diamond \Phi \equiv \Phi \lor \exists \Box \exists \Diamond \Phi$$

$$\exists \Box \Phi \equiv \Phi \land \exists \Box \exists \Box \Phi$$
Distributive laws

Recall in LTL: $\Box(\varphi \land \psi) \equiv \Box \varphi \land \Box \psi$ and $\Diamond(\varphi \lor \psi) \equiv \Diamond \varphi \lor \Diamond \psi$

In CTL:

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

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

note that $\exists \Box (\Phi \land \Psi) \not\equiv \exists \Box \Phi \land \exists \Box \Psi$ and $\forall \Diamond (\Phi \lor \Psi) \not\equiv \forall \Diamond \Phi \lor \forall \Diamond \Psi$
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:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \Box \Phi \mid \exists (\Phi_1 \cup \Phi_2) \mid \exists \Box \Phi$$

For each CTL formula, there exists an equivalent CTL formula in ENF.
Model checking CTL

- Convert the formula $\Phi'$ into an equivalent $\Phi$ in ENF

- How to check whether $TS$ satisfies $\Phi$?
  - compute \textit{recursively} the set $Sat(\Phi)$ of states that satisfy $\Phi$
  - check whether all initial states belong to $Sat(\Phi)$

- Recursive \textbf{bottom-up} computation:
  - consider the \textit{parse-tree} of $\Phi$
  - 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., $\Phi$) is checked
Example

\[ \Phi = \exists \bigcirc a \land \exists (b \cup \exists \square \neg c) . \]

\( Sat(\Psi) \) \( \exists \bigcirc \) \( Sat(\Phi) \) \( \land \)
\( Sat(\Psi') \) \( \exists \bigcup \) \( Sat(\Psi'') \)
\( a \) \( b \) \( \neg \)
\( c \)
Characterization of $Sat$ (1)

For all $CTL$ formulas $\Phi, \Psi$ over $AP$ it holds:

\[
Sat(true) = S \\
Sat(a) = \{ s \in S \mid a \in L(s) \}, \text{ for any } a \in AP \\
Sat(\Phi \land \Psi) = Sat(\Phi) \cap Sat(\Psi) \\
Sat(\neg \Phi) = S \setminus Sat(\Phi) \\
Sat(\exists \Box \Phi) = \{ s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset \}
\]

where $TS = (S, Act, \rightarrow, I, AP, L)$ is a transition system without terminal states
Characterization of $Sat$ (2)

For all $CTL$ formulas $\Phi, \Psi$ over $AP$ it holds:

- $Sat(\exists (\Phi U \Psi))$ is the smallest subset $T$ of $S$, such that:
  1. $Sat(\Psi) \subseteq T$ and
  2. $s \in Sat(\Phi)$ and $Post(s) \cap T \neq \emptyset$ 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 $Post(s) \cap T \neq \emptyset$

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

switch($\Phi$):

\begin{align*}
a & : \quad \text{return } \{ s \in S \mid a \in L(s) \}; \\
\ldots & : \quad \ldots \\
\exists \bigcirc \Psi & : \quad \text{return } \{ s \in S \mid \text{Post}(s) \cap \text{Sat}(\Psi) \neq \emptyset \}; \\
\exists (\Phi_1 \cup \Phi_2) & : \quad T := \text{Sat}(\Phi_2); \quad (* \text{compute the smallest fixed point} *) \\
& \quad \text{while } \text{Sat}(\Phi_1) \setminus T \cap \text{Pre}(T) \neq \emptyset \text{ do} \\
& \quad \quad \text{let } s \in \text{Sat}(\Phi_1) \setminus T \cap \text{Pre}(T); \\
& \quad \quad T := T \cup \{ s \}; \\
& \quad \text{od; } \\
& \quad \text{return } T; \\
\exists \square \Psi & : \quad T := \text{Sat}(\Psi); \quad (* \text{compute the greatest fixed point} *) \\
& \quad \text{while } \exists s \in T. \text{Post}(s) \cap T = \emptyset \text{ do} \\
& \quad \quad \text{let } s \in \{ s \in T \mid \text{Post}(s) \cap T = \emptyset \}; \\
& \quad \quad T := T \setminus \{ s \}; \\
& \quad \text{od; } \\
& \quad \text{return } T; \\
\end{align*}

end switch
Computing $\text{Sat}(\exists(\Phi \cup \Psi))$
Computing $Sat(\exists(\Phi \cup \Psi))$

**Input:** finite transition system $TS$ with state-set $S$ and CTL-formula $\exists(\Phi \cup \Psi)$

**Output:** $Sat(\exists(\Phi \cup \Psi))$

\[
E := Sat(\Psi); \quad \text{(* $E$ administers the states $s$ with $s \models \exists(\Phi \cup \Psi)$ *)}
\]
\[
T := E; \quad \text{(* $T$ contains the already visited states $s$ with $s \models \exists(\Phi \cup \Psi)$ *)}
\]

while $E \neq \emptyset$ 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 $\exists \diamond ((p = r) \land (p \neq q))$
The computation in snapshots

\[
\begin{align*}
\{ p, q, r \} & \quad \{ r \} \\
\{ q, r \} & \quad \{ p \} \\
\{ q \} & \quad \{ p, q \} \\
\{ p, q \} & \quad \{ p, r \} \\
\{ p \} & \quad \{ p \} \\
\{ q \} & \quad \{ q \} \\
\{ r \} & \quad \{ r \} \\
\emptyset & \quad \emptyset
\end{align*}
\]
Computing \( \text{Sat}(\exists \Box \Phi) \)

\[
E := S \setminus \text{Sat}(\Phi); \\
T := \text{Sat}(\Phi); \\
\text{for all } s \in \text{Sat}(\Phi) \text{ do } c[s] := |\text{Post}(s)|; \text{ od} \\
\text{while } E \neq \emptyset \text{ do } \\
\quad \text{let } s' \in E; \\
\quad E := E \setminus \{s'\}; \\
\quad \text{for all } s \in \text{Pre}(s') \text{ do } \\
\quad \quad \text{if } s \in T \text{ then } \\
\quad \quad \quad c[s] := c[s] - 1; \\
\quad \quad \quad \text{if } c[s] = 0 \text{ then } \\
\quad \quad \quad \quad T := T \setminus \{s\}; E := E \cup \{s\}; \\
\quad \quad \text{fi} \\
\quad \quad \text{fi} \\
\quad \text{od} \\
\text{od} \\
\text{return } T
\]
**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 $s \in S'$
   ⇒ 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
   ⇒ 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)
For transition system $TS$ with $N$ states and $K$ transitions, and CTL formula $\Phi$, the CTL model-checking problem $TS \models \Phi$ can be determined in time $O(|\Phi| \cdot (N + M))$.
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
Equivalence of LTL and CTL formulas

• CTL-formula $\Phi$ and LTL-formula $\varphi$ (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 $\Phi$ be a CTL-formula, and $\varphi$ the LTL-formula that is obtained by eliminating all path quantifiers in $\Phi$. Then:

$\Phi \equiv \varphi$ 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.,
  
  - ♦□a
  - ♦(a ∧ ◯ a)

• Some CTL-formulas cannot be expressed in LTL, e.g.,
  
  - ∀♦∀□a
  - ∀♦(a ∧ ∀◊a)
  - ∀□∃♦a

⇒ Cannot be expressed = there does not exist an equivalent formula
Comparing LTL and CTL (1)

\[ \Diamond (a \land \Box a) \text{ is not equivalent to } \forall \Diamond (a \land \forall \Box a) \]

![Diagram showing the comparison between LTL and CTL expressions.](image)
Comparing LTL and CTL (1)

\( \Diamond (a \land \Diamond a) \) is not equivalent to \( \forall \Diamond (a \land \forall \Diamond a) \)

Since path \( s_0 s_1 (s_2)^{\omega} \) violates \( \Diamond (a \land \forall \Diamond a) \)
Comparing LTL and CTL (2)

\( \forall \Diamond \forall \Box a \) is not equivalent to \( \Diamond \Box a \)

\[ s_0 \rightarrow s_1 \rightarrow s_2 \]
Comparing LTL and CTL (2)

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

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

since path $s_0^\omega$ violates $\Diamond \forall \Box a$
Comparing LTL and CTL (3)

- No LTL-formula $\varphi$ is equivalent to $\forall \square \exists \Diamond a$

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

$$TS\vdash \forall \square \exists \Diamond a,$$

and thus—by assumption—$TS\vdash \varphi$

- $Paths(TS') \subseteq Paths(TS)$, thus $TS' \models \varphi$

- But $TS' \not\models \forall \square \exists \Diamond a$ as path $s^\omega \not\models \square \exists \Diamond a$

© JPK
Model-checking LTL versus CTL

• Let $TS$ be a transition system with $N$ states and $M$ transitions

• Model-checking LTL-formula $\Phi$ has time-complexity $O((N+M) \cdot 2^{|\Phi|})$
  – linear in the state space of the system model
  – exponential in the length of the formula

• Model-checking CTL-formula $\Phi$ has time-complexity $O((N+M) \cdot |\Phi|)$
  – 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 \textit{exponentially shorter} than their equivalent in CTL

- Existence of Hamiltonian path in LTL: \( \neg ((\Diamond p_0 \land \ldots \land \Diamond p_3) \land \lozenge^4 q) \)

- 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

⇒ Fairness
  - fair CTL semantics, model checking
Fairness constraints in CTL

- For LTL it holds: $TS \models^{\text{fair}} \varphi$ if and only if $TS \models (\text{fair} \rightarrow \varphi)$

- An analogous approach for CTL is not possible!

- Formulas form $\forall(\text{fair} \rightarrow \varphi)$ and $\exists(\text{fair} \land \varphi)$ needed

- **But:** boolean combinations of path formulae are not allowed in CTL

- **and:** e.g., strong fairness constraints $\Box \Diamond b \rightarrow \Box \Diamond c \equiv \Diamond \Box \neg b \lor \Diamond \Box c$
  - 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:

\[
sfair = \bigwedge_{0 < i \leq k} (\Box \Diamond \Phi_i \rightarrow \Box \Diamond \Psi_i)
\]

- where \(\Phi_i\) and \(\Psi_i\) (for \(0 < i \leq k\)) are CTL-formulas over \(AP\)
- weak and unconditional CTL fairness constraints are defined analogously, e.g.

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

- a **CTL fairness assumption** \(fair\) is a combination of \(ufair\), \(sfair\) and \(wfair\)

\Rightarrow a CTLa 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:

\[
\begin{align*}
  s &\models_{fair} a & \text{iff} & & a \in Label(s) \\
  s &\models_{fair} \neg \Phi & \text{iff} & & \neg (s \models_{fair} \Phi) \\
  s &\models_{fair} \Phi \lor \Psi & \text{iff} & & (s \models_{fair} \Phi) \lor (s \models_{fair} \Psi) \\
  s &\models_{fair} \exists \varphi & \text{iff} & & \pi \models_{fair} \varphi \text{ for some fair path } \pi \text{ that starts in } s \\
  s &\models_{fair} \forall \varphi & \text{iff} & & \pi \models_{fair} \varphi \text{ for all fair paths } \pi \text{ that start in } s \\
\end{align*}
\]

\[
\begin{align*}
  \pi &\models_{fair} \bigcirc \Phi & \text{iff} & & \pi[1] \models_{fair} \Phi \\
  \pi &\models_{fair} \Phi \lor \Psi & \text{iff} & & (\exists j \geq 0. \pi[j] \models_{fair} \Psi \land (\forall 0 \leq k < j. \pi[k] \models_{fair} \Phi)) \\
\end{align*}
\]

\( \pi \) is a fair path iff \( \pi \models_{LTL} fair \) for CTL fairness assumption \( fair \)
Transition system semantics

• For CTL-state-formula $\Phi$, and fairness assumption $fair$:

$$Sat_{\text{fair}}(\Phi) = \{ s \in S \mid s \models_{\text{fair}} \Phi \}$$

• $TS$ satisfies CTL-formula $\Phi$ iff $\Phi$ holds in all its initial states:

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

  – this is equivalent to $I \subseteq Sat_{\text{fair}}(\Phi)$
Randomized arbiter

\[ TS_1 \parallel Arbiter \parallel TS_2 \not\models (\forall \square \forall \Diamond crit_1) \land (\forall \square \forall \Diamond crit_2) \]

But: \[ TS_1 \parallel Arbiter \parallel TS_2 \models fair \ \forall \square \forall \Diamond crit_1 \land \forall \square \forall \Diamond crit_2 \text{ with} \]

\[ fair = \Box \Diamond head \land \Box \Diamond tail \]
Fair CTL model-checking problem

For:

- finite transition system $TS$ without terminal states
- CTL formula $\Phi$ 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: \( \text{sfair} = \bigwedge_{0 < i \leq k} (\Box \Diamond \Phi_i \rightarrow \Box \Diamond \Psi_i) \)
  - where \( \Phi_i \) and \( \Psi_i \) (for \( 0 < i \leq k \)) are CTL-formulas over \( AP \)

- Replace the CTL state-formulas in \( \text{sfair} \) by fresh atomic propositions:
  \[
  \text{sfair} := \bigwedge_{0 < i \leq k} (\Box \Diamond a_i \rightarrow \Box \Diamond b_i)
  \]
  - where \( a_i \in L(s) \) if and only if \( s \in \text{Sat}(\Phi_i) \)
  - \( \ldots b_i \in L(s) \) if and only if \( s \in \text{Sat}(\Psi_i) \)
  - (for unconditional and weak fairness this goes similarly)

- Note: \( \pi \models fair \) iff \( \pi[j..] \models fair \) for some \( j \geq 0 \) iff \( \pi[j..] \models fair \) for all \( j \geq 0 \)
Results for $\models_{fair} (1)$

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

$s \models_{fair} \exists (a \cup a')$ if and only if there exists a finite path fragment $s_0 s_1 s_2 \ldots s_{n-1} s_n \in \text{Paths}_{\text{fin}}(s)$ with $n \geq 0$

such that $s_i \models a$ for $0 \leq i < n$, $s_n \models a'$, and $\text{FairPaths}(s_n) \neq \emptyset$
Results for $\models_{fair} (2)$

$s \models_{fair} \exists \circ a$ if and only if $\exists s' \in Post(s)$ with $s' \models a$ and $\exists s' \models \Box true$  if and only if $\exists s' \in Post(s)$ with $s' \models a$ and $\forall s' \not= \emptyset$

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

$$s_0 s_1 s_2 \ldots s_{n-1} s_n \in Paths_{fin}(s) \quad \text{with } n \geq 0$$

such that $s_i \models a$ for $0 \leq i < n$, $s_n \models a'$, and $\forall s_n \not= \emptyset$

$s_n \models_{fair} \exists \Box true$  if and only if $\exists s' \models \Box true$
Basic algorithm

- Determine $\text{Sat}_{\text{fair}}(\exists \Box \text{true}) = \{ s \in S \mid \text{FairPaths}(s) \neq \emptyset \}$

- Introduce an atomic proposition $a_{\text{fair}}$ and adjust labeling where:
  - $a_{\text{fair}} \in L(s)$ if and only if $s \in \text{Sat}_{\text{fair}}(\exists \Box \text{true})$

- Compute the sets $\text{Sat}_{\text{fair}}(\Psi)$ for all subformulas $\Psi$ of $\Phi$ (in ENF) by:

  $\text{Sat}_{\text{fair}}(a) = \{ s \in S \mid a \in L(s) \}$
  $\text{Sat}_{\text{fair}}(\neg a) = S \setminus \text{Sat}_{\text{fair}}(a)$
  $\text{Sat}_{\text{fair}}(a \land a') = \text{Sat}_{\text{fair}}(a) \cap \text{Sat}_{\text{fair}}(a')$
  $\text{Sat}_{\text{fair}}(\exists \Diamond a) = \text{Sat}(\exists \Diamond (a \land a_{\text{fair}}))$
  $\text{Sat}_{\text{fair}}(\exists(a \lor a')) = \text{Sat}(\exists(a \lor (a' \land a_{\text{fair}})))$
  $\text{Sat}_{\text{fair}}(\exists a) = \ldots \ldots$

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

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

thus a single algorithm suffices for $\text{Sat}_{\text{fair}}(\exists \Box a)$ and $\text{Sat}_{\text{fair}}(\exists \Box \text{true})$
Core model-checking algorithm

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

compute $Sat_{fair}(∃□true) = \{ s \in S \mid FairPaths(s) \neq \emptyset \}$

forall $s \in Sat_{fair}(∃□true)$ do $L(s) := L(s) \cup \{ a_{fair} \}$ od

(* compute $Sat_{fair}(Φ)$ *)

for all $0 < i \leq |Φ|$ do
  for all $Ψ \in Sub(Φ)$ with $|Ψ| = i$ do
    switch(Ψ):
      true : $Sat_{fair}(Ψ) := S$;
      $a$ : $Sat_{fair}(Ψ) := \{ s \in S \mid a \in L(s) \}$;
      $a \land a'$ : $Sat_{fair}(Ψ) := \{ s \in S \mid a, a' \in L(s) \}$;
      $¬a$ : $Sat_{fair}(Ψ) := \{ s \in S \mid a \notin L(s) \}$;
      $∃□a$ : $Sat_{fair}(Ψ) := Sat(∃□(a \land a_{fair}))$;
      $∃(a ∪ a')$ : $Sat_{fair}(Ψ) := Sat(∃(a ∪ (a' \land a_{fair})))$;
      $∃□a$ : compute $Sat_{fair}(∃□a)$
    end switch
    replace all occurrences of $Ψ$ (in $Φ$) by the fresh atomic proposition $a_Ψ$
    forall $s \in Sat_{fair}(Ψ)$ do $L(s) := L(s) \cup \{ a_Ψ \}$ od
  od
return $I \subseteq Sat_{fair}(Φ)$
Characterization of $\text{Sat}_{\text{fair}}(\exists \Box a)$

$s \models_{\text{fair}} \exists \Box a$ \quad where \quad \text{sfair} = \bigwedge_{0<i\leq 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 $0 \leq i \leq n$, and $s'_j \models a$, for any $0 \leq j \leq r$, and

3. $\text{Sat}(a_i) \cap \{ s'_1, \ldots, s'_r \} = \emptyset$ or $\text{Sat}(b_i) \cap \{ s'_1, \ldots, s'_r \} \neq \emptyset$ for $0 < i \leq k$
Computing $\text{Sat}_{\text{fair}}(\exists \Box a)$

- Consider only state $s$ if $s \models a$, otherwise *eliminate* $s$
  
  - change $TS$ into $TS[a] = (S', Act, \rightarrow', I', AP, L')$ with $S' = \text{Sat}(a)$,
  - $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $I' = I \cap S'$, and $L'(s) = L(s)$ for $s \in S'$
  
  ⇒ each infinite path fragment in $TS[a]$ satisfies $\Box a$

- $s \models_{\text{fair}} \exists \Box a$ iff there is a non-trivial SCC $D$ in $TS[a]$ reachable from $s$:
  \[ D \cap \text{Sat}(a_i) = \emptyset \quad \text{or} \quad D \cap \text{Sat}(b_i) \neq \emptyset \quad \text{for} \quad 0 < i \leq k \]  
  
- $\text{Sat}_{\text{sfair}}(\exists \Box a) = \{ s \in S \mid \text{Reach}_{TS[a]}(s) \cap T \neq \emptyset \}$
  
  - $T$ is the union of all non-trivial SCCs $C$ that contain $D$ satisfying (*)

how to compute the set $T$ of SCCs?
Unconditional fairness

\[ \text{ufair} \equiv \bigwedge_{0<i\leq k} \square \Diamond b_i \]

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

\[ C \cap \text{Sat}(b_i) \neq \emptyset \text{ for all } 0 < i \leq k \]

It now follows:

\[ s \models_{\text{ufair}} \exists \square a \text{ if and only if } \text{Reach}_{TS[a]}(s) \cap T \neq \emptyset \]

\[ \Rightarrow T \text{ can be determined by a simple graph analysis (DFS) } \]
Example

\[
\begin{align*}
TS[a] &\models_{ufair} \exists a \text{ but } \overline{TS[a]} \not\models_{ufair} \exists a \\
\text{with } ufair = \Box \Diamond b_1 \land \Box \Diamond b_2
\end{align*}
\]
Strong fairness

- $sfair = □◊a_1 → □◊b_1$, i.e., $k=1$

- $s \models_{sfair} ∃a$ iff $C$ is a non-trivial SCC in $TS[a]$ reachable from $s$ with:
  1. $C \cap Sat(b_1) ≠ ∅$, or
  2. $D \cap Sat(a_1) = ∅$, for some non-trivial SCC $D$ in $C$

- $D$ is a non-trivial SCC in the graph that is obtained from $C[¬a_1]$

- For $T$ the union of non-trivial SCCs in satisfying (1) and (2):

  $s \models_{sfair} ∃a$ if and only if $Reach_{TS[a]}(s) \cap T ≠ ∅$

  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 $\Phi$, and CTL fairness constraint $fair$ with $k$ conjuncts, the CTL model-checking problem $TS \models_{fair} \Phi$ can be determined in time $O(|\Phi| \cdot (N + M) \cdot k)$.