Model Checking
Lecture #14: Fairness
[Baier & Katoen, Chapter 3.5, 5.1.6, 6.5]

Joost-Pieter Katoen

Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
Does This Multi-Threaded Program Terminate?

\[ \text{Inc} \parallel \parallel \text{Reset} \]

where

\[
\begin{align*}
\text{thread Inc} &= \textbf{while} \langle x \geq 0 \textbf{ do } x := x + 1 \textbf{ od} \\
\text{thread Reset} &= x := -1
\end{align*}
\]

\( x \) is a shared integer variable that initially has value 0
Is It Possible To Starve?
Thread Two Starves

Is it **fair** that thread two never gets access to the critical section despite infinitely often having the possibility to do so?
Fairness

- Starvation freedom is often considered under *thread fairness*
  \[\Rightarrow\] there is a fair scheduling of the execution of threads
Fairness

- Starvation freedom is often considered under thread fairness
  ⇒ there is a fair scheduling of the execution of threads

- Fairness is concerned with a fair resolution of non-determinism
  | such that it is not biased to consistently ignore a possible option
The Relevance of Fairness

Fairness

- Starvation freedom is often considered under thread fairness
  ⇒ there is a fair scheduling of the execution of threads

- Fairness is concerned with a fair resolution of non-determinism
  ▶ such that it is not biased to consistently ignore a possible option

- Fairness is typically needed to prove a liveness property
  ▶ to prove some form of progress, progress needs to be possible
  ▶ fairness does not affect safety properties
Fairness

▶ Starvation freedom is often considered under thread fairness
  ⇒ there is a fair scheduling of the execution of threads

▶ Fairness is concerned with a fair resolution of non-determinism
  ▶ such that it is not biased to consistently ignore a possible option

▶ Fairness is typically needed to prove a liveness property
  ▶ to prove some form of progress, progress needs to be possible
  ▶ fairness does not affect safety properties

▶ Problem: liveness properties constrain infinite behaviours
  ▶ but some traces—that are unfair—refute the liveness property
Fairness Constraints

- What is wrong with our examples? Nothing!
  - interleaving: not realistic as no processor is $\infty$ faster than another
  - semaphore-based mutual exclusion: level of abstraction
Fairness Constraints

- What is wrong with our examples? Nothing!
  - interleaving: not realistic as no processor is $\infty$ faster than another
  - semaphore-based mutual exclusion: level of abstraction

- Rule out “unrealistic” executions by imposing fairness constraints
  - what to rule out? $\Rightarrow$ different kinds of fairness constraints
The Relevance of Fairness

Fairness Constraints

- What is wrong with our examples? Nothing!
  - interleaving: not realistic as no processor is $\infty$ faster than another
  - semaphore-based mutual exclusion: level of abstraction

- Rule out “unrealistic” exectuions by imposing fairness constraints
  - what to rule out? $\Rightarrow$ different kinds of fairness constraints

- “A thread gets its turn infinitely often”
  - always unconditional fairness
  - if it is enabled infinitely often strong fairness
  - if it is continuously enabled from some point on weak fairness
This program terminates assuming unconditional (thread) fairness:

\[
\text{thread } \text{Inc} = \text{while } (x \geq 0 \text{ do } x := x + 1) \text{ od}
\]

\[
\text{thread } \text{Reset} = x := -1
\]

as thread Reset eventually will set \( x \) to \(-1\)

\( x \) is a shared integer variable that initially has value 0.
Avoiding Starvation by Fairness

If the infinitely often enabled $\text{enter}_2$ action is not ignored infinitely often, thread two does not starve.
Avoiding Starvation by Fairness

Note that $\text{enter}_2$ is not enabled continuously during the run. Weak fairness this does not suffice.
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
LTL Fairness Constraints

Definition: LTL fairness constraints

Let $\Phi$ and $\Psi$ 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 is of the form:

   $$ sfair = \Box \Diamond \Phi \rightarrow \Box \Diamond \Psi $$

3. A weak LTL fairness constraint is of the form:

   $$ wfair = \Diamond \Box \Phi \rightarrow \Box \Diamond \Psi $$

$\Phi$ stands for “... is enabled”; $\Psi$ for “... is taken”
Relating Fairness Constraints

unconditional fair \implies \text{strong fair} \implies \text{weak fair}.
Fairness Assumptions

Definition: fairness assumption

An LTL fairness assumption is a conjunction of LTL fairness constraints. The general format of fairness assumption \( \text{fair} \) is

\[
\text{fair} = \text{ufair} \land \text{sfair} \land \text{wfair}
\]
Fair Traces and Fair Satisfaction

Definition: fair paths and fair traces

For state $s$ in transition system $TS$ (over $AP$) and LTL fairness assumption $fair$, let

$$\text{FairPaths}_{\text{fair}}(s) = \{ \pi \in \text{Paths}(s) \mid \pi \vDash fair \}$$

$$\text{FairTraces}_{\text{fair}}(s) = \{ \text{trace}(\pi) \mid \pi \in \text{FairPaths}_{\text{fair}}(s) \}.$$
**Fair Traces and Fair Satisfaction**

**Definition: fair paths and fair traces**

For state $s$ in transition system $TS$ (over $AP$) and LTL fairness assumption $fair$, let

$$FairPaths_{fair}(s) = \{ \pi \in Paths(s) \mid \pi \Vdash fair \}$$

$$FairTraces_{fair}(s) = \{ \text{trace}(\pi) \mid \pi \in FairPaths_{fair}(s) \}.$$ 

**Definition: fair satisfaction relation**

For LTL-formula $\varphi$, and LTL fairness assumption $fair$:

$$s \Vdash_{fair} \varphi \text{ if and only if } \forall \pi \in FairPaths_{fair}(s). \pi \Vdash \varphi$$

$$TS \Vdash_{fair} \varphi \text{ if and only if } \forall s_0 \in I. s_0 \Vdash_{fair} \varphi.$$ 

The relation $\Vdash_{fair}$ is the $fair$ satisfaction relation for LTL.
Example: Fair Runs and Fair Traces

Let $\Phi = \text{"action enter 2 is enabled"}$ and $\Psi = \text{"action enter 2 is taken"}$.

Run $\langle n_1, n_2, 1 \rangle \xrightarrow{\text{req} 1} \langle w_1, n_2, 1 \rangle \xrightarrow{\text{enter} 1} \langle c_1, n_2, 0 \rangle \xrightarrow{\text{rel}} \langle n_1, n_2, 1 \rangle \xrightarrow{\text{req} 1} \ldots$

... is not unconditionally fair

... but strongly fair, as action enter 2 is never enabled along the run

Run $\langle n_1, n_2, 1 \rangle \xrightarrow{\text{req} 2} \langle n_1, w_2, 1 \rangle \xrightarrow{\text{req} 1} \langle w_1, w_2, 1 \rangle \xrightarrow{\text{enter} 1} \langle c_1, w_2, 0 \rangle \xrightarrow{\text{rel}} \ldots$

... is not strongly fair as enter 2 is $\infty$ often enabled but never taken

... but weakly fair for as enter 2 is not always enabled along the run
Example: Fair Runs and Fair Traces

Let $\Phi = \text{“action } enter_2 \text{ is enabled”}$ and $\Psi = \text{“action } enter_2 \text{ is taken”}$
Example: Fair Runs and Fair Traces

- Let $\Phi = \text{“action enter}_2\text{ is enabled” and } \Psi = \text{“action enter}_2\text{ is taken”}

- Run $\langle n_1, n_2, 1 \rangle \xrightarrow{\text{req}_1} \langle w_1, n_2, 1 \rangle \xrightarrow{\text{enter}_1} \langle c_1, n_2, 0 \rangle \xrightarrow{\text{rel}} \langle n_1, n_2, 1 \rangle \xrightarrow{\text{req}_1} \ldots$
  
    - ... is not unconditionally fair
    - ... but strongly fair, as action enter$_2$ is never enabled along the run
Example: Fair Runs and Fair Traces

Let $\Phi = \text{“action } enter_2 \text{ is enabled”}$ and $\Psi = \text{“action } enter_2 \text{ is taken”}$

Run $\langle n_1, n_2, 1 \rangle \xrightarrow{\text{req}_1} \langle w_1, n_2, 1 \rangle \xrightarrow{\text{enter}_1} \langle c_1, n_2, 0 \rangle \xrightarrow{\text{rel}} \langle n_1, n_2, 1 \rangle \xrightarrow{\text{req}_1} \ldots$

... is not unconditionally fair

... but strongly fair, as action $enter_2$ is never enabled along the run

Run $\langle n_1, n_2, 1 \rangle \xrightarrow{\text{req}_2} \langle n_1, w_2, 1 \rangle \xrightarrow{\text{req}_1} \langle w_1, w_2, 1 \rangle \xrightarrow{\text{enter}_1} \langle c_1, w_2, 0 \rangle \xrightarrow{\text{rel}} \ldots$

... is not strongly fair as $enter_2$ is $\infty$ often enabled but never taken

... but weakly fair for as $enter_2$ is not always enabled along the run
Example: An Arbiter for Mutual Exclusion

\[
TS_1 \parallel \text{Arbiter} \parallel TS_2 \not\models \square \Diamond \text{crit}_1
\]

But:
\[
TS_1 \parallel \text{Arbiter} \parallel TS_2 \models_{\text{fair}} \square \Diamond \text{crit}_1 \land \square \Diamond \text{crit}_2
\]

with \( \text{fair} = \square \Diamond \text{head} \land \square \Diamond \text{tail} \)
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
Realisable Fairness

Definition: realisable fairness

Fairness assumption \( \text{fair} \) is realisable for transition system \( TS \) if for any reachable state \( s \): \( \text{FairPaths}_{\text{fair}}(s) \neq \emptyset \).

A fairness assumption is realisable for \( TS \) if every initial finite path fragment of \( TS \) can be completed to a fair run.
The Fairness Suffix Property

For any (infinite) fair path $\pi$, it holds

1. all suffixes of $\pi$ are fair too.
2. any finite path extended by $\pi$ is fair.

Proof.

Rather straightforward.
Realisable Fairness and Safety

Safety properties are preserved under realisable fairness

For transition system $TS$ and safety property $E_{safe}$ (both over $AP$) and fair $a$ realisable fairness assumption for $TS$:

$$TS \models E_{safe} \text{ if and only if } TS \models_{fair} E_{safe}.$$ 

Proof.
Realisable Fairness and Safety

Safety properties are preserved under realisable fairness

For transition system $TS$ and safety property $E_{safe}$ (both over $AP$) and $fair$ a realisable fairness assumption for $TS$:

$$TS \models E_{safe} \quad \text{if and only if} \quad TS \models_{fair} E_{safe}.$$  

Proof.

Non-realisable fairness may harm safety properties. Shown by example.
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
The Fair LTL Model-Checking Problem

Given:

1. a finite transition system $TS$
2. an LTL formula $\varphi$, and
3. an LTL fairness assumption $fair$

Question: does $TS \models_{fair} \varphi$?
For transition system $TS$, LTL formula $\varphi$ and LTL fairness assumption $fair$: $TS \models \varphi$ if and only if $TS \models (fair \to \varphi)$. This approach is not applicable to CTL (as we will discuss).
Fair LTL Model Checking

For transition system $TS$, LTL formula $\varphi$ and LTL fairness assumption $fair$:

$$TS \models_{fair} \varphi$$  \hspace{1cm} if and only if \hspace{1cm} $$TS \models (fair \rightarrow \varphi)$$

fair LTL model checking

LTL model checking

The fair LTL model-checking problem for $\varphi$ under fairness assumption $fair$ can be reduced to the LTL model-checking problem for $fair \rightarrow \varphi$. 
Fair LTL Model Checking

For transition system $TS$, LTL formula $\varphi$ and LTL fairness assumption $fair$:

$$TS \models_{fair} \varphi$$

if and only if

$$TS \models (fair \rightarrow \varphi)$$

The fair LTL model-checking problem for $\varphi$ under fairness assumption $fair$ can be reduced to the LTL model-checking problem for $fair \rightarrow \varphi$.

This approach is not applicable to CTL (as we will discuss).
Which Fairness Notion?

- Fairness constraints aim to rule out “unreasonable” runs
Which Fairness Notion?

- Fairness constraints aim to rule out “unreasonable” runs

- **Too strong?** $\Rightarrow$ reasonable runs ruled out. Verification result:
  - “false”: error found
  - “true”: don’t know as some relevant execution may refute it

- **Too weak?** $\Rightarrow$ too many runs considered. Verification result:
  - “true”: formula holds
  - “false”: don’t know, as refutation maybe due to an unreasonable run
Which Fairness Notion?

- Fairness constraints aim to rule out “unreasonable” runs

- Too strong? ⇒ reasonable runs ruled out. Verification result:
  - “false”: error found
  - “true”: don’t know as some relevant execution may refute it

- Too weak? ⇒ too many runs considered. Verification result:
  - “true”: formula holds
  - “false”: don’t know, as refutation maybe due to an unreasonable run
Which Fairness Notion?

- Fairness constraints aim to rule out “unreasonable” runs

- **Too strong?** ⇒ reasonable runs ruled out. Verification result:
  - “false”: error found
  - “true”: don’t know as some relevant execution may refute it

- **Too weak?** ⇒ too many runs considered. Verification result:
  - “true”: formula holds
  - “false”: don’t know, as refutation maybe due to an unreasonable run

Rules of thumb:

- strong (or unconditional) fairness is useful for solving contentions
- weak fairness is useful to resolve unfair scheduling of threads
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
Fairness Constraints in CTL

▶ For LTL it holds: $TS \models_{fair} \varphi$ if and only if $TS \models (\text{fair} \rightarrow \varphi)$
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
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 \( \exists (fair \land \varphi) \) needed

Solution: change the semantics of CTL by ignoring unfair paths
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
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: strong fairness constraint $\Box \Diamond b \rightarrow \Box \Diamond c$, i.e., $\Diamond \Box \neg b \lor \Diamond \Box c$ cannot be expressed in CTL as persistence properties are not in CTL
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: strong fairness constraint $\Box \Diamond b \rightarrow \Box \Diamond c$, i.e., $\Diamond \Box \neg b \lor \Diamond \Box c$ cannot be expressed in CTL as persistence properties are not in CTL

- Solution: change the semantics of CTL by ignoring unfair paths
**Definition: 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 state-formulas over \( AP \).
### CTL Fairness Constraints

**Definition: 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 state-formulas over $AP$.

Weak and unconditional CTL fairness constraints are defined similarly, e.g.:

$$ ufair = \bigwedge_{0<i\leq k} \Box \Diamond \Psi_i \quad \text{and} \quad wfair = \bigwedge_{0<i\leq k} (\Diamond \Box \Phi_i \rightarrow \Box \Diamond \Psi_i). $$
**Definition: CTL fairness constraints**

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

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

where \(\Phi_i\) and \(\Psi_i\) (for \(0 < i \leq k\)) are CTL state-formulas over \(AP\).

Weak and unconditional CTL fairness constraints are defined similarly, e.g.:

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

**Definition: CTL fairness assumption**

A CTL fairness assumption is a conjunction of \(\text{ufair}, \text{sfair}\) and \(\text{wfair}\).
CTL Fairness Assumptions

**Definition: CTL fairness constraints**

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

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

where \(\Phi_i\) and \(\Psi_i\) (for \(0 < i \leq k\)) are CTL state-formulas over \(AP\).

Weak and unconditional CTL fairness constraints are defined similarly, e.g.:

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

**Definition: CTL fairness assumption**

A CTL fairness assumption is a conjunction of \(\text{ufair}, \text{sfair}\) and \(\text{wfair}\).

A CTL fairness constraint is an LTL formula over CTL state formulas. \(\Phi_i\) and \(\Psi_i\) are interpreted by the standard (unfair) CTL semantics.
Semantics of Fair CTL

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

- $s \models_{\textit{fair}} a$ iff $a \in L(s)$
- $s \models_{\textit{fair}} \neg \Phi$ iff $\neg (s \models_{\textit{fair}} \Phi)$
- $s \models_{\textit{fair}} \Phi \lor \Psi$ iff $(s \models_{\textit{fair}} \Phi) \lor (s \models_{\textit{fair}} \Psi)$
- $s \models_{\textit{fair}} \exists \varphi$ iff $\pi \models_{\textit{fair}} \varphi$ for some fair path $\pi$ that starts in $s$
- $s \models_{\textit{fair}} \forall \varphi$ iff $\pi \models_{\textit{fair}} \varphi$ for all fair paths $\pi$ that start in $s$

- $\pi \models_{\textit{fair}} \bigcirc \Phi$ iff $\pi[1] \models_{\textit{fair}} \Phi$
- $\pi \models_{\textit{fair}} \Phi \cup \Psi$ iff $(\exists j \geq 0. \pi[j] \models_{\textit{fair}} \Psi$ and $(\forall 0 \leq i < j. \pi[i] \models_{\textit{fair}} \Phi))$

$\pi$ is a fair path iff $\pi \models_{LTL} \textit{fair}$ for CTL fairness assumption $\textit{fair}$
Transition System Semantics

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

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

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

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

- This is equivalent to $I \subseteq Sat_{fair}(\Phi)$
Example: An Arbiter for Mutual Exclusion

\[ TS_1 \parallel \text{Arbiter} \parallel TS_2 \not\models (\forall \square \forall \Diamond \text{crit}_1) \land (\forall \square \forall \Diamond \text{crit}_2) \]

But: \[ TS_1 \parallel \text{Arbiter} \parallel TS_2 \models_{\text{fair}} \forall \square \forall \Diamond \text{crit}_1 \land \forall \square \forall \Diamond \text{crit}_2 \]

with \[ \text{fair} = \square \Diamond \text{head} \land \square \Diamond \text{tail} \]
Example

unconditional fairness: $\textit{ufair} = \Box \Diamond \exists \Box \textit{start}$

$\text{Sat}(\exists \Box \textit{start}) = \{\text{delivered}\}$

$\textit{ufair} \equiv \Box \Diamond \text{delivered}$
Example

- **unconditional fairness:** \( ufair = \square \Diamond \exists \Diamond start \)

- **weak fairness:** \( wfair = \Diamond \square \exists \Diamond delivered \rightarrow \square \Diamond delivered \)

\[
\text{Sat}(\exists \Diamond delivered) = \{ try\_to\_send \}
\]

\( wfair \equiv \Diamond \square try\_to\_send \rightarrow \square \Diamond delivered \)
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
The Fair CTL Model-Checking Problem

Given:

1. a finite transition system $T_S$

2. an CTL state-formula\(^1\) $\Phi$, and

3. a CTL fairness assumption $fair$

Question: does $T_S \vDash_{fair} \Phi$?

\(^1\)Assumed to be in existential normal form.
The Fair CTL Model-Checking Problem

Given:

1. a finite transition system $TS$

2. an CTL state-formula\(^1\) $\Phi$, and

3. a CTL fairness assumption $fair$

Question: does $TS \models_{fair} \Phi$?

use recursive descent à la CTL to determine $Sat_{fair}(\Phi)$

using as much as possible standard CTL model-checking algorithms

\(^1\)Assumed to be in existential normal form.
Treating Strong CTL Fairness Constraints

Let **strong** CTL fairness constraint: \( sfair = \bigwedge_{0 < i \leq k} (\Box \Diamond \Phi_i \rightarrow \Box \Diamond \Psi_i) \)

where \( \Phi_i \) and \( \Psi_i \) (for \( 0 < i \leq k \)) are CTL state-formulas over \( AP \)
Treating Strong CTL Fairness Constraints

Let strong CTL fairness constraint: 

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

where \( \Phi_i \) and \( \Psi_i \) (for \( 0 < i \leq k \)) are CTL state-formulas over \( AP \)

Replace the CTL state-formulas in \( sfair \) by fresh atomic propositions:

\[ sfair := \bigwedge_{0<i\leq k} (\Box \Diamond a_i \rightarrow \Box \Diamond b_i) \]

where \( a_i \in L(s) \) if and only if \( s \in Sat(\Phi_i) \)

\( \text{not Sat}_{fair}(\Phi_i) \)

\( \ldots b_i \in L(s) \) if and only if \( s \in Sat(\Psi_i) \)

\( \text{not Sat}_{fair}(\Psi_i) \)
Treating Strong CTL Fairness Constraints

- Let strong CTL fairness constraint: 
  \[ sfair = \bigwedge_{0<i\leq k} (\Box \Diamond \Phi_i \rightarrow \Box \Diamond \Psi_i) \]
  where \( \Phi_i \) and \( \Psi_i \) (for \( 0 < i \leq k \)) are CTL state-formulas over \( AP \)

- Replace the CTL state-formulas in \( sfair \) by fresh atomic propositions:
  \[ sfair := \bigwedge_{0<i\leq k} (\Box \Diamond a_i \rightarrow \Box \Diamond b_i) \]
  where \( a_i \in L(s) \) if and only if \( s \in \text{Sat}(\Phi_i) \)
  \[ \text{(not Sat}_{fair}(\Phi_i)) \]
  where \( b_i \in L(s) \) if and only if \( s \in \text{Sat}(\Psi_i) \)
  \[ \text{(not Sat}_{fair}(\Psi_i)) \]

- For unconditional and weak fairness this goes similarly
Treating Strong CTL Fairness Constraints

Let strong CTL fairness constraint: $sfair = \bigwedge_{0 < i \leq k} (\Box \Diamond \Phi_i \rightarrow \Box \Diamond \Psi_i)$

where $\Phi_i$ and $\Psi_i$ (for $0 < i \leq k$) are CTL state-formulas over $AP$

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

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

where $a_i \in L(s)$ if and only if $s \in Sat(\Phi_i)$ and $b_i \in L(s)$ if and only if $s \in Sat(\Psi_i)$

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 $j \geq 0$
Some Useful Results

For CTL fairness assumption $fair$ and $a, a' \in AP$ it holds:

1. $s \models_{fair} \exists \Diamond a$ iff $\exists s' \in Post(s)$ with $s' \models a$ and $\text{FairPaths}_{fair}(s') \neq \emptyset$

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

$$s_0 \, s_1 \, s_2 \, \ldots \, s_{n-1} \, s_n \in \text{Paths}^*(s) \quad \text{with} \quad n \geq 0$$

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

**Proof.**

On the black board.
Example

\[ T : \]

\[ \{ c \} \not\models a_{fair} \]

\[ \{ b \} \not\models a_{fair} \]

\[ s \not\models_{fair} \exists (\neg b U c) \]

\[ s \not\models \exists (\neg b U (c \land a_{fair})) \]

\[ Sat(c \land a_{fair}) = \emptyset \]

strong fairness assumption: \( fair = \Box \Diamond b \rightarrow \Box \Diamond c \)

\[ T \models \exists (\neg b U c), \text{ but } T \not\models_{fair} \exists (\neg b U c) \]
Fair Path Existence

\[ \text{FairPaths}_{\text{fair}(s)} \neq \emptyset \text{ if and only if } s \models_{\text{fair}} \exists \Box \text{true}. \]
**Fair Path Existence**

\[ \text{FairPaths}_{\text{fair}}(s) \neq \emptyset \text{ if and only if } s \models_{\text{fair}} \exists \Box \text{true}. \]

**Example**

![Graph](image)

- \( \mathbb{O} = \{a\} \)
- \( \mathbb{E} = \emptyset \)

\( \text{fair} = \Box \diamond a \)

\[ \text{Sat}_{\text{fair}}(\exists \Box \text{true}) = \{s_0, s_2\} \]
Basic Model-Checking Algorithm for Fair CTL

Determine $Sat_{\text{fair}}(\exists \Box \text{true}) = \{ s \in S \mid FairPaths_{\text{fair}}(s) \neq \emptyset \}$
Basic Model-Checking Algorithm for Fair CTL

▶ Determine $Sat_{fair}(\exists \Box \text{true}) = \{ s \in S \mid FairPaths_{fair}(s) \neq \emptyset \}$

▶ Introduce an atomic proposition $a_{fair}$ and adjust labeling where:
  
  ▶ $a_{fair} \in L(s)$ if and only if $s \in Sat_{fair}(\exists \Box \text{true})$
Basic Model-Checking Algorithm for Fair CTL

▶ Determine $Sat_{\text{fair}}(\exists \Box \text{true}) = \{ s \in S \mid FairPaths_{\text{fair}}(s) \neq \emptyset \}$

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

▶ Compute the sets $Sat_{\text{fair}}(\Psi)$ for all sub-formulas $\Psi$ of $\Phi$ (in ENF) by:

\[
\begin{align*}
Sat_{\text{fair}}(a) & = \{ s \in S \mid a \in L(s) \} \\
Sat_{\text{fair}}(\neg a) & = S \setminus Sat_{\text{fair}}(a) \\
Sat_{\text{fair}}(a \land a') & = Sat_{\text{fair}}(a) \cap Sat_{\text{fair}}(a') \\
Sat_{\text{fair}}(\exists \bigcirc a) & = Sat(\exists \bigcirc(a \land a_{\text{fair}})) \\
Sat_{\text{fair}}(\exists(a \lor a')) & = Sat(\exists(a \lor (a' \land a_{\text{fair}}))) \\
Sat_{\text{fair}}(\exists \Box a) & = \ldots.
\end{align*}
\]
Basic Model-Checking Algorithm for Fair CTL

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

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

3. Compute the sets $Sat_{fair}(\Psi)$ for all sub-formulas $\Psi$ of $\Phi$ (in ENF) by:
   - $Sat_{fair}(a) = \{ s \in S \mid a \in L(s) \}$
   - $Sat_{fair}(\neg a) = S \setminus Sat_{fair}(a)$
   - $Sat_{fair}(a \land a') = Sat_{fair}(a) \cap Sat_{fair}(a')$
   - $Sat_{fair}(\exists \bigcirc a) = Sat(\exists \bigcirc (a \land a_{fair}))$
   - $Sat_{fair}(\exists(a U a')) = Sat(\exists(a U (a' \land a_{fair})))$
   - $Sat_{fair}(\exists \Box a) = \ldots \ldots$

4. Thus: model checking CTL under fairness constraints is
   - CTL model checking + algorithm for computing $Sat_{fair}(\exists \Box a)$
Model Checking CTL with Fairness

Model checking CTL with fairness can be done by combining

- the model-checking algorithm for CTL (without fairness), and
- an algorithm for computing $\text{Sat}_{\text{fair}}(\exists \Box a)$ for $a \in AP$. 

Model Checking CTL with Fairness

Model checking CTL with fairness can be done by combining

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

- an algorithm for computing $\text{Sat}_{\text{fair}}(\exists \Box a)$ for $a \in AP$.

As $\exists \Box \text{true}$ is a special case of $\exists \Box a$, an algorithm for $\text{Sat}_{\text{fair}}(\exists \Box a)$ can be used for $\text{Sat}_{\text{fair}}(\exists \Box \text{true})$. 
Basic Fair CTL Algorithm

\[
\text{compute } Sat_{\text{fair}}(\exists \Box \text{true}) = \{ s \in S \mid \text{FairPaths}(s) \neq \emptyset \} \\
\text{forall } s \in Sat_{\text{fair}}(\exists \Box \text{true}) \text{ do } L(s) := L(s) \cup \{ a_{\text{fair}} \} \od \\
\text{forall } 0 < i \leq |\Phi| \text{ do} \\
\quad \text{for all } \Psi \in \text{Sub}(\Phi) \text{ with } |\Psi| = i \text{ do} \\
\quad \quad \text{switch}(\Psi): \\
\quad \quad \text{true} : Sat_{\text{fair}}(\Psi) := S; \\
\quad \quad a : Sat_{\text{fair}}(\Psi) := \{ s \in S \mid a \in L(s) \}; \\
\quad \quad a \land a' : Sat_{\text{fair}}(\Psi) := \{ s \in S \mid a, a' \in L(s) \}; \\
\quad \quad \neg a : Sat_{\text{fair}}(\Psi) := \{ s \in S \mid a \not\in L(s) \}; \\
\quad \quad \exists \Box a : Sat_{\text{fair}}(\Psi) := Sat(\exists \Box (a \land a_{\text{fair}})); \\
\quad \quad \exists (a \lor a') : Sat_{\text{fair}}(\Psi) := Sat(\exists (a \lor (a' \land a_{\text{fair}}))); \\
\quad \quad \exists \Box a : \text{compute } Sat_{\text{fair}}(\exists \Box a) \\
\quad \text{end switch} \\
\quad \text{replace all occurrences of } \Psi \text{ (in } \Phi \text{) by the fresh atomic proposition } a_{\Psi} \\
\quad \text{forall } s \in Sat_{\text{fair}}(\Psi) \text{ do } L(s) := L(s) \cup \{ a_{\Psi} \} \od \\
\text{forall } s \in Sat_{\text{fair}}(\Psi) \text{ do } L(s) := L(s) \cup \{ a_{\Psi} \} \od \\
\text{return } I \subseteq Sat_{\text{fair}}(\Phi)
\]
Characterising $Sat_{fair}(\exists \Box a)$

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

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

1. $s_0 = s$ and $s_n = s'_0 = s'_r$
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. $Sat(a_i) \cap \{ s'_1, \ldots, s'_r \} = \emptyset$ or $Sat(b_i) \cap \{ s'_1, \ldots, s'_r \} \neq \emptyset$ for $0 < i \leq k$

Proof.

Next slide.
Proof
Computing $Sat_{fair}(\exists \Box a)$

- Consider only state $s$ if $s \models a$, otherwise eliminate $s$
  - consider $TS[a] = (S', Act, \to', I', AP, L')$ with $S' = Sat(a)$,
  - $\to' = \to \cap (S' \times Act \times S')$, $I' = I \cap S'$, and $L'(s) = L(s)$ for $s \in S'$
  - each infinite path fragment in $TS[a]$ satisfies $\Box a$

---

This is not necessarily an SCC (a maximal strongly-connected set).
Computing $Sat_{fair}(\exists \Box a)$

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

- Let $fair = \bigwedge_{0<i\leq k} (\Box \Diamond a_i \rightarrow \Box \Diamond b_i)$

\[ ^2 \text{This is not necessarily an SCC (a maximal strongly-connected set).} \]
Computing $Sat_{fair}(\exists \square a)$

- Consider only state $s$ if $s \models a$, otherwise eliminate $s$
  - consider $TS[a] = (S', Act, \rightarrow', l', AP, L')$ with $S' = Sat(a)$,
  - $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $l' = l \cap S'$, and $L'(s) = L(s)$ for $s \in S'$
  - each infinite path fragment in $TS[a]$ satisfies $\square a$

- Let $fair = \bigwedge_{0<i\leq k} (\square \Diamond a_i \rightarrow \square \Diamond b_i)$

- $s \models_{fair} \exists \square a$ iff $s$ can reach a strongly connected node-set\(^2\) $D$ in $TS[a]$ with:
  \[ D \cap Sat(a_i) = \emptyset \quad \text{or} \quad D \cap Sat(b_i) \neq \emptyset \quad \text{for} \quad 0 < i \leq k \quad (*) \]

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

---

\(^2\)This is not necessarily an SCC (a maximal strongly-connected set).
Example

Computing $\text{Sat}_{\text{fair}}(\exists \Box a)$ by analysing the digraph $G_a$ of $TS[a]$. 

\[ T \]

\[ \text{digraph } G_a \]
Example

\[ \text{fair} = (\Box \Diamond b_1 \rightarrow \Box \Diamond c_1) \land (\Box \Diamond b_2 \rightarrow \Box \Diamond c_2) \]

\[ s_0 \models_{\text{fair}} \exists \Box a \quad \text{as} \quad s_0 \ s_1 \ s_2 \ s_1 \ s_2 \ \ldots \models_{\text{LTL}} \text{fair} \]

\[ \text{Sat}_{\text{fair}}(\exists \Box a) = \{ s_0, s_1, s_2, s_3 \} \]
∃□a under Unconditional Fairness

Let \( u_{fair} = \bigwedge_{0<i\leq k} \Box \Diamond b_i \)

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

\[ C \cap Sat(b_i) \neq \emptyset \quad \text{for all } 0 < i \leq k \]
∃□a under Unconditional Fairness

Let $ufair = \bigwedge_{0<i\leq k} \Box \Diamond b_i$

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

$$C \cap Sat(b_i) \neq \emptyset$$

for all $0 < i \leq k$

It now follows:

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

$\Rightarrow T$ can be determined by a depth-first search procedure
Example

digraph $G_a$

fairness assumption:
$$\text{fair} = \square \Diamond c_1 \land \square \Diamond c_2$$

$s \not\models_{\text{fair}} \exists \square a$
∃□a Under One Strong Fairness Constraint

- $sfair = □◇ a_1 \rightarrow □◇ b_1$, i.e., $k=1$
∃□a Under One Strong Fairness Constraint

- $sfair = \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$, 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. $C \cap Sat(b_1) \neq \emptyset$, or
  2. $D \cap Sat(a_1) = \emptyset$, for some non-trivial SCC $D$ in $C$
$\exists \Box a$ Under One Strong Fairness Constraint

- $sfair = \Box \Diamond a_1 \rightarrow \Box \Diamond b_1$, 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. $C \cap \text{Sat}(b_1) \neq \emptyset$, or
  2. $D \cap \text{Sat}(a_1) = \emptyset$, for some non-trivial SCC $D$ in $C$

- $D$ is a non-trivial SCC in the graph that is obtained from $C[\neg a_1]$
∃□a Under One Strong Fairness Constraint

- \( sfair = □◇ a_1 → □◇ b_1 \), i.e., \( k=1 \)

- \( s ⊨_{sfair} ∃□a \) iff \( C \) is a non-trivial SCC in \( TS[a] \) reachable from \( s \) with:
  1. \( C \cap Sat(b_1) \neq ∅ \), 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):
  \[ s ⊨_{sfair} ∃□a \] if and only if \( Reach_{TS[a]}(s) \cap T \neq ∅ \)
∃□a Under One Strong Fairness Constraint

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

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

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

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

   \[ s \models_{sfair} ∃□a \text{ if and only if } Reach_{TS[a]}(s) \cap T \neq \emptyset \]

For several strong fairness constraints ($k > 1$), this is applied recursively. $T$ is determined by standard graph analysis (DFS)
Example: One Strong Fairness Constraint

\[ \text{fair} = \square \Diamond b \rightarrow \square \Diamond c \]

digraph $G_a$

nontrivial SCC $C$ of $G_a$ with $C \cap \text{Sat}(c) \neq \emptyset$
Example: One Strong Fairness Constraint

$$\text{fair} = \Box \Diamond b \rightarrow \Box \Diamond c$$

digraph $G_a$

- $\{c\}$
- $\{b\}$

strongly connected node-set $D$ of $G_a$ with $D \cap \text{Sat}(b) = \emptyset$
Example: One Strong Fairness Constraint

\[ \text{fair} = \Box \Diamond b \rightarrow \Box \Diamond c \]

digraph \( G_a \)

nontrivial SCC \( C \) of \( G_a \) that contains a nontrivial SCC \( D \) of \( G_a|_c \ \backslash \ SAT(b) \)
Example: Two Strong Fairness Constraints

\[ \text{fair} = (\Box \Diamond b_1 \rightarrow \Box \Diamond c_1) \land (\Box \Diamond b_2 \rightarrow \Box \Diamond c_2) \]

digraph \; G_a

first SCC: \; C_1 \cap Sat(c_2) = \emptyset

analyze \; C_1 \setminus Sat(b_2) \; \text{w.r.t.} \; \Box \Diamond b_1 \rightarrow \Box \Diamond c_1
Example: Two Strong Fairness Constraints

\[ \text{fair} = (\Box \Diamond b_1 \rightarrow \Box \Diamond c_1) \land (\Box \Diamond b_2 \rightarrow \Box \Diamond c_2) \]

second SCC: \[ C_2 \cap \text{Sat}(c_1) = \emptyset \]

analyze \[ C_2 \setminus \text{Sat}(b_1) \text{ w.r.t. } \Box \Diamond b_2 \rightarrow \Box \Diamond c_2 \]
Algorithm

compute the SCCs of the digraph $G_a$;

$T := \emptyset$;

FOR ALL nontrivial SCCs $C$ of $G_a$ DO

    IF $\text{CheckFair}(C, \ldots)$ THEN $T := T \cup C$ FI

OD

$\text{Sat}_{\text{fair}}(\exists \square a) := \{s \in S : \text{Reach}_{G_a}(s) \cap T \neq \emptyset\}$

backward search from $T$

CheckFair is a recursive procedure over the $k$ strong fairness constraints

Basically an SCC analysis per fairness constraint. Time complexity: $O(|TS| \cdot |\text{fair}|)$. 

Joost-Pieter Katoen
Lecture#14
CheckFair Algorithm (for completeness)

pseudo code for \texttt{CheckFair}(C, k, \bigwedge_{1 \leq i \leq k} (\Box ◻ b_i \rightarrow ◻ ◻ c_i))

IF \forall i \in \{1, ..., k\}. C \cap \text{Sat}(c_i) \neq \emptyset \ THEN \text{ return "true" FI} \\
choose j \in \{1, ..., k\} \ with \ C \cap \text{Sat}(c_j) = \emptyset; \\
remove all states in \text{Sat}(b_j); \\
IF the resulting graph G \ is acyclic \ THEN \text{ return "false" FI} \\
FOR ALL nontrivial SCCs D of G DO \\
\text{IF } \texttt{CheckFair}(D, k-1, \bigwedge_{i \neq j} (\Box ◻ b_i \rightarrow ◻ ◻ c_i)) \\
THEN \text{ return "true"} \\
OD \\
\text{return "false"} \\

\textbf{time complexity:} \ O(\text{size}(C) \cdot k)
Time complexity

The CTL model-checking problem under fairness assumption \textit{fair} can be solved in $O(|\Phi| \cdot |TS| \cdot |\text{fair}|)$.

**Proof.**

Follows from the complexity $O(|\Phi| \cdot |TS|)$ of CTL model checking.
Overview

1. The Relevance of Fairness
2. Fairness Assumptions
3. Fairness and Safety Properties
4. LTL Model Checking Under Fairness
5. CTL Fairness Assumptions
6. CTL Model Checking Under Fairness
7. Summary
## Model Checking Complexity

### Summary

<table>
<thead>
<tr>
<th>Model Checking Algorithmic Complexity with Fairness</th>
<th>CTL</th>
<th>LTL</th>
<th>CTL*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PTIME</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>TS</td>
<td>\cdot</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>TS</td>
<td>\cdot</td>
</tr>
</tbody>
</table>

All theoretical complexity indications are complete.
Summary

- Fairness constraints rule out “unreasonable” computations
- Fairness assumptions are conjunctions of fairness constraints
- Fair LTL model checking is reduced to standard LTL model checking
- CTL fairness constraints are fair “LTL”-formulas over CTL state-formulas
- Fair CTL model checking is standard CTL model checking . . .
- . . . plus a dedicated procedure for $\exists\Box a$
- Complexity of fair CTL model checking is $O(|TS| \cdot |\Phi| \cdot |fair|)$
Next Lecture

Thursday December 12, 10:30

No Lecture on Friday December 6