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

Overview

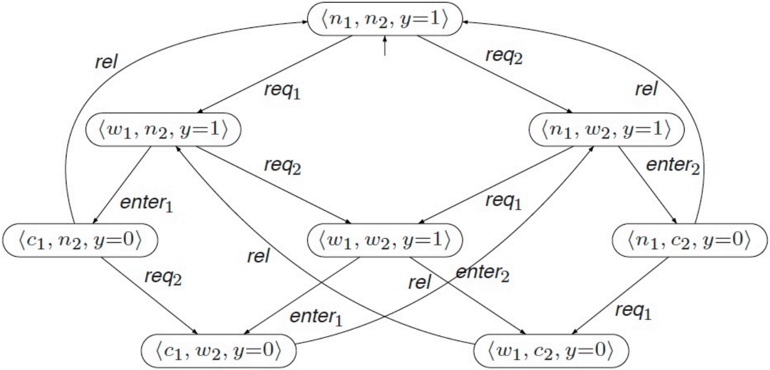
Does This Multi-Threaded Program Terminate?

$$\mathsf{Inc} \,|||\, \mathsf{Reset}$$

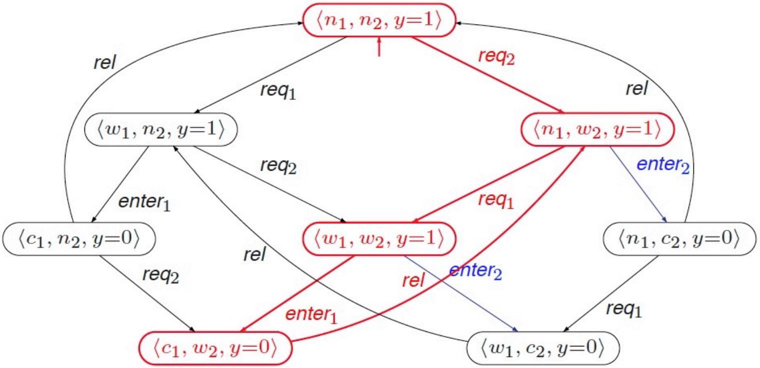
$$\begin{array}{rcl} & \textit{where} & \\ \textbf{thread}\ \mathsf{Inc} & = & \textbf{while}\ \langle\, x \geq 0\ \textbf{do}\ x := x+1 \,\rangle\ \textbf{od} \\ \textbf{thread}\ \mathsf{Reset} & = & x := -1 \end{array}$$

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
  - $\Rightarrow$ 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

- Starvation freedom is often considered under thread fairness
  - $\Rightarrow$ 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
  - $\Rightarrow$ 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 ∞ 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 ∞ 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

Fairness Constraints

- What is wrong with our examples? Nothing!
  - interleaving: not realistic as no processor is ∞ 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

Fairness

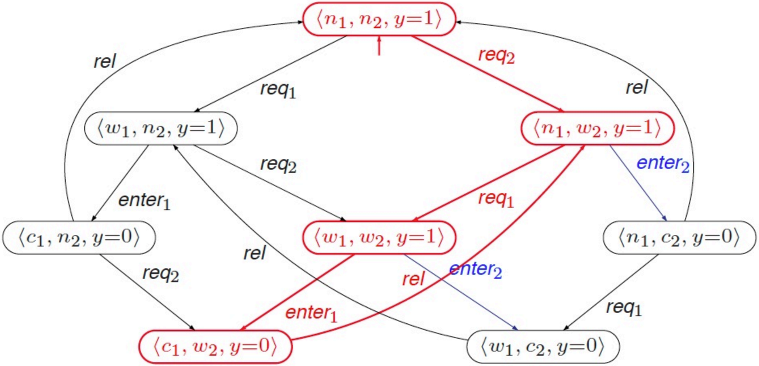
This program terminates assuming unconditional (thread) fairness:

$$\begin{array}{rcl} \textbf{thread}\ \mathrm{Inc} & = & \textbf{while}\ \langle\, x \geq 0\ \textbf{do}\ x := x+1 \,\rangle\ \textbf{od} \\ \textbf{thread}\ \mathrm{Reset} & = & x := -1 \end{array}$$

as thread Reset eventually will set x to -1

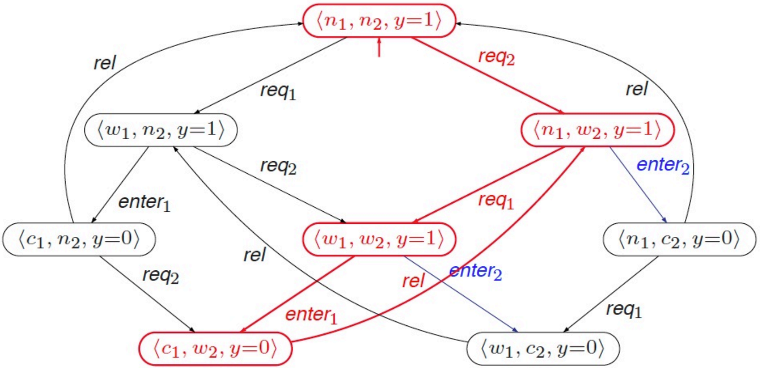
x is a shared integer variable that initially has value 0

Avoiding Starvation by Fairness



If the infinitely often enabled *enter*$_2$ action is not ignored infinitely often, thread two does not starve.

Avoiding Starvation by Fairness



Note that *enter*$_2$ is not enabled continuously during the run. Weak fairness this does not suffice.

Overview

LTL Fairness Constraints

Definition: 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 is of the form:

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

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

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

Φ stands for "... is enabled"; Ψ for "... is taken"

Relating Fairness Constraints

unconditional fair $\Rightarrow$ strong fair $\Rightarrow$ weak fair.

Fairness Assumptions

Definition: fairness assumption

An LTL fairness assumption is a conjunction of LTL fairness constraints.

The general format of fairness assumption *fair* is

$$\mathit{fair} \;=\; \mathit{ufair} \wedge \mathit{sfair} \wedge \mathit{wfair}$$

.

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

$$
\begin{aligned}
FairPaths_{fair}(s) &= \{\, \pi \in Paths(s) \mid \pi \models fair \,\} \\
FairTraces_{fair}(s) &= \{\, trace(\pi) \mid \pi \in FairPaths_{fair}(s) \,\}.
\end{aligned}
$$

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

$$\begin{aligned} FairPaths_{fair}(s) &= \{\, \pi \in Paths(s) \mid \pi \models fair \,\} \\ FairTraces_{fair}(s) &= \{\, trace(\pi) \mid \pi \in FairPaths_{fair}(s) \,\}. \end{aligned}$$

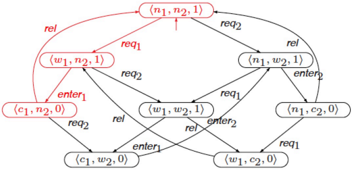
Definition: fair satisfaction relation

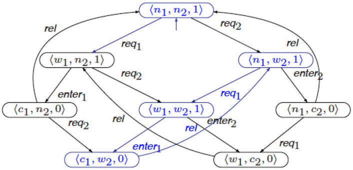
For LTL-formula φ, and LTL fairness assumption *fair*:

$$\begin{aligned} s \models_{fair} \varphi \quad &\text{if and only if} \quad \forall \pi \in FairPaths_{fair}(s).\, \pi \models \varphi \\ TS \models_{fair} \varphi \quad &\text{if and only if} \quad \forall s_0 \in I.\, s_0 \models_{fair} \varphi. \end{aligned}$$

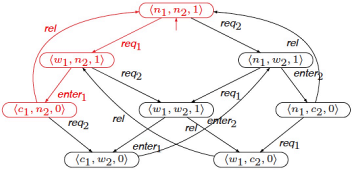
The relation $\models_{fair}$ is the fair satisfaction relation for LTL.

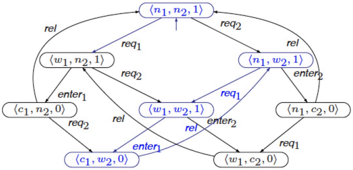
Example: Fair Runs and Fair Traces





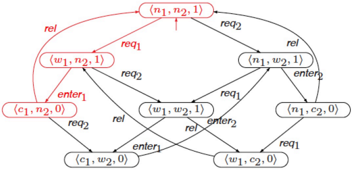
Example: Fair Runs and Fair Traces

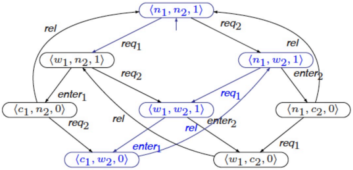




- Let Φ = "action $enter_2$ is enabled" and Ψ = "action $enter_2$ is taken"

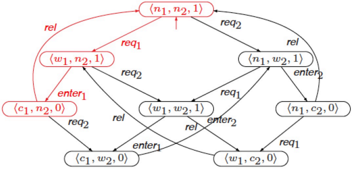
Example: Fair Runs and Fair Traces

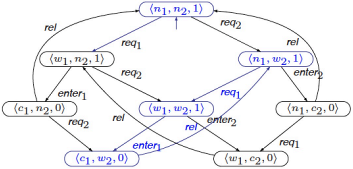




- Let Φ = "action $enter_2$ is enabled" and Ψ = "action $enter_2$ is taken"
- Run $\langle n_1, n_2, 1\rangle \xrightarrow{req_1} \langle w_1, n_2, 1\rangle \xrightarrow{enter_1} \langle c_1, n_2, 0\rangle \xrightarrow{rel} \langle n_1, n_2, 1\rangle \xrightarrow{req_1} \ldots$
  - ... is not unconditionally fair
  - ... but strongly fair, as action $enter_2$ is never enabled along the run

Example: Fair Runs and Fair Traces



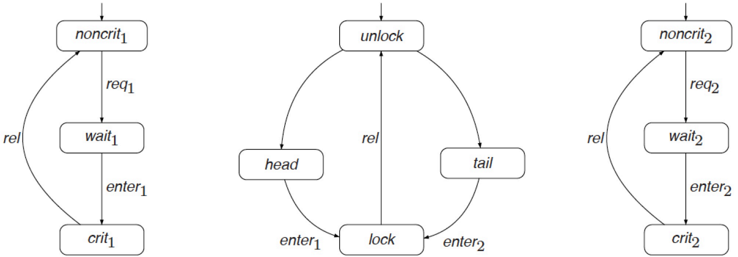


- Let Φ = "action $enter_2$ is enabled" and Ψ = "action $enter_2$ is taken"

- Run $\langle n_1, n_2, 1\rangle \xrightarrow{req_1} \langle w_1, n_2, 1\rangle \xrightarrow{enter_1} \langle c_1, n_2, 0\rangle \xrightarrow{rel} \langle n_1, n_2, 1\rangle \xrightarrow{req_1} \ldots$
  - ... is not unconditionally fair
  - ... but strongly fair, as action $enter_2$ is never enabled along the run

- Run $\langle n_1, n_2, 1\rangle \xrightarrow{req_2} \langle n_1, w_2, 1\rangle \xrightarrow{req_1} \langle w_1, w_2, 1\rangle \xrightarrow{enter_1} \langle c_1, w_2, 0\rangle \xrightarrow{rel} \ldots$
  - ... is not strongly fair as $enter_2$ is ∞ 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



$$TS_1 \,||\, Arbiter \,||\, TS_2 \;\not\models\; \Box\Diamond\, crit_1$$

But: $TS_1 \,||\, Arbiter \,||\, TS_2 \;\models_{fair}\; \Box\Diamond\, crit_1 \;\wedge\; \Box\Diamond\, crit_2$

with *fair* $= \Box\Diamond\, head \;\wedge\; \Box\Diamond\, tail$

Overview

Realisable Fairness

Definition: realisable fairness

Fairness assumption *fair* is realisable for transition system *TS* if for any reachable state s: $FairPaths_{fair}(s) \neq \varnothing$.

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 π, it holds

1. all suffixes of π are fair too.
2. any finite path extended by π is fair.

Proof.

Rather straightforward. $\square$

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.

□

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 φ, and
3. an LTL fairness assumption *fair*

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

Fair LTL Model Checking

For transition system TS, LTL formula φ and LTL fairness assumption *fair*:

$$\underbrace{TS \models_{fair} \varphi}_{\text{fair LTL model checking}} \quad \text{if and only if} \quad \underbrace{TS \models \left(fair \rightarrow \varphi \right)}_{\text{LTL model checking}}$$

Fair LTL Model Checking

For transition system TS, LTL formula φ and LTL fairness assumption *fair*:

$$\underbrace{TS \models_{fair} \varphi}_{\text{fair LTL model checking}} \quad \text{if and only if} \quad \underbrace{TS \models (fair \rightarrow \varphi)}_{\text{LTL model checking}}$$

The fair LTL model-checking problem for φ 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 φ and LTL fairness assumption *fair*:

$$\underbrace{TS \models_{fair} \varphi}_{\text{fair LTL model checking}} \quad \text{if and only if} \quad \underbrace{TS \models (fair \rightarrow \varphi)}_{\text{LTL model checking}}$$

The fair LTL model-checking problem for φ 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

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? $\Rightarrow$ 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

Rules of thumb:

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

Overview

Fairness Constraints in CTL

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

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

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

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

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: strong fairness constraint $\Box\Diamond\, b \rightarrow \Box\Diamond\, c$, i.e., $\Diamond\Box\neg b \vee \Diamond\Box\, c$
  cannot be expressed in CTL as persistence properties are not in CTL

Fairness Constraints in CTL

- For LTL it holds: $TS \models_{fair} \varphi$ 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: strong fairness constraint $\Box\Diamond\, b \rightarrow \Box\Diamond\, c$, i.e., $\Diamond\Box\,\neg b \vee \Diamond\Box\, c$
  cannot be expressed in CTL as persistence properties are not in CTL
- Solution: change the semantics of CTL by ignoring unfair paths

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 Φ_i and Ψ_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 Φ_i and Ψ_i (for $0 < i \leq k$) are CTL state-formulas over AP.

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

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

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 Φ_i and Ψ_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 assumption

A CTL fairness assumption is a conjunction of *ufair*, *sfair* and *wfair*.

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 Φ_i and Ψ_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 assumption

A CTL fairness assumption is a conjunction of *ufair*, *sfair* and *wfair*.

A CTL fairness constraint is an LTL formula over CTL state formulas.

Φ_i and Ψ_i are interpreted by the standard (unfair) CTL semantics

Semantics of Fair CTL

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

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

$$
\begin{array}{ll}
\pi \models_{fair} \bigcirc \Phi & \text{iff } \pi[1] \models_{fair} \Phi \\
\pi \models_{fair} \Phi \,\mathsf{U}\, \Psi & \text{iff } \big(\exists j \geq 0.\, \pi[j] \models_{fair} \Psi \text{ and } (\forall\, 0 \leq i < j.\, \pi[i] \models_{fair} \Phi)\big)
\end{array}
$$

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*, the satisfaction set $Sat_{fair}(\Phi)$ is defined by:

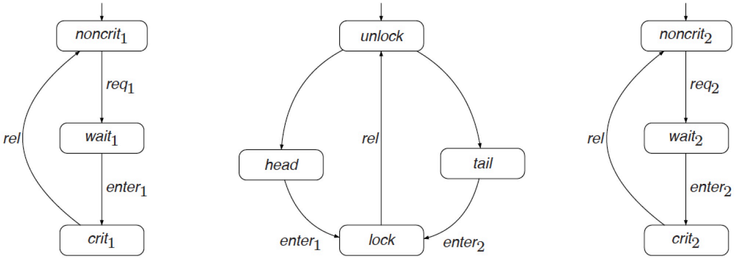
$$Sat_{fair}(\Phi) \; = \; \{ \, s \in S \mid s \models_{fair} \Phi \, \}$$

- TS satisfies CTL-formula Φ iff Φ 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

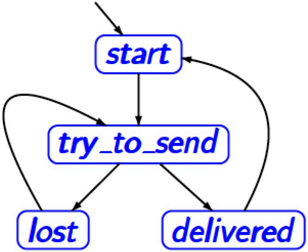


$$TS_1 \parallel Arbiter \parallel TS_2 \;\not\models\; (\forall\Box\forall\Diamond\, crit_1) \wedge (\forall\Box\forall\Diamond\, crit_2)$$

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

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

Example



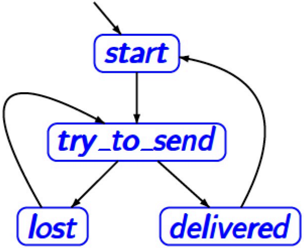
$\Phi = \forall\Box\forall\Diamond \mathit{start}$

$\mathcal{T} \models_{\mathit{ufair}} \Phi \quad \checkmark$

unconditional fairness: $\mathit{ufair} = \Box\Diamond\, \exists\bigcirc \mathit{start}$

$$\begin{aligned} \mathit{Sat}(\exists\bigcirc \mathit{start}) &= \{\mathit{delivered}\} \\ \mathit{ufair} &\,\widehat{=}\, \Box\Diamond \mathit{delivered} \end{aligned}$$

Example



$\Phi = \forall\Box\forall\Diamond start$

$\mathcal{T} \models_{ufair} \Phi$ $\checkmark$

$\mathcal{T} \models_{wfair} \Phi$ wrong

unconditional fairness: $ufair = \Box\Diamond\, \exists\bigcirc start$

weak fairness: $wfair = \Diamond\Box\, \exists\bigcirc delivered \rightarrow \Box\Diamond delivered$

$Sat(\exists\bigcirc delivered) = \{try\_to\_send\}$

$wfair \mathrel{\widehat{=}} \Diamond\Box try\_to\_send \rightarrow \Box\Diamond delivered$

Overview

The Fair CTL Model-Checking Problem

Given:

1. a finite transition system TS
2. an CTL state-formula[1] Φ, and
3. a CTL fairness assumption *fair*

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

[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] Φ, 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 Φ_i and Ψ_i (for $0 < i \leq k$) are CTLstate-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 Φ_i and Ψ_i (for $0 < i \leq k$) are CTLstate-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 $s \in Sat(\Phi_i)$ (not $Sat_{fair}(\Phi_i)$)
  - ... $b_i \in L(s)$ if and only if $s \in Sat(\Psi_i)$ (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 Φ_i and Ψ_i (for $0 < i \leq k$) are CTLstate-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 $s \in Sat(\Phi_i)$ (not $Sat_{fair}(\Phi_i)$)
  - ... $b_i \in L(s)$ if and only if $s \in Sat(\Psi_i)$ (not $Sat_{fair}(\Psi_i)$)

- For unconditional and weak fairness this goes similarly

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 Φ_i and Ψ_i (for $0 < i \leq k$) are CTLstate-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 $s \in Sat(\Phi_i)$ (not $Sat_{fair}(\Phi_i)$)
  - ... $b_i \in L(s)$ 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 $j \geq 0$ 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. $s \models_{fair} \exists \bigcirc a$ iff $\exists s' \in Post(s)$ with $s' \models a$ and $FairPaths_{fair}(s') \neq \varnothing$
2. $s \models_{fair} \exists(a \,\mathsf{U}\, a')$ if and only if there exists a finite path fragment

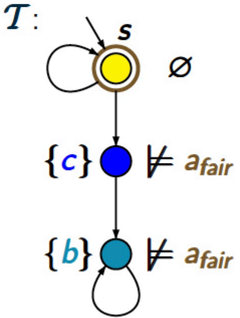
$$s_0\, s_1\, s_2 \ldots s_{n-1}\, s_n \in Paths^*(s) \quad \text{with } n \geq 0$$

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

Proof.

On the black board. □

Example



$$s \not\models_{fair} \exists(\neg b \,\mathsf{U}\, c)$$

$$\Uparrow$$

$$s \not\models \exists(\neg b \,\mathsf{U}\, (c \wedge a_{fair}))$$

$$\Uparrow$$

$$Sat(c \wedge a_{fair}) = \varnothing$$

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

$$\boxed{\mathcal{T} \models \exists(\neg b \,\mathsf{U}\, c), \quad \text{but } \mathcal{T} \not\models_{fair} \exists(\neg b \,\mathsf{U}\, c)}$$

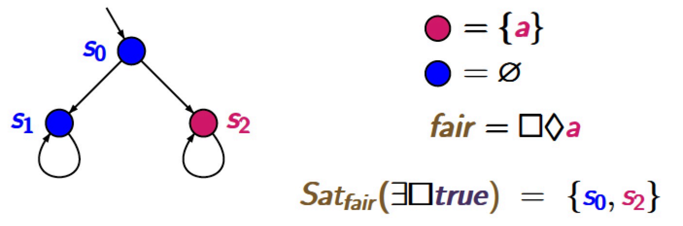
Fair Path Existence

$$FairPaths_{fair(s)} \neq \varnothing \quad \text{if and only if} \quad s \models_{fair} \exists \Box \text{true}.$$

Fair Path Existence

$$FairPaths_{fair(s)} \neq \varnothing \quad \text{if and only if} \quad s \models_{fair} \exists\Box\text{true}.$$

Example



Basic Model-Checking Algorithm for Fair CTL

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

Basic Model-Checking Algorithm for Fair CTL

- Determine $Sat_{fair}(\exists\Box\text{true}) \;=\; \{\, s \in S \mid FairPaths_{fair}(s) \neq \varnothing \,\}$

- 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 $Sat_{fair}(\exists\Box\text{true}) \;=\; \{\, s \in S \mid FairPaths_{fair}(s) \neq \varnothing \,\}$

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

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

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

Basic Model-Checking Algorithm for Fair CTL

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

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

- 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 $Sat_{fair}(\exists\Box a)$ 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 $Sat_{fair}(\exists\Box a)$ for $a \in AP$.

As $\exists\Box\text{true}$ is a special case of $\exists\Box a$,
an algorithm for $Sat_{fair}(\exists\Box a)$ can be used for $Sat_{fair}(\exists\Box\text{true})$

Basic Fair CTL Algorithm

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

compute $\mathit{Sat}_{\mathit{fair}}(\exists\Box\mathrm{true}) = \{ s \in S \mid \mathit{FairPaths}(s) \neq \varnothing \}$
forall $s \in \mathit{Sat}_{\mathit{fair}}(\exists\Box\mathrm{true})$ **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 $I \subseteq \mathit{Sat}_{\mathit{fair}}(\Phi)$

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

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

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. $Sat(a_i) \cap \{ s'_1, \ldots, s'_r \} = \varnothing$ or $Sat(b_i) \cap \{ s'_1, \ldots, s'_r \} \neq \varnothing$ 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 $TS[a] = (S', Act, \rightarrow', I', AP, L')$ 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 $\Box\, a$

[2] 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 $s \in S'$

  $\Rightarrow$ 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] 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 $s \in S'$
  - $\Rightarrow$ each infinite path fragment in $TS[a]$ satisfies $\Box a$

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

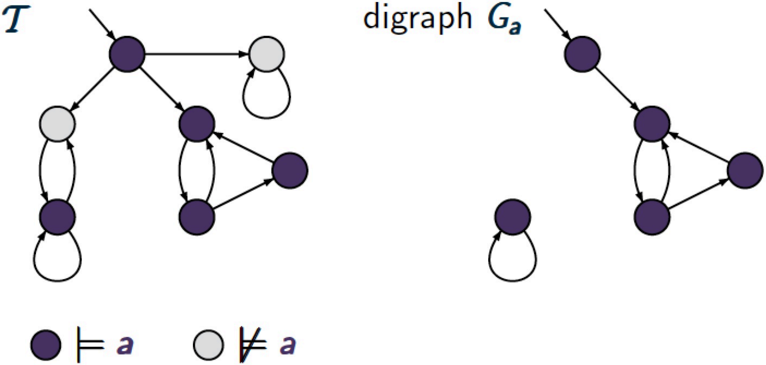
- $s \models_{fair} \exists\Box a$ iff s can reach a strongly connected node-set[2] D in $TS[a]$ with:

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

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

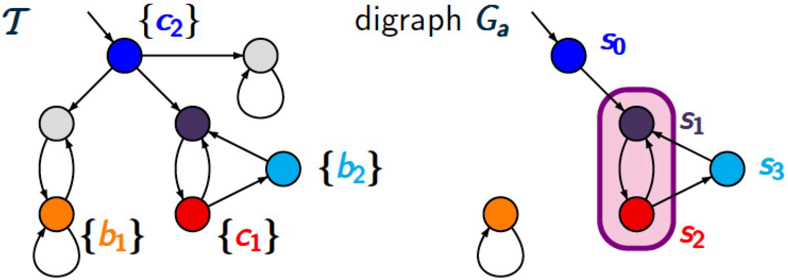
[2] This is not necessarily an SCC (a maximal strongly-connected set).

Example



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

Example



$$fair = (\Box\Diamond b_1 \rightarrow \Box\Diamond c_1) \wedge (\Box\Diamond b_2 \rightarrow \Box\Diamond c_2)$$

$s_0 \models_{fair} \exists\Box a$ as $s_0\, s_1\, s_2\, s_1\, s_2 \ldots \models_{LTL} fair$

$Sat_{fair}(\exists\Box a) = \{s_0, s_1, s_2, s_3\}$

$\exists\Box a$ under Unconditional Fairness

$$\text{Let} \quad \mathit{ufair} \; = \bigwedge_{0<i\leq k} \Box\Diamond\, b_i$$

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

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

$\exists\Box a$ under Unconditional Fairness

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

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

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

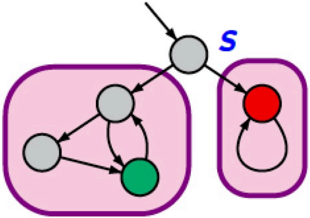
It now follows:

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

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

Example

digraph G_a



fairness assumption:

$fair = \Box\Diamond c_1 \wedge \Box\Diamond c_2$

$s \not\models_{fair} \exists\Box a$

$\exists\Box a$ Under One Strong Fairness Constraint

- $sfair = \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$, i.e., k=1

$\exists\Box a$ 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. $C \cap Sat(b_1) \neq \varnothing$, or
  2. $D \cap Sat(a_1) = \varnothing$, 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. $C \cap Sat(b_1) \neq \varnothing$, 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 $C[\neg a_1]$

$\exists\Box a$ 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. $C \cap Sat(b_1) \neq \varnothing$, 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):

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

$\exists\Box a$ 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. $C \cap Sat(b_1) \neq \varnothing$, 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):

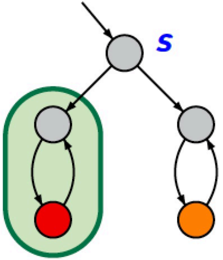
$$s \models_{sfair} \exists\Box a \quad \text{if and only if} \quad Reach_{TS[a]}(s) \cap T \neq \varnothing$$

For several strong fairness constraints ($k > 1$), this is applied recursively
T is determined by standard graph analysis (DFS)

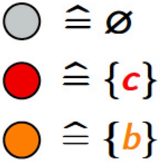
Example: One Strong Fairness Constraint

$fair = \Box\Diamond b \rightarrow \Box\Diamond c$

digraph G_a



$s \models_{fair} \exists\Box a$

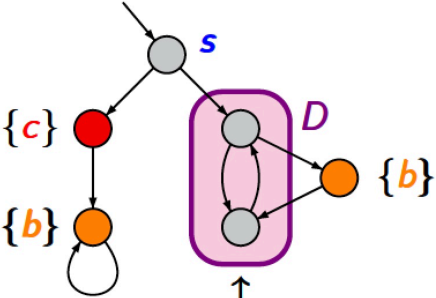


nontrivial SCC C of G_a with $C \cap Sat(c) \neq \varnothing$

Example: One Strong Fairness Constraint

$fair = \Box\Diamond b \rightarrow \Box\Diamond c$

digraph G_a



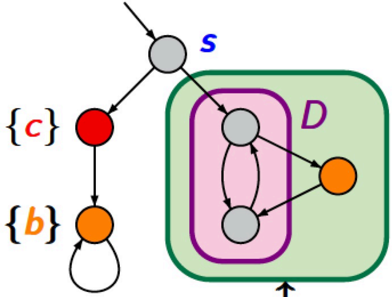
$s \models_{fair} \exists\Box a$

strongly connected node-set D of G_a with
$D \cap Sat(b) = \varnothing$

Example: One Strong Fairness Constraint

$fair = \Box\Diamond b \rightarrow \Box\Diamond c$

digraph G_a



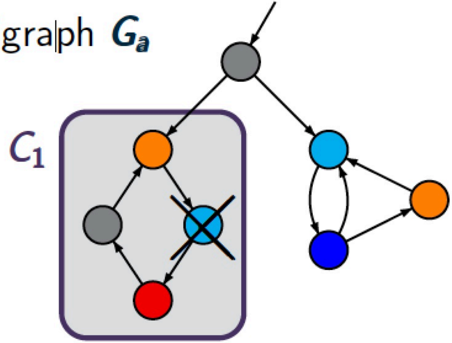
$s \models_{fair} \exists\Box a$

nontrivial SCC C of G_a that contains a nontrivial SCC D of $G_a|_C \setminus Sat(b)$

Example: Two Strong Fairness Constraints

$$fair = (\Box\Diamond b_1 \rightarrow \Box\Diamond c_1) \wedge (\Box\Diamond b_2 \rightarrow \Box\Diamond c_2)$$

digraph G_a



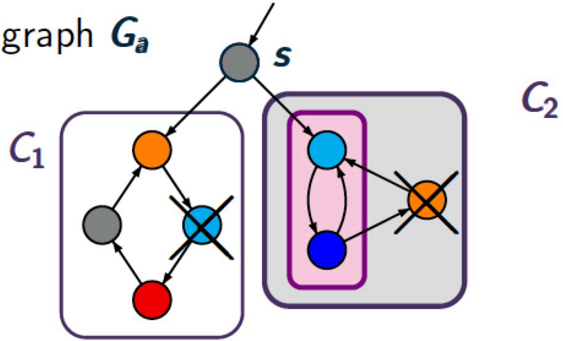
first SCC: $C_1 \cap Sat(c_2) = \varnothing$

analyze $C_1 \setminus Sat(b_2)$ w.r.t. $\Box\Diamond b_1 \rightarrow \Box\Diamond c_1$

Example: Two Strong Fairness Constraints

$$fair = (\Box\Diamond b_1 \rightarrow \Box\Diamond c_1) \ \wedge \ (\Box\Diamond b_2 \rightarrow \Box\Diamond c_2)$$

digraph G_a



second SCC: $C_2 \cap Sat(c_1) = \varnothing$

analyze $C_2 \setminus Sat(b_1)$ w.r.t. $\Box\Diamond b_2 \rightarrow \Box\Diamond c_2$

Algorithm

compute the SCCs of the digraph G_a;

$T := \varnothing$;

FOR ALL nontrivial SCCs C of G_a DO

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

OD

$Sat_{fair}(\exists\Box a) := \{ s \in S : \underbrace{Reach_{G_a}(s)}_{\text{backward search from } T} \cap T \neq \varnothing \}$

CheckFair is a recursive procedure over the k strong fairness constraints

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

CheckFair Algorithm (for completeness)

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

```
IF ∀i ∈ {1, ..., k}. C ∩ Sat(c_i) ≠ ∅ THEN return "true" FI
choose j ∈ {1, ..., k} with C ∩ Sat(c_j) = ∅;
remove all states in Sat(b_j);
IF the resulting graph G is acyclic THEN return "false" FI
FOR ALL nontrivial SCCs D of G DO
   IF CheckFair(D, k−1, ⋀_{i≠j} (□◊b_i→□◊c_i))
   THEN return "true"
OD
return "false"
```

time complexity:

$\mathcal{O}(\mathit{size}(C) \cdot k)$

Time complexity

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

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

	CTL	LTL	CTL*
model checking	PTIME	PSPACE	PSPACE
algorithmic complexity	$\|TS\| \cdot \|\Phi\|$	$\|TS\| \cdot \exp(\|\varphi\|)$	$\|TS\| \cdot \exp(\|\Phi\|)$
with fairness	$\|TS\| \cdot \|\Phi\| \cdot \|\mathit{fair}\|$	$\|TS\| \cdot \exp(\|\varphi\|+\|\mathit{fair}\|)$	$\|TS\| \cdot \exp(\|\Phi\|+\|\mathit{fair}\|)$

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 $\exists\square a$
- 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