Model Checking

Fairness

[Baier & Katoen, Chapter 3.5, 5.1.6, 6.5]

Joost-Pieter Katoen and Tim Quatmann

Software Modeling and Verification Group

RWTH Aachen, SoSe 2022

Overview

Overview

Does This Multi-Threaded Program Terminate?

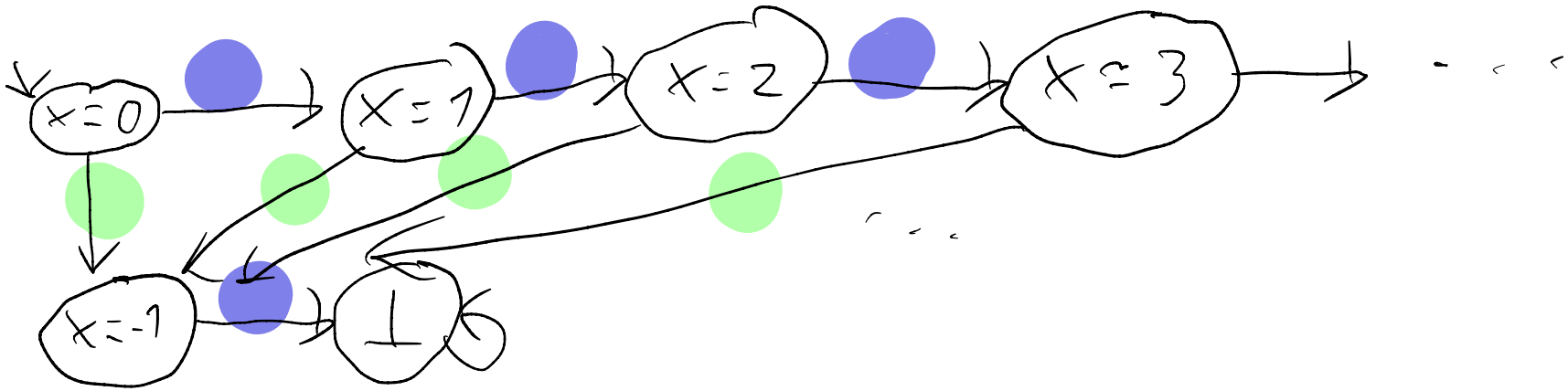
$$\text{Inc} \;|||\; \text{Reset}$$

where

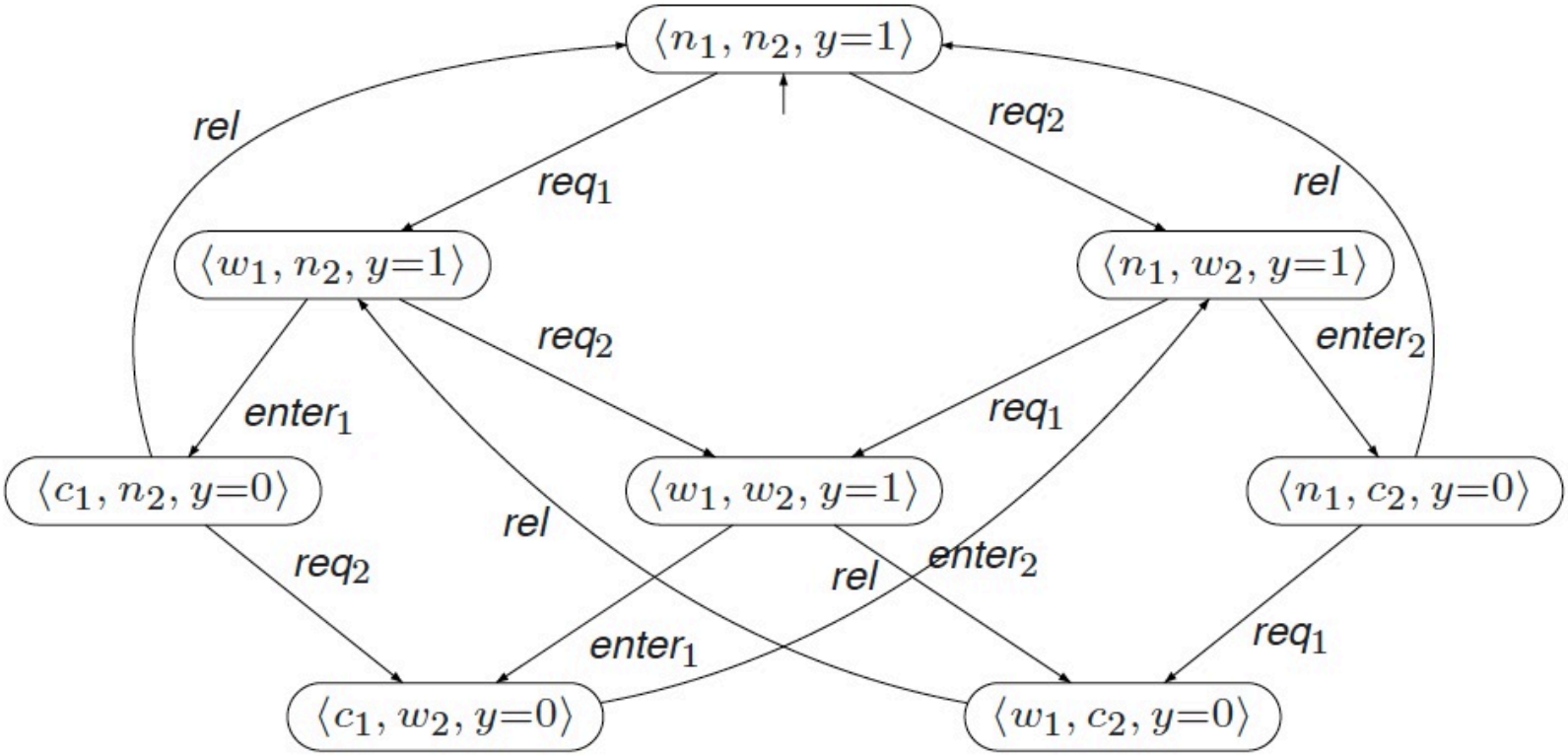
$$\textbf{thread}\ \text{Inc} \;=\; \textbf{while}\ \langle\, x \geq 0\ \textbf{do}\ x := x + 1 \,\rangle\ \textbf{od}$$

$$\textbf{thread}\ \text{Reset} \;=\; x := -1$$

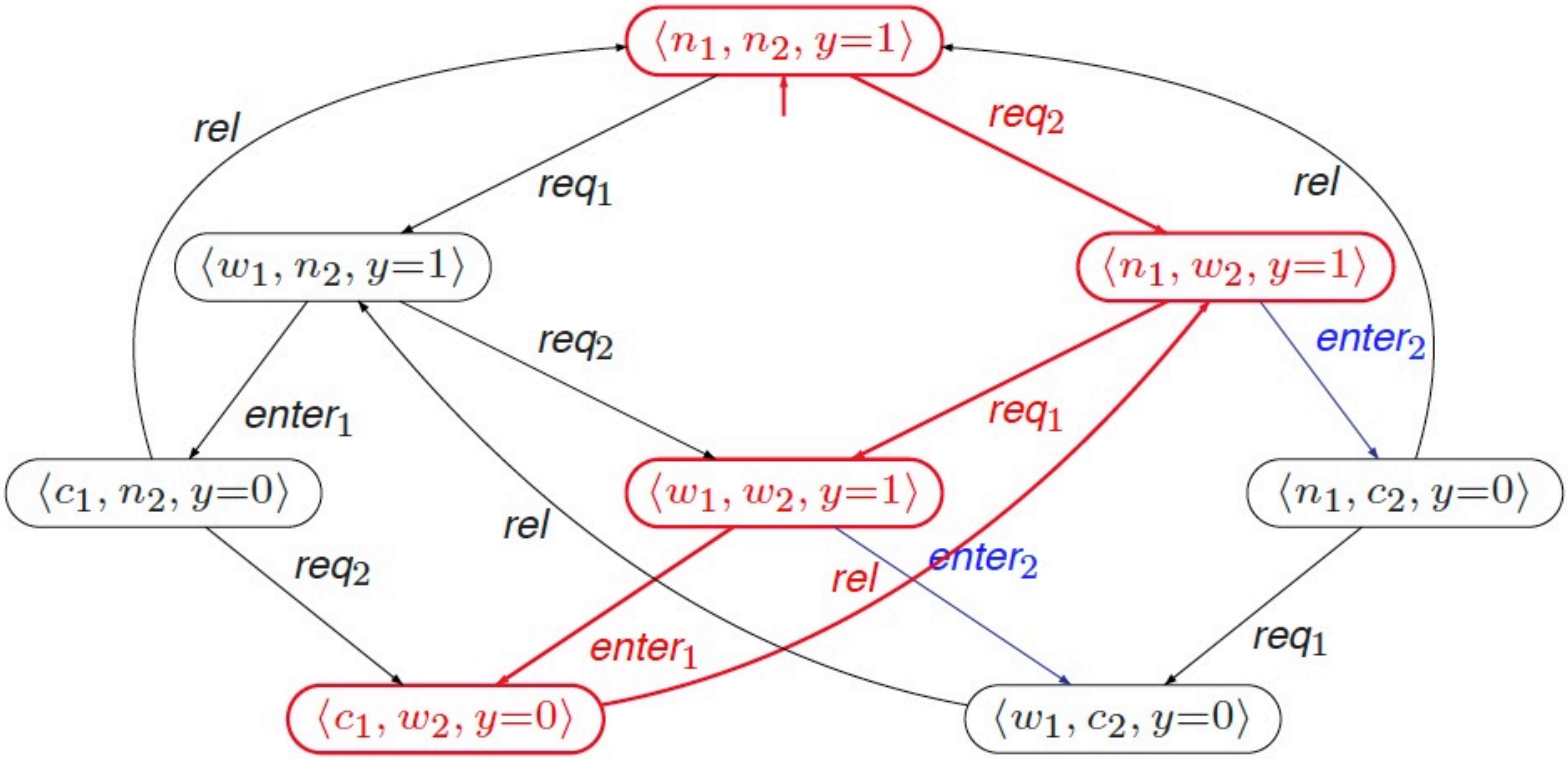
x is a shared integer variable that initially has value 0



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 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
- 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
- "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

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

where

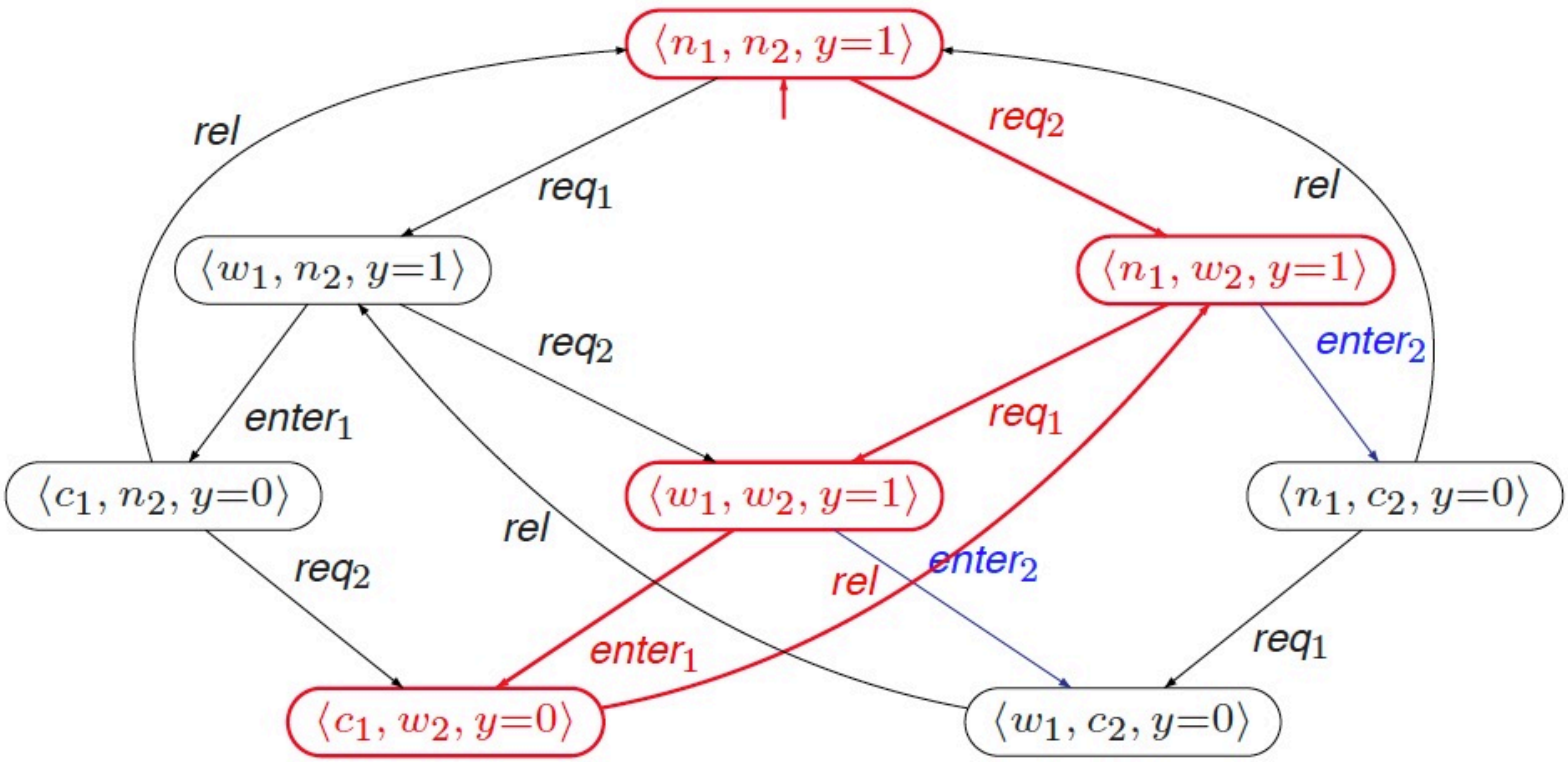
$$\textbf{thread}\ \text{Inc} = \textbf{while}\ \langle\, x \geq 0\ \textbf{do}\ x := x + 1 \,\rangle\ \textbf{od}$$

$$\textbf{thread}\ \text{Reset} = x := -1$$

x is a shared integer variable that initially has value 0

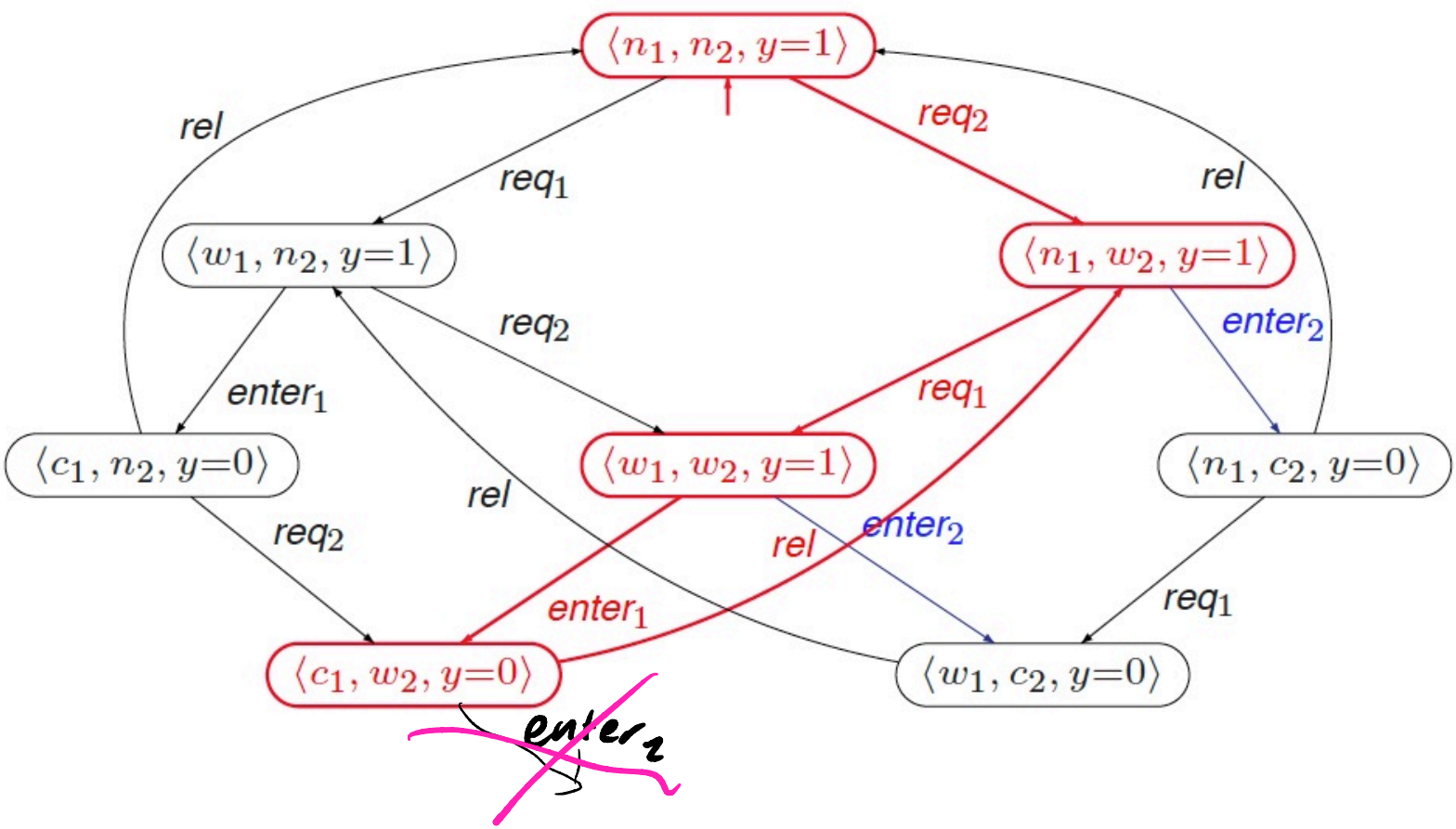
This program terminates assuming unconditional (thread) fairness as thread Reset eventually will set x to -1

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 thus 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:

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

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

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

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

$$fair = ufair \wedge sfair \wedge wfair$$

.

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

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

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

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

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

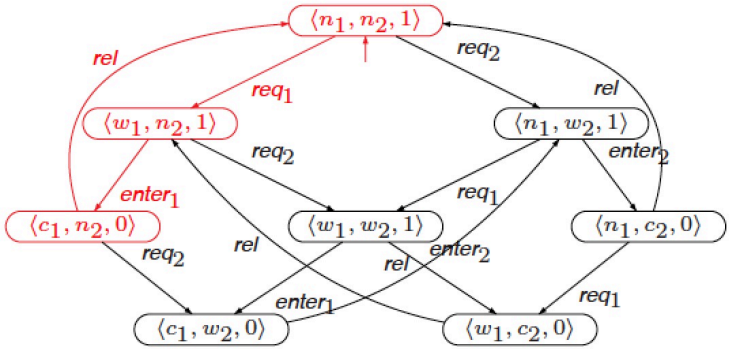
Definition: fair satisfaction relation

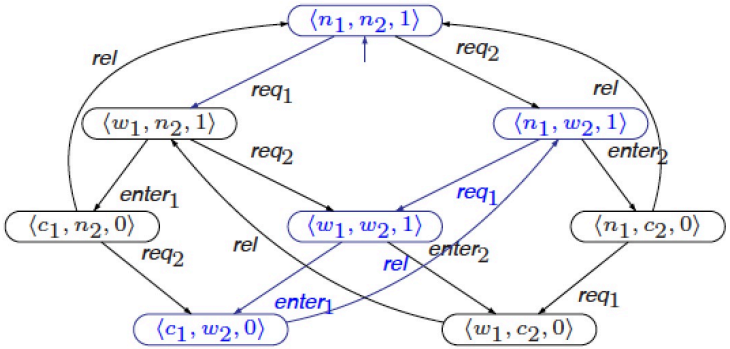
For LTL-formula φ, and LTL fairness assumption *fair*:

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

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

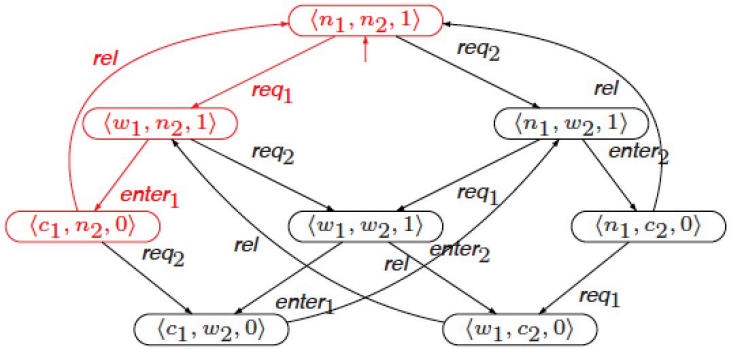
Example: Fair Runs and Fair Traces

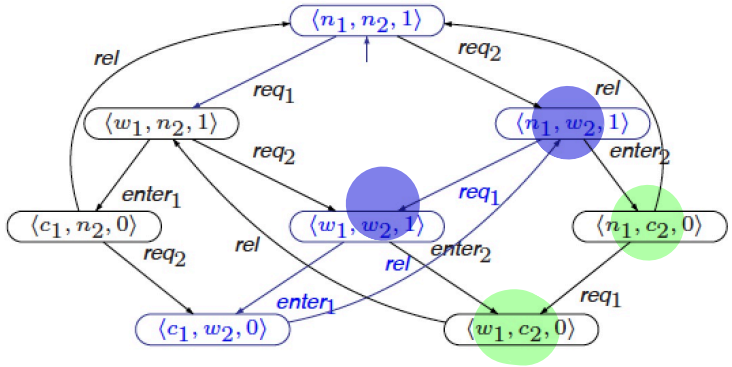




- Let Φ = "action $enter_2$ is enabled" and Ψ = "action $enter_2$ is taken"
- Run $\langle n_1, n_2, 1\rangle \xrightarrow{req_1} \langle w_1, n_2, 1\rangle \xrightarrow{enter_1} \langle c_1, n_2, 0\rangle \xrightarrow{rel} \langle n_1, n_2, 1\rangle \xrightarrow{req_1} \ldots$
  - ... is not unconditionally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

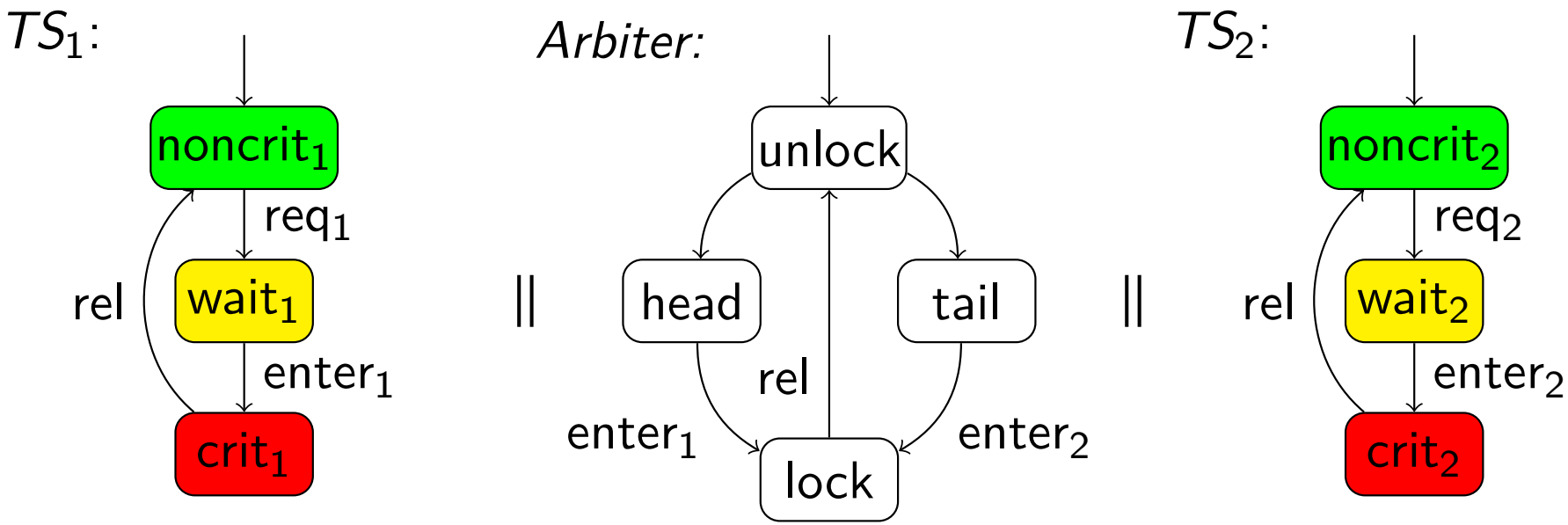
Example: Fair Runs and Fair Traces





- 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 as $enter_2$ is not always enabled along the run

Example: An Arbiter for Mutual Exclusion



$$TS_1 \parallel \mathit{Arbiter} \parallel TS_2 \not\models \Box\Diamond\, \mathit{crit}_1$$

But: $TS_1 \parallel \mathit{Arbiter} \parallel TS_2 \models_{\mathit{fair}} \Box\Diamond\, \mathit{crit}_1 \wedge \Box\Diamond\, \mathit{crit}_2$

with $\mathit{fair} = \Box\Diamond\, \mathit{head} \wedge \Box\Diamond\, \mathit{tail}$

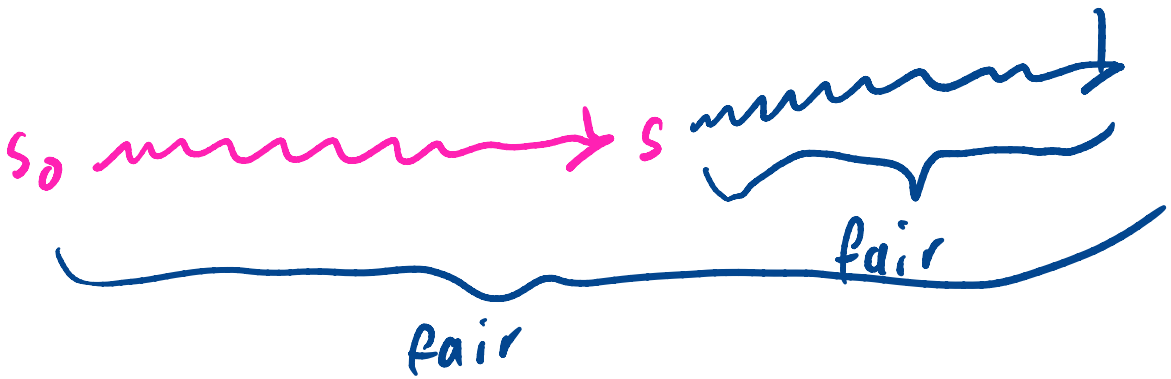
Overview

Realisable Fairness

Definition: realisable fairness

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

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 π, it holds

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

Proof.

Rather straightforward. □

It follows that

$$\pi \vDash \mathit{fair} \text{ iff } \pi[j..] \vDash \mathit{fair} \text{ for some } j \geq 0 \text{ iff } \pi[j..] \vDash \mathit{fair} \text{ for all } j \geq 0$$

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.

$\Longrightarrow$: $\forall\, s_0 \in I :\ FairTraces_{fair}(s_0) \subseteq Traces(s_0) \subseteq E_{safe}$

$\Longleftarrow$:

□

Proof for "$\Leftarrow$":

Let $TS \nvDash_{fair} E_{safe}$ for realisable fair.

To show: $TS \nvDash E_{safe}$

By contraposition: Assume there is

$\sigma \in Traces(TS) \setminus E_{safe}$

As $\sigma \notin E_{safe}$, there is a bad prefix $\hat{\sigma} \in pref(\sigma)$, i.e.,

$$P = \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \in pref(\sigma')\}$$

satisfies $P \cap E_{safe} = \emptyset$

Let $\hat{\pi} = s_0 \dots s_n \in Paths_{fin}(TS)$ with $trace(\hat{\pi}) = \hat{\sigma}$

Since fair is realisable, $FairPaths_{fair}(s_n) \neq \emptyset$

let $s_n s_{n+1} s_{n+2} \dots \in fair$ and consider

$\pi = s_0 \dots s_n s_{n+1} s_{n+2} \dots$

bad pref | fair

- $\pi \in fair$. From $TS \nvDash_{fair} E_{safe}$ we get $trace(\pi) \in E_{safe}$
- $\hat{\sigma} = trace(\hat{\pi})$ is a prefix of $trace(\pi)$
  $\rightsquigarrow trace(\pi) \in P$

↯ This contradicts $P \cap E_{safe} = \emptyset$ □

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.

$\Longrightarrow$: $\forall\, s_0 \in I :\ FairTraces_{fair}(s_0) \subseteq Traces(s) \subseteq E_{safe}$

$\Longleftarrow$:

□

Non-realisable fairness may harm safety properties.

Overview

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

Fair LTL Model Checking

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

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

Fair LTL Model Checking

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

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

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

This approach is not applicable to CTL (as we will discuss)

Which Fairness Notion?

- Fairness constraints aim to rule out "unreasonable" runs
- Too strong? ⇒ reasonable runs ruled out. Verification result:
  - "true": don't know as some relevant execution may refute it
  - "false": error found
- 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:
  - "true": don't know as some relevant execution may refute it
  - "false": error found
- 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

Fairness Constraints in CTL

- For LTL it holds: $TS \models_{\mathit{fair}} \varphi$ if and only if $TS \models (\mathit{fair} \rightarrow \varphi)$
- An analogous approach for CTL is not possible
- Formulas of the form $\forall(\mathit{fair} \rightarrow \varphi)$ and $\exists(\mathit{fair} \wedge \varphi)$ 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., $\Diamond\Box\, \neg b \vee \Box\Diamond\, c$ 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

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

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.:

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

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:

$s \models_{fair} a$ iff $a \in L(s)$

$s \models_{fair} \neg \Phi$ iff $\neg (s \models_{fair} \Phi)$

$s \models_{fair} \Phi \vee \Psi$ iff $(s \models_{fair} \Phi) \vee (s \models_{fair} \Psi)$

$s \models_{fair} \exists \varphi$ iff $\pi \models_{fair} \varphi$ for some fair path π that starts in s

$s \models_{fair} \forall \varphi$ iff $\pi \models_{fair} \varphi$ for all fair paths π that start in s

as before

$\pi \models_{fair} \bigcirc \Phi$ iff $\pi[1] \models_{fair} \Phi$

$\pi \models_{fair} \Phi \, U \, \Psi$ iff $\big(\exists j \geq 0. \, \pi[j] \models_{fair} \Psi$ and $(\forall \, 0 \leq i < j. \, \pi[i] \models_{fair} \Phi)\big)$

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:

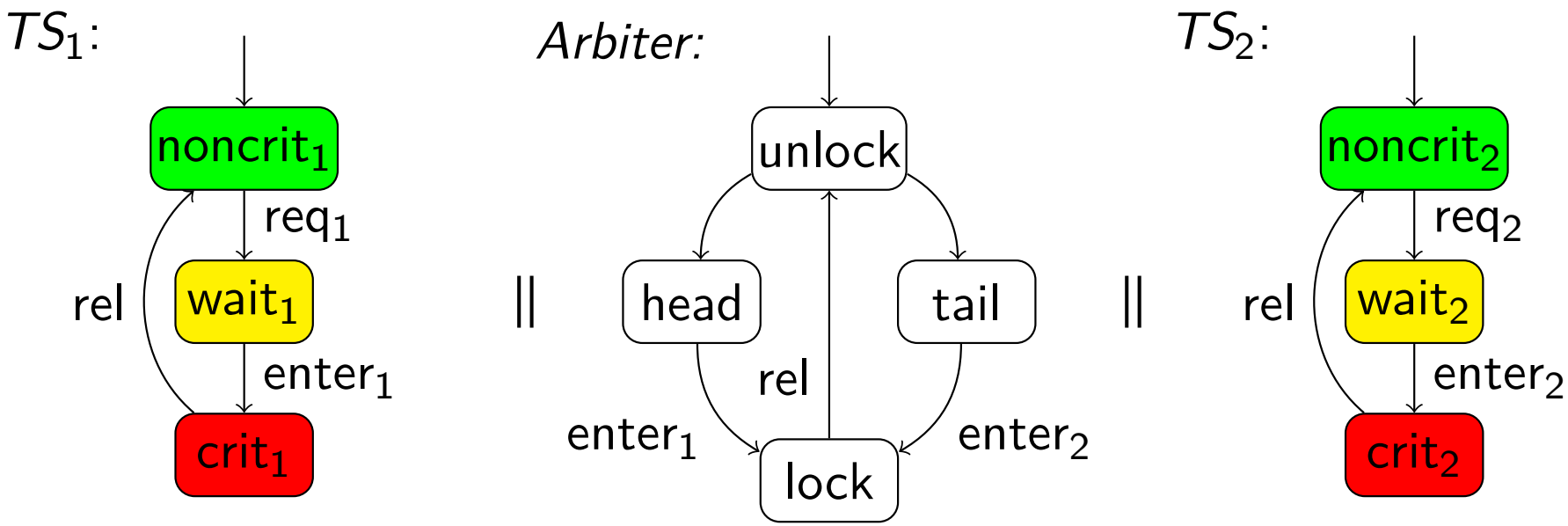
$$Sat_{fair}(\Phi) \;=\; \{\, s \in S \mid s \models_{fair} \Phi \,\}$$

- 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



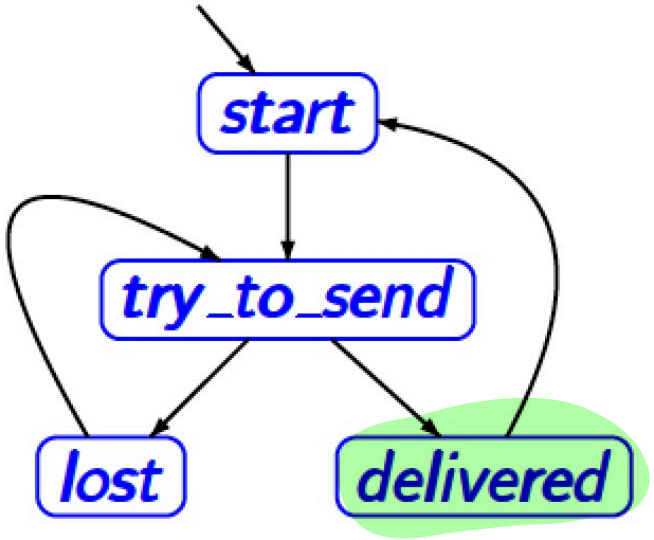
$$TS_1 \parallel Arbiter \parallel TS_2 \not\models (\forall\Box\forall\Diamond\, crit_1) \wedge (\forall\Box\forall\Diamond\, crit_2)$$

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

$\mathcal{T} \not\models \Phi$



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

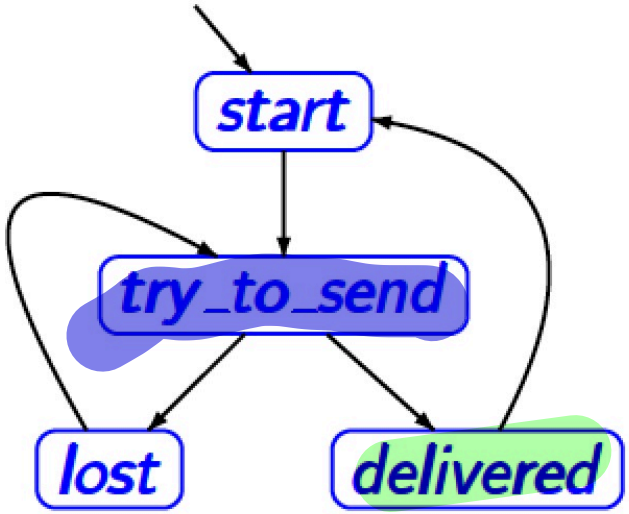
$\mathcal{T} \models_{ufair} \Phi$ ✓

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

$Sat(\exists\bigcirc start) = \{delivered\}$

$ufair \;\widehat{=}\; \Box\Diamond delivered$

Example



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

$\mathcal{T} \models_{ufair} \Phi$ ✓

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

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

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

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

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

Overview

The Fair CTL Model-Checking Problem

Given:

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

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

Our approach: 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 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 $s \in Sat(\Phi_i)$ (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 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 $s \in Sat(\Phi_i)$ (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

Some Useful Results

Recall: $\pi \vDash \mathit{fair}$ iff $\pi[j..] \vDash \mathit{fair}$ for some $j \geq 0$ iff $\pi[j..] \vDash \mathit{fair}$ for all $j \geq 0$

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

1. $s \vDash_{\mathit{fair}} \exists \bigcirc a$ iff $\exists s' \in \mathit{Post}(s)$ with $s' \vDash a$ and $\mathit{FairPaths}_{\mathit{fair}}(s') \neq \varnothing$
2. $s \vDash_{\mathit{fair}} \exists (a \,\mathsf{U}\, a')$ if and only if there exists a finite path fragment

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

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

Proof.

On the black board. □

$s \models_{fair} \exists \bigcirc a$ iff $\exists s' \in Post(s)$ with $s' \models a$ and $FairPaths_{fair}(s') \neq \emptyset$

Proof:

"$\Rightarrow$": Since $s \models_{fair} \exists \bigcirc a$, there is $\pi = s\, s_1\, s_2 \ldots \in FairPaths(s) : \pi[1] = s_1 \models a$

It follows that

- $s_1 \in Post(s)$
- $s_1 \models a$ and
- $\pi[1\ldots] = s_1\, s_2 \ldots$ is also fair, i.e. $FairPaths(s_1) \neq \emptyset$

"$\Leftarrow$": Let $s' \models a$ with $FairPaths(s') \neq \emptyset$ for some $s' \in Post(s)$

It follows that there is $\pi = s\, s'\, s_2\, s_3 \ldots \in Paths(s)$ with

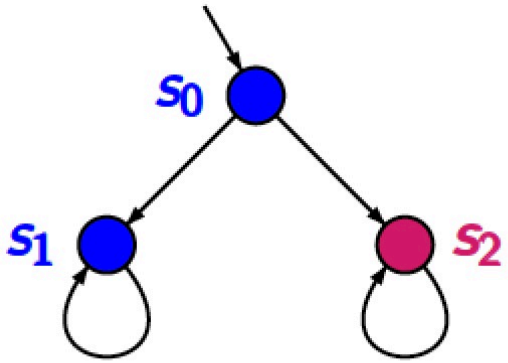
- $\pi \models \bigcirc a$
- $\pi[1\ldots] \models fair \rightsquigarrow \pi \models fair$

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

Fair Path Existence

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

Example



$\bullet = \{a\}$

$\bullet = \varnothing$

$fair = \Box\Diamond a$

$Sat_{fair}(\exists\Box true) = \{s_0, s_2\}$

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:

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

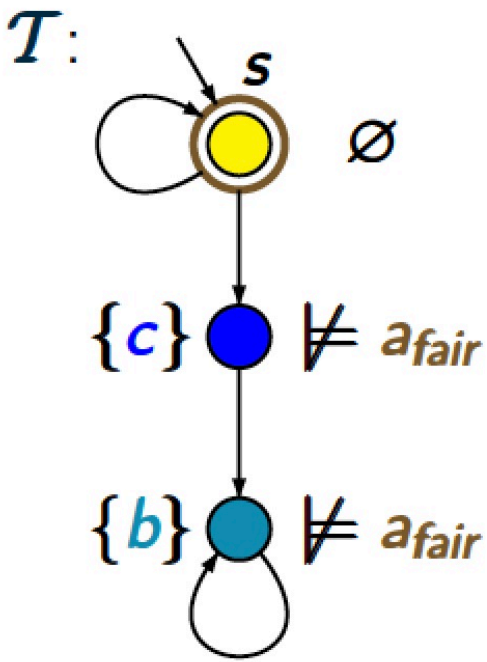
Basic Model-Checking Algorithm for Fair CTL

- 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:

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

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

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

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

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

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

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

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

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

Characterising $Sat_{fair}(\exists\Box a)$ for Strong Fairness (1)

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

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. $s_i \vDash a$, for any $0 \leq i \leq n$, and $s'_j \vDash a$, 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 $0 < i \leq k$

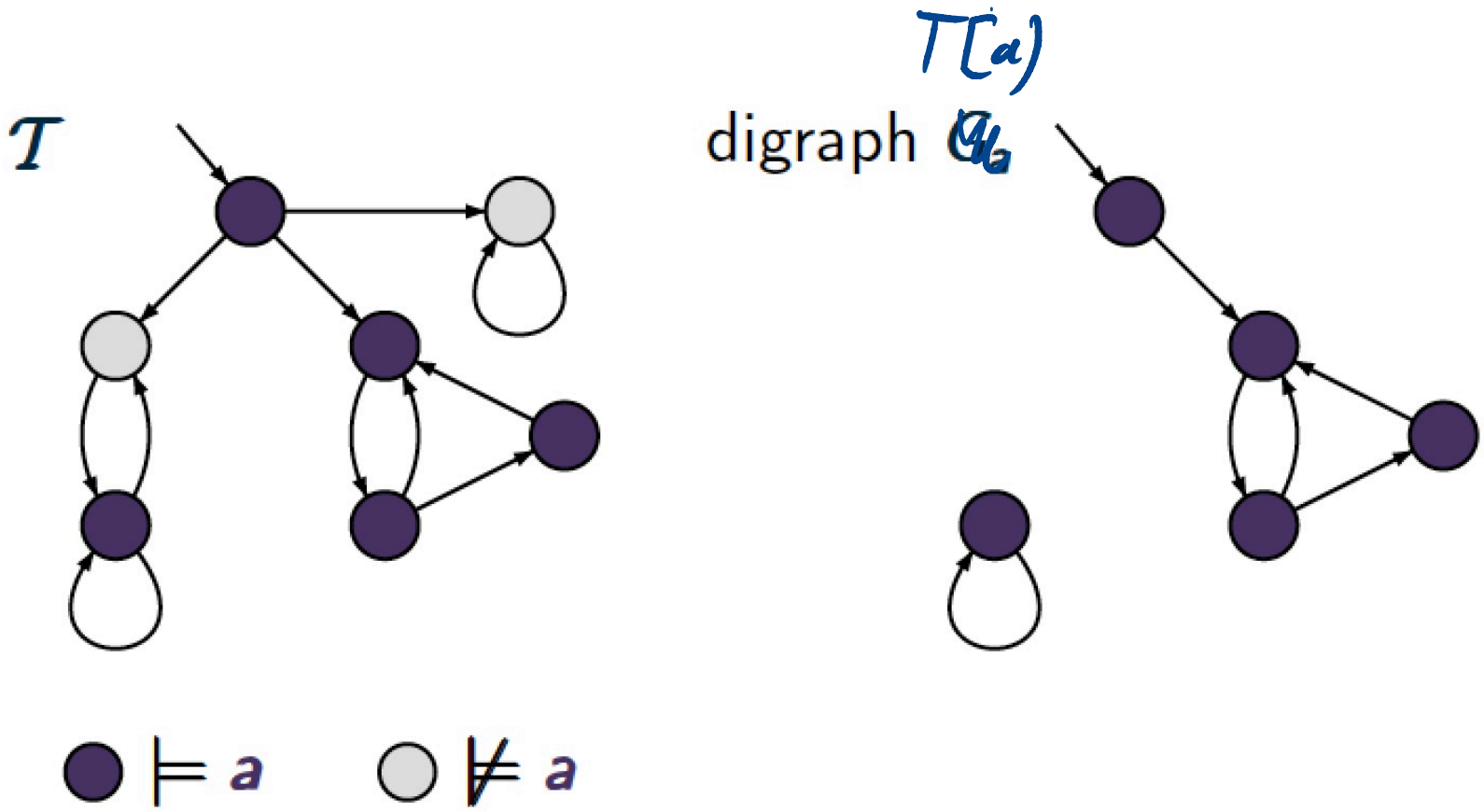
Characterising $Sat_{fair}(\exists\Box a)$ for Strong Fairness (2)

- 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).

Example



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

Characterising $Sat_{fair}(\exists\Box a)$ for Strong Fairness (2)

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

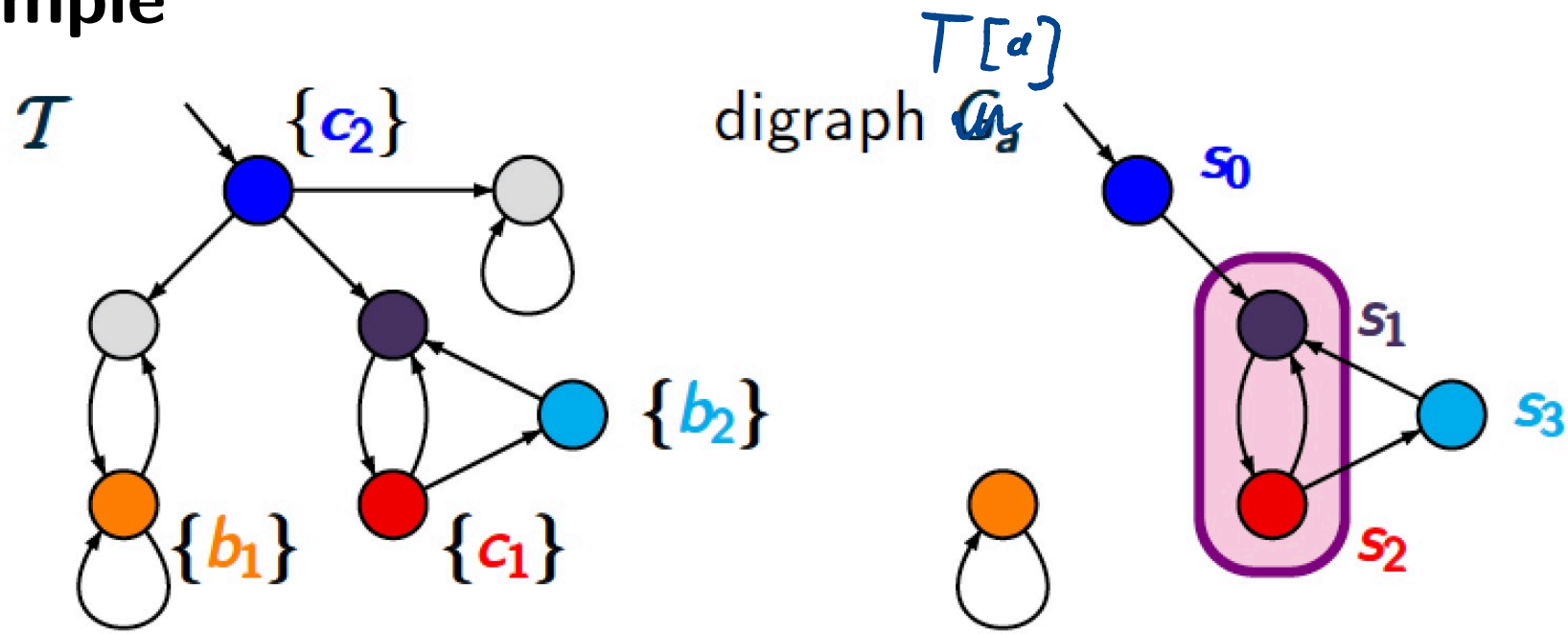
- $s \models_{fair} \exists\Box a$ iff s can reach a strongly connected node-set[2] D in $TS[a]$ with:

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

- $Sat_{fair}(\exists\Box a) = \{ s \in S \mid Reach_{TS[a]}(s) \cap T \neq \varnothing \}$
  - 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



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

$s_0 \models_{fair} \exists\Box a$ 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

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

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

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

$\exists\Box a$ 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:

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

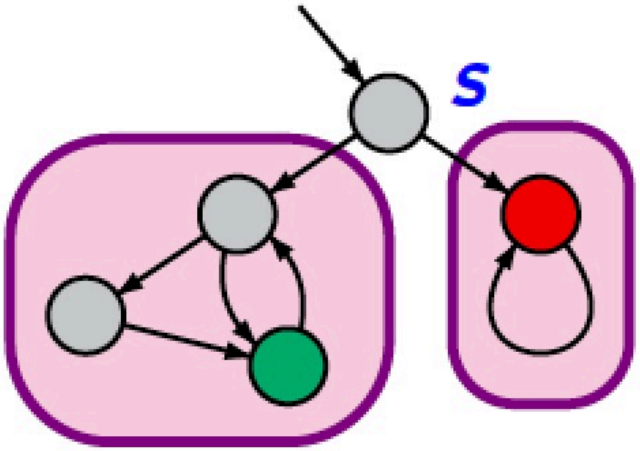
$$\Rightarrow T \text{ can be determined by a depth-first search procedure}$$

We can also see unconditional fairness as a special case of strong fairness:

$$ufair \;\equiv\; \bigwedge_{0<i\leq k} \Box\Diamond \text{ true} \rightarrow \Box\Diamond\, b_i$$

Example

digraph G_a



fairness assumption:

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

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

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

- Assume a single strong fairness constraint $sfair = \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$
- $s \models_{sfair} \exists\Box a$ iff C is a non-trivial SCC in $TS[a]$ reachable from s with:
  - (i) $C \cap Sat(b_1) \neq \emptyset$, or
  - (ii) $D \cap Sat(a_1) = \emptyset$, for some strongly connected node-set D in C

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

- Assume a single strong fairness constraint $sfair = \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$
- $s \models_{sfair} \exists\Box a$ iff C is a non-trivial SCC in $TS[a]$ reachable from s with:

  (i) $C \cap Sat(b_1) \neq \emptyset$, or
  (ii) $D \cap Sat(a_1) = \emptyset$, for some strongly connected node-set D in C

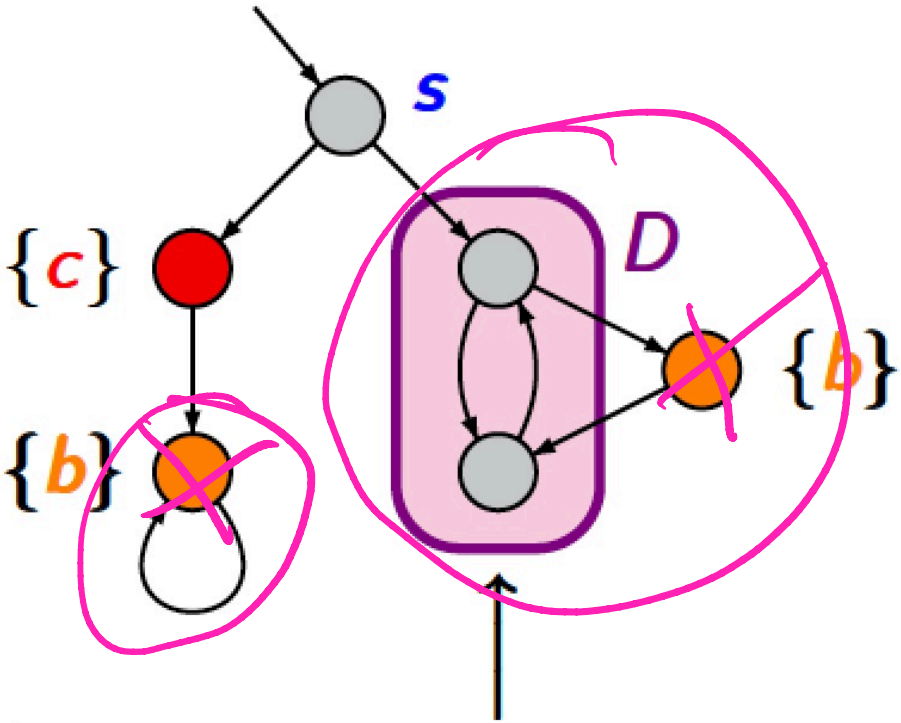
- D is a non-trivial SCC in the graph that is obtained from $C[\neg a_1]$
- For T the union of non-trivial SCCs in satisfying (i) and (ii):

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

Example: One Strong Fairness Constraint

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

digraph G_a



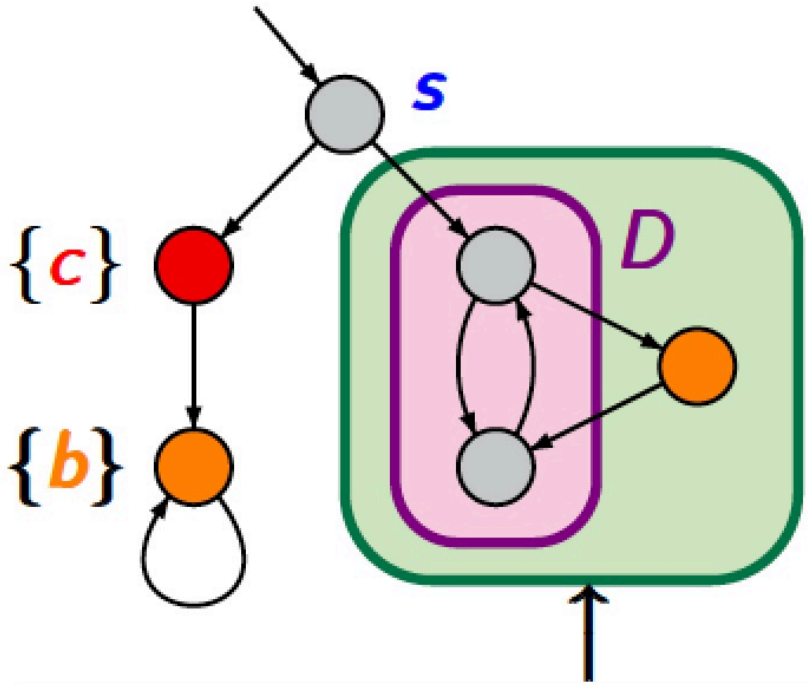
$\leftarrow$ $s \models_{fair} \exists\Box a$

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 $G_a|_C \setminus Sat(b)$

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

- Assume a single strong fairness constraint $sfair = \Box\Diamond a_1 \rightarrow \Box\Diamond b_1$
- $s \models_{sfair} \exists\Box a$ iff C is a non-trivial SCC in $TS[a]$ reachable from s with:

  (i) $C \cap Sat(b_1) \neq \emptyset$, or
  (ii) $D \cap Sat(a_1) = \emptyset$, for some strongly connected node-set 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 (i) and (ii):

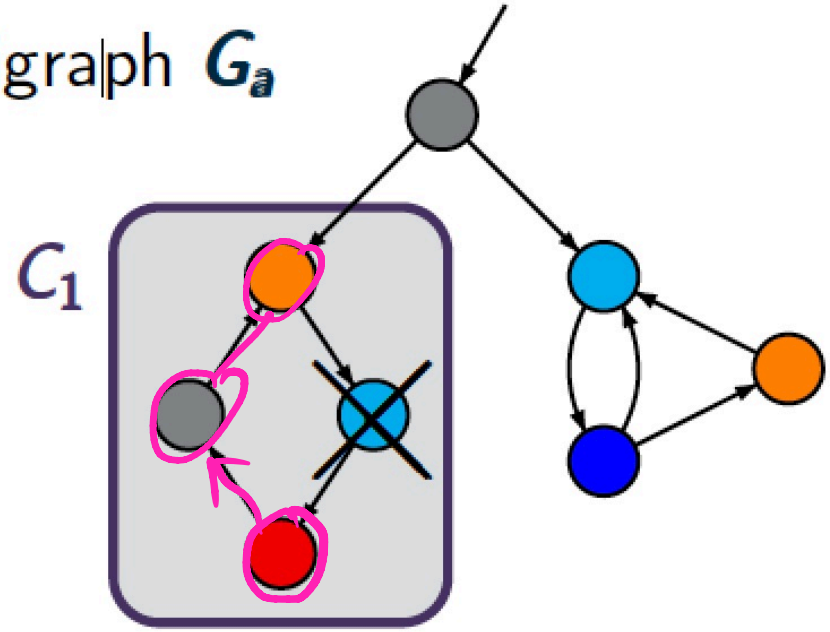
$$s \models_{sfair} \exists\Box a \quad \text{if and only if} \quad Reach_{TS[a]}(s) \cap T \neq \emptyset$$

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

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



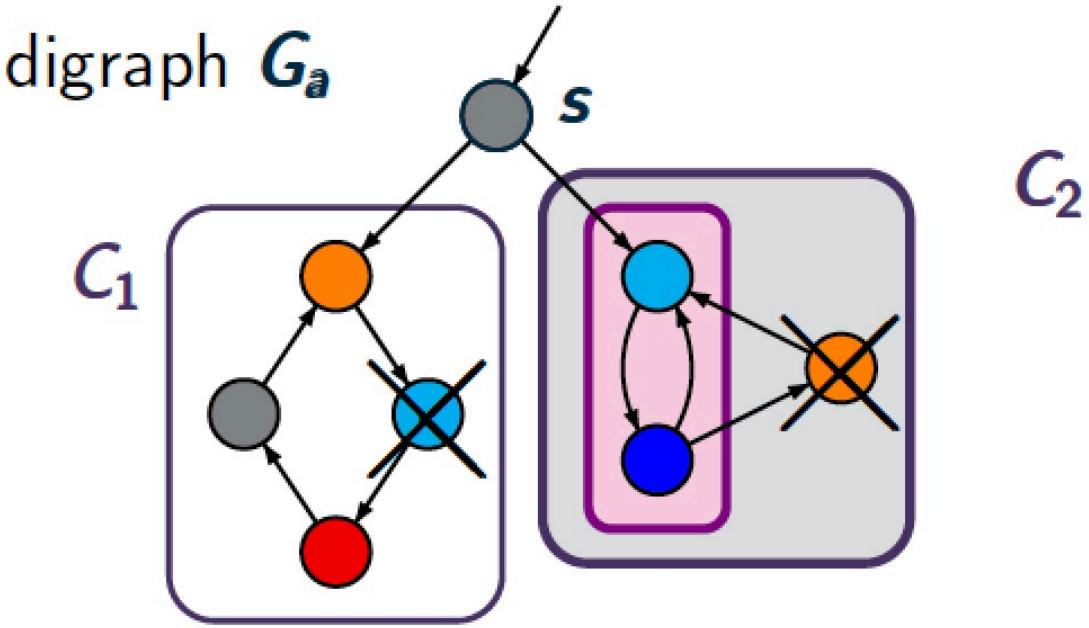
first SCC: $C_1 \cap Sat(c_2) = \varnothing$

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

$\rightsquigarrow$ there is no cycle

Example: Two Strong Fairness Constraints

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



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

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

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

Algorithm

compute the SCCs of the digraph G_a;

$T := \emptyset$;

FOR ALL nontrivial SCCs C of G_a DO

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

OD

$Sat_{fair}(\exists \Box a) := \{s \in S : Reach_{G_a}(s) \cap T \neq \emptyset\}$

backward search from T

CheckFair is a recursive procedure over the k strong fairness constraints

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

CheckFair Algorithm (for completeness)

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

IF $\forall i \in \{1, \ldots, k\}.\ C \cap \mathit{Sat}(c_i) \neq \emptyset$ THEN return "true" FI

choose $j \in \{1, \ldots, k\}$ with $C \cap \mathit{Sat}(c_j) = \emptyset$;

remove all states in $\mathit{Sat}(b_j)$;

IF the resulting graph G is acyclic THEN return "false" FI

FOR ALL nontrivial SCCs D of G DO

  IF $\mathit{CheckFair}(D, k-1, \bigwedge_{i \neq j} (\Box\Diamond b_i \rightarrow \Box\Diamond c_i))$

  THEN return "true"

OD

return "false"

time complexity:

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

$\exists\Box a$ under Weak Fairness

Weak fairness constraints are treated analogously:

- Let $wfair = \bigwedge_{0<i\leq k} (\Diamond\Box\, a_i \rightarrow \Box\Diamond\, b_i)$
- $s \models_{wfair} \exists\Box a$ iff s can reach a strongly connected node-set D in $TS[a]$ with:

$$\underline{D \not\subseteq Sat(a_i)} \quad \text{or} \quad D \cap Sat(b_i) \neq \varnothing \quad \text{for all} \quad 0 < i \leq k$$

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

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 $\exists\Box a$
- Complexity of fair CTL model checking is $O(|TS|\cdot|\Phi|\cdot|fair|)$

Next Lecture

Monday June 20, 10:30