Formal Methods in Software Development

Ivano Salvo
and Igor Melatti

Computer Science Department

SAPIENZA
UNIVERSITÀ DI ROMA

Lesson **6**, October 29th, 2019

Lesson 6a:

The Fairness problem

Fairness

- System models often **abstracts from details** such as, for example, **scheduler policies**.
- Interleaving semantics does not rule out unrealistic behaviour, for example, those in which **some processes do not make any progress.**

Of course, one can embody a fair process scheduling in the model, as in the case of Peterson mutual exclusion algorithm.

Alternatively, one can **assume some fairness properties**, and perform model checking under such assumptions.

Example: (un)fair Schedulers

Mutual Exclusion with shared variables: The starvation freedom property:

> *"Once access is requested, a process does not have to wait infinitely long before acquiring access to its critical section"*

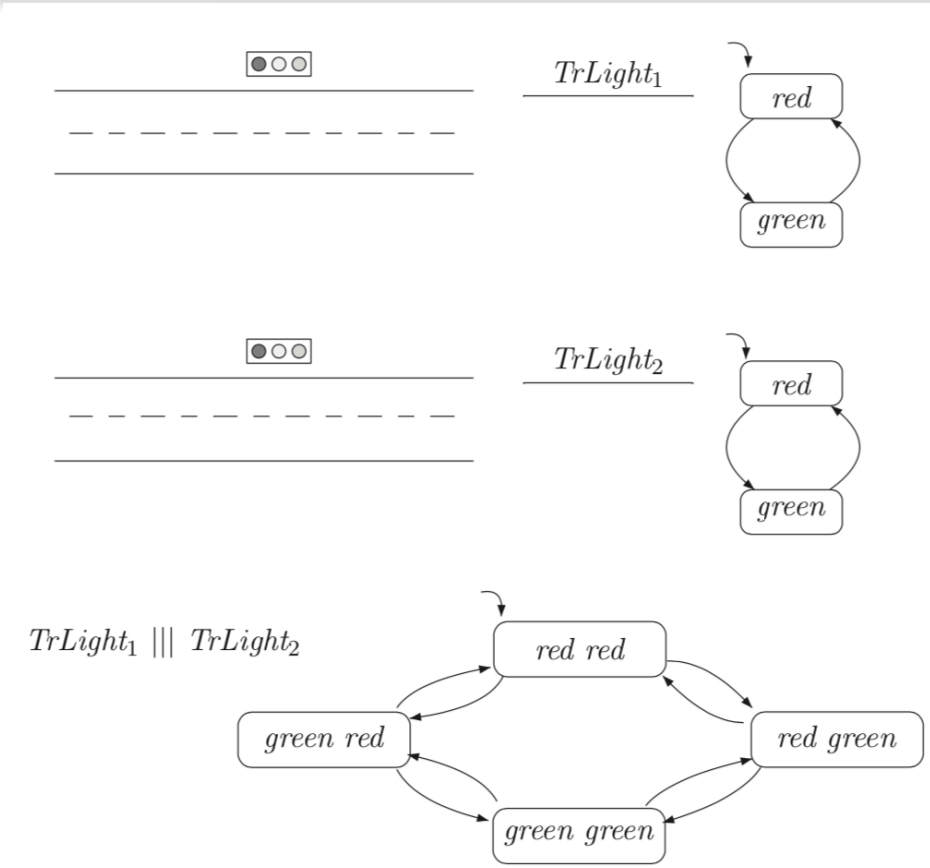
is violated, just because, **abstracting from the scheduling policy**, a legal execution of the system assignes the critical resource always to the same process.

The property:

"Each of the processes is infinitely often in its critical section"

is violated also by the Peterson protocol, as it **does not exclude** that **a process would never** (or finitely often**) request** to enter its critical section.

Example: Interleaving Semantics



Two independent traffic lights (lesson **1**): Interleaving semantics allows infinite executions in which **only the first traffic light commute**: {red_1, red_2}{green1, red_2} {red_1, red_2}{green1, red_2} {red_1, red_2}{green1, red_2}…

Fairness notions

Unconditional Fairness: "every process gets its turn infinitely often" (without conditions, aka **impartiality**)

Strong Fairness: "every process that is **enabled infinitely often** gets its turn infinitely often" (aka **compassion**)

Weak Fairness: every process that is **continously enabled from a certain point on** gets its turn infinitely often" (aka **justice**)

Many other fairness notions have been introduced in literature and there is no clear consensus about which notion should be used in some scenario.

Fairness def. (action based)

Definition 3.43. Unconditional, Strong, and Weak Fairness

For transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states, $A \subseteq Act$, and infinite execution fragment $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ of TS:

LTL (state based) equivalent formulas

1. ρ is *unconditionally A-fair* whenever $\overset{\infty}{\exists}\, j.\ \alpha_j \in A$.

   G F φ

2. ρ is *strongly A-fair* whenever

$$\left(\overset{\infty}{\exists}\, j.\ Act(s_j) \cap A \neq \varnothing\right) \implies \left(\overset{\infty}{\exists}\, j.\ \alpha_j \in A\right).$$

   G F $\varphi \rightarrow$ G F ψ

3. ρ is *weakly A-fair* whenever

$$\left(\overset{\infty}{\forall}\, j.\ Act(s_j) \cap A \neq \varnothing\right) \implies \left(\overset{\infty}{\exists}\, j.\ \alpha_j \in A\right).$$

   F G $\varphi \rightarrow$ G F ψ

■

Here, $\overset{\infty}{\exists}\, j$ stands for "there are infinitely many j" and $\overset{\infty}{\forall}\, j$ for "for nearly all j" in the sense of "for all, except for finitely many j". The variable j, of course, ranges over the natural numbers.

Ex: Shared variables program

$$\begin{aligned} \textbf{proc}\ \mathsf{Inc} &= \textbf{while}\ \langle x \geqslant 0\ \textbf{do}\ x := x + 1 \rangle\ \textbf{od} \\ \textbf{proc}\ \mathsf{Reset} &= x := -1 \end{aligned}$$

This process terminates only if **unconditional fairness** is assumed…If process Inc or process Reset can execute infinitely often, the concurrent program does not terminate.

[Brackets ⟨…⟩ means "atomic actions"]

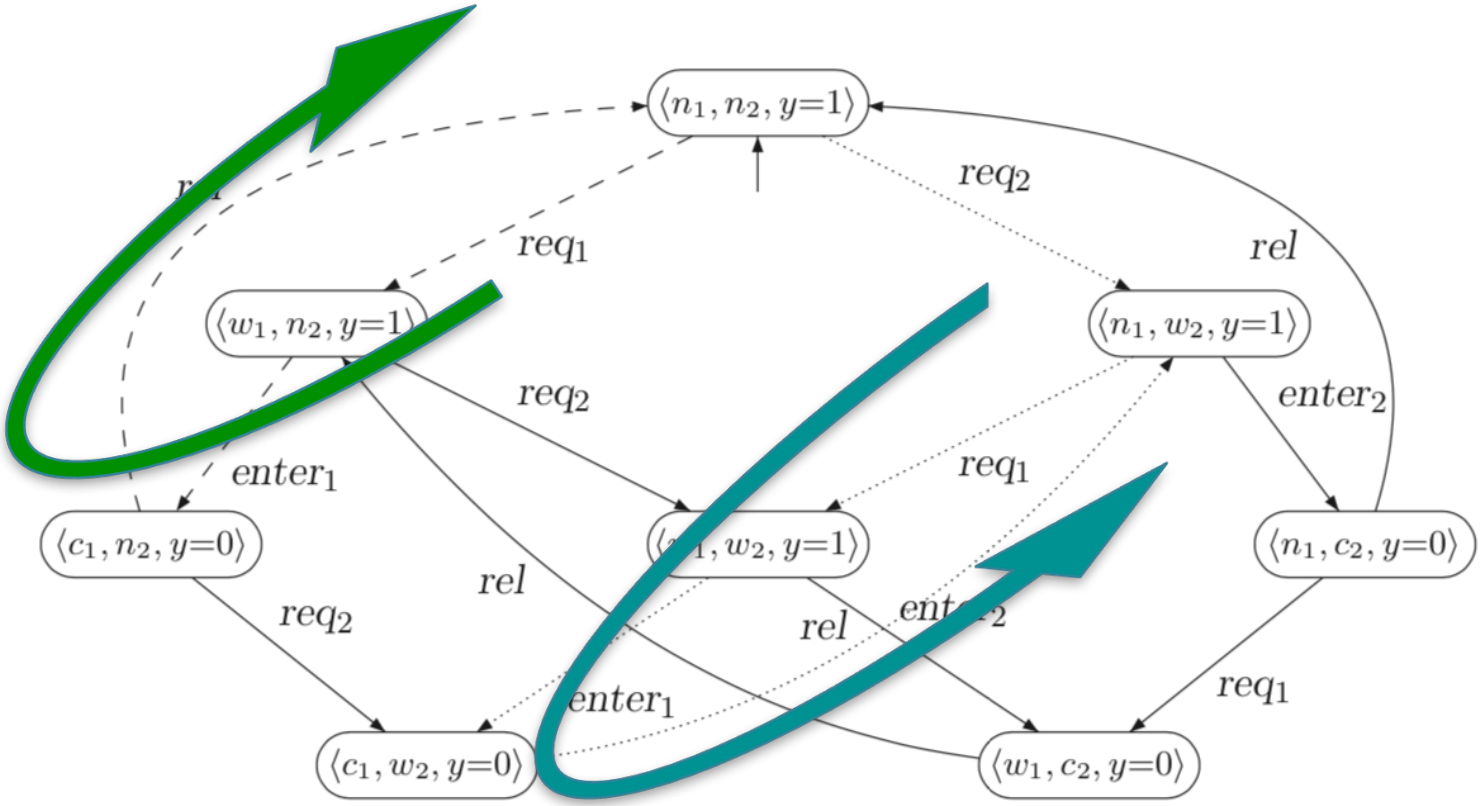
Which notion of fairness we should use? No answer!

Keep in mind: if the fairness constraints are **too strong, relevant computation can be ruled out**. By contrast, if the fairness constraints are **too weak**, we refute a property because we consider **unrealistic behaviour** of a system.

Uncond. Fairnes A ⟹ Strong Fairness A ⟹ Weak Fairness A

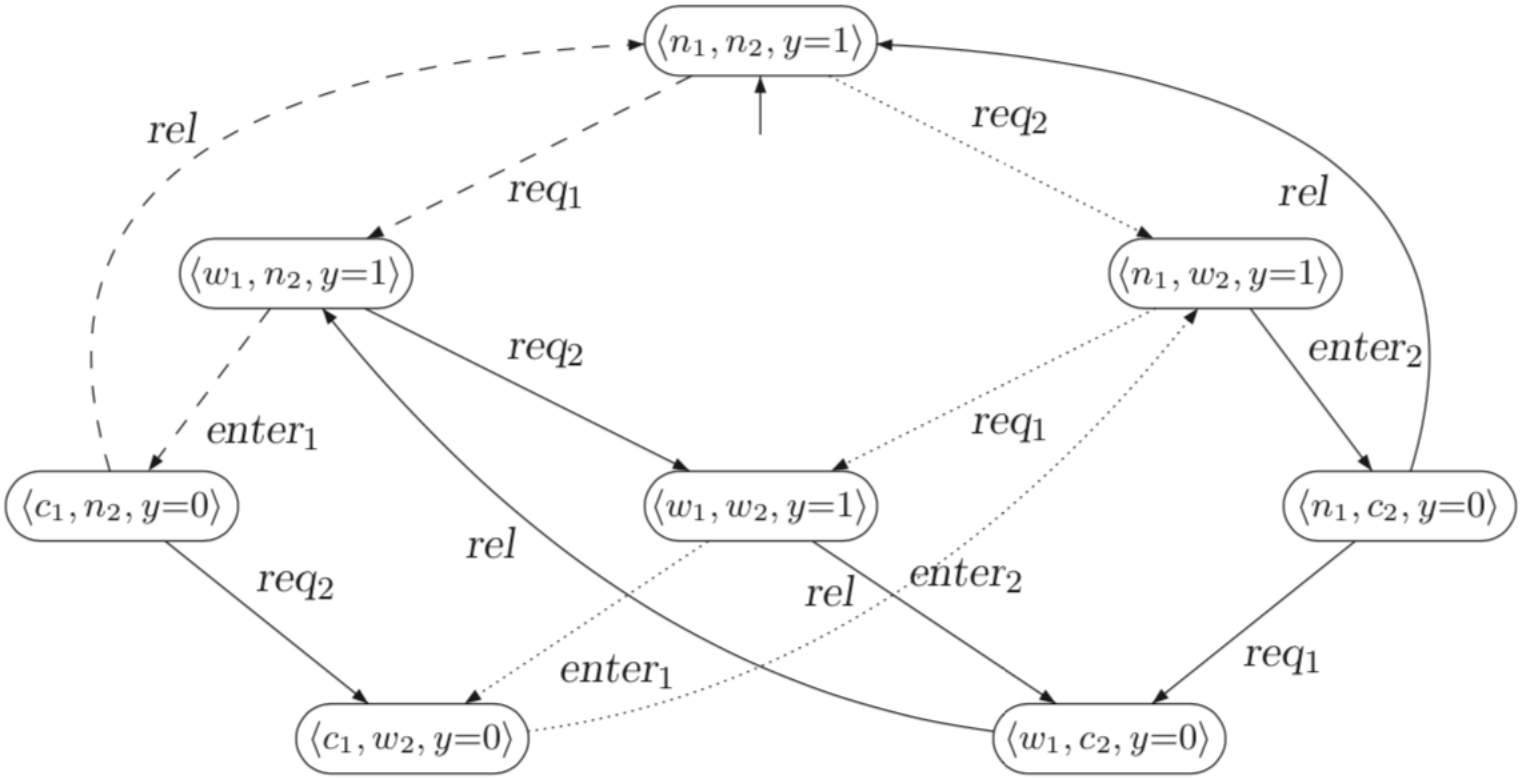
Mutual Exclusion Reloaded

The dashed execution fragment **is strongly fair** premises are vacously true), but **not unconditionally fair** for {enter$_2$}. The dotted **is weakly fair, but not strongly fair.** Process 1 requests access infinitely often, but not continously.



Example: How to model Fairness

The strong fairness assumption {{**enter$_1$, enter$_2$**}} ensure only that one of the two process enter its critical section infinitely often. Probably, {{**enter$_1$**}, {**enter$_2$**}} is the intended solution!



Fairness: Linear Time Properties

Definition: Let $P \subseteq (2^{AP})^{\omega}$ be an LT property over AP and let $\mathcal{F}$ be a fairness assumption over A. A transition system $\mathcal{M}$ **fairly satisfies *P***, notation $\mathcal{M} \models_{\mathcal{F}} P$, if and only if $\mathsf{fairTraces}(\mathcal{M}) \subseteq P$.

If all executions of $\mathcal{M}$ satisfies $\mathcal{F}$, then $\mathcal{M} \models_{\mathcal{F}} P$ iff $\mathcal{M} \models P$. More in general, we have that $\mathcal{M} \models P \Longrightarrow \mathcal{M} \models_{\mathcal{F}} P$.

As said before, we also have:

$$\mathcal{M} \models_{\text{weak } \mathcal{F}} P \Longrightarrow \mathcal{M} \models_{\text{strong } \mathcal{F}} P \Longrightarrow \mathcal{M} \models_{\text{uncond } \mathcal{F}} P$$

Example: Independent Traffic Lights. The fair assumption:

$\{\{\text{switchToGreen}_1, \text{switchToRed}_1\}, \{\text{switchToGreen}_2, \text{switchToRed}_2\}\}$

rules out unrealistic behaviour, no matter if this is interpreted as strong, weak or unconditional fairness constraint.

$\text{TrLight}_1 \,|||\, \text{TrLight}_2 \models_{\mathcal{F}} \mathbf{F}\,\mathbf{G}$ green ≡"each traffic light is green infin. often"

Example: Mutual Exclusion

Let us consider again the semaphore based mutual exclusion protocol. Let us define the following fairness constraints:

$\mathcal{F}_{weak} = \{\{req_1\},\{req_2\}\}$ $\mathcal{F}_{strong} = \{\{enter_1\},\{enter_2\}\}$ $\mathcal{F}_{uncond} = \varnothing$

The strong fairness assumption $\mathcal{F}_{strong}$ does not forbid a process to never release its critical section.

The weak fairness assumption $\mathcal{F}_{weak}$ implies that a process requires to enter critical section infinitely often (and hence it has to leave infinitely often its critical section).

Also Peterson's protocol ensure that process will enter its critical section if it requires it infinitely often. But it does not ensure that processes leave their critical section. To ensure this, we should impose the weak fairness assumption $\mathcal{F}_{weak} = \{\{req_1\},\{req_2\}\}$.

Weak or Strong Fairness

Rule of Thumb:

Strong fairness is appropriate to obtain an adequate resolution of **contentions between processes**

Weak fairness suffices for sets of actions that represent the concurrent execution of independent actions (**interleaving**)

Example: Let us assume we have n processes represented by transition systems $\mathcal{M}_i = (S_i, A_i, \rightarrow_i, AP_i, L_i)$ and consider the parallel composition $\mathcal{M}_i = \mathcal{M}_1 \,|||\, \mathcal{M}_2 \,|||\, \ldots \,|||\, \mathcal{M}_n$. Let us suppose that each pair of processes synchronize on a set of actions $H_{i,j}$.

Example: Fair Synchronization

The **strong fairness assumption** $\{A_1, A_2, \ldots, A_n\}$ means that each process progress infinitely often (provided that infinitely often a process has an enabled action to execute). This assumption is satisfied, however, only with internal actions and no sync!

$\{\{\alpha\} \mid \alpha \in H_{i,j}\ 0 < i < j \leq n\}$ forces every sync action to be performed infinitely often.

$\{H_{i,j} \mid 0 < i < j \leq n\}$ forces every pair of processes to synchronize infinitely often, maybe on the same action.

$\{\bigcup_{0<i<j\leq n} H_{i,j}\}$ just requires that there are infinite synchronization actions, regardless of which are processes involved.

For **internal actions**, the **weak fairness assumption**: $\{A_1 \backslash H_1, \ldots, A_n \backslash H_n\}$, where $H_i = \bigcup_{i \neq j} H_j$, is appropriate, since internal actions are continously ready to be executed.

Lesson 6b:

Fairness in LTL Model Checking

Fairness is expressible in LTL

The three notions of fairness we have considered can be expressed by LTL formulas of the shape:

Unconditional fairness: $\mathbf{G}\,\mathbf{F}\,\varphi$

Strong fairness: $\mathbf{G}\,\mathbf{F}\,\varphi \rightarrow \mathbf{G}\,\mathbf{F}\,\psi$

Weak fairness: $\mathbf{F}\,\mathbf{G}\,\varphi \rightarrow \mathbf{G}\,\mathbf{F}\,\psi$

The only problem is that LTL formulas are built on atomic propositions that label states: therefore φ and ψ depend on the state labeling and they single out set of states of a transition system $\mathcal{M}$, i.e. $\{ s \mid \mathcal{M}, s \vDash \varphi \}$.

However, this is not a limitation…

Mutual Exclusion

The strong (**action based**) fairness assumption $\mathcal{F}_{strong}$= {{enter$_1$},{enter$_2$}} can be represented by the (**state based**) LTL formula (observe that enter$_1$ can be executed only if P_1 is in the state wait$_1$ and P_2 is not in its critical section):

$$\text{sfair}_1 = \mathbf{G}\,\mathbf{F}\,(\text{wait}_1 \wedge \neg\text{crit}_2) \rightarrow \mathbf{G}\,\mathbf{F}\,\text{crit}_1$$

The assumption sfair$_2$ can be defined analogously.

$\mathcal{F}_{strong}$ does not forbid a process to never leave its critical section. The (**action based**) weak assumption = {{req$_1$}, {req$_2$}} can be encoded as the (**state based**) LTL formula (observe that the action req$_1$ is executable only if P_1 is in the state noncrit$_1$)

$$\text{wfair}_1 = \mathbf{F}\,\mathbf{G}\,\text{noncrit}_1 \rightarrow \mathbf{G}\,\mathbf{F}\,\text{wait}_1$$

The assumption wfair$_2$ can be defined analogously.

Action vs State based Fairness 1

Kripke structures have no action labels. One can always keep **information into states**.

Let $\mathcal{M}$ be a transition system $(S, A, \rightarrow, I, AP, L)$. We can define the system $\mathcal{M}'=(S', A', \rightarrow', I', AP', L')$ where:

- $A' = A \cup \{ \text{begin} \}$,
- $I' = I \times \{ \text{begin} \}$,
- $S' = I' \cup (S \times A)$,
- If $s_0 \rightarrow_\alpha s$ $(s_0 \in I)$, then $(s_0, \text{begin}) \rightarrow'_\alpha (s, \alpha)$.
  If $s \rightarrow_\alpha s'$ then $(s, \beta) \rightarrow'_\alpha (s', \alpha)$
- $AP' = AP \cup \{\text{enabled}(\alpha), \text{taken}(\alpha) \mid \alpha \in A\}$
- $L'(s, \alpha) = L(s) \cup \{\text{taken}(\alpha)\} \cup \{\text{enabled}(\beta) \mid \beta \in \text{Act}(s)\}$ and $L'(s_0, \text{begin}) = L(s_0) \cup \{\text{enabled}(\beta) \mid \beta \in \text{Act}(s_0)\}$

Action vs State based Fairness 2

Theorem. traces($\mathcal{M}$) = traces($\mathcal{M}'$). Moreover, strong fairness for a set of actions $F \subseteq A$ can be described by the LTL formula:

$strongFair_F \equiv \mathbf{G}\ \mathbf{F}\ \mathsf{enabled}(F) \rightarrow \mathsf{taken}(F)$

[similar for weak fairness and unconditional fairness]

Theorem. Let $\mathcal{M}$ be a transition system without terminal states and let φ be a LTL formula and let $\mathcal{F}$ be a fairness assumption that can be modeled by a LTL formula ψ. Then:

$$\mathcal{M} \models_{\mathcal{F}} \varphi \text{ if and only if } \mathcal{M} \models \psi \rightarrow \varphi$$

Be careful about complexity

Best LTL model checking algorithms ***are exponential on the size of the formula*** φ to be verified.

If fairness constraints are modeled by complex LTL formula ψ, the **computational cost** to solve the model checking problem

$$\mathcal{M} \models \psi \rightarrow \varphi$$

could be huge!

Lesson 6c:

Fairness in CTL & CTL Model Checking with fairness constraints

Strong Fairness in CTL

We will consider **strong fairness constraints** of the form:

$$sfair = \bigwedge_{1 \le i \le k} \mathbf{G}\,\mathbf{F}\,\varphi_i \rightarrow \mathbf{G}\,\mathbf{F}\,\psi_i$$

Where φ_i and ψ_i are CTL formulas (without fairness). Observe that being CTL formulas, φ_i and ψ_i identify a set of states of a Kripke structure $\mathcal{M}$: $\text{Sat}(\varphi_i) = \{ s \mid \mathcal{M}, s \models \varphi_i \}$.

On the other hand, given a path $\pi = s_0 s_1 s_2 \ldots$, we have:

$$\pi \models_{\text{LTL}} \mathbf{G}\,\mathbf{F}\,\varphi_i \rightarrow \mathbf{G}\,\mathbf{F}\,\psi_i$$

if for all $1 \le i \le k$, there exists j such that $s_j \models_{\text{CTL}} \varphi_i$ for **finitely many** indices j, or $s_j \models_{\text{CTL}} \psi_i$ for **infinitely many** indices (remember that $a \rightarrow b \equiv \neg a \vee b$).

A path π is fair $\mathcal{M}$, if $\pi \models_{\text{LTL}} \mathbf{G}\,\mathbf{F}\,\varphi_i \rightarrow \mathbf{G}\,\mathbf{F}\,\psi_i$. We denote with:

- fairPaths(s) the set of fair paths starting in a state s,
- fairPaths($\mathcal{M}$) the set of fair paths starting in an initial state s_0 of $\mathcal{M}$.

Fairness is not expressible in CTL

Formulas of the form **G F** $\varphi \rightarrow$ **G F** ψ are not in CTL, because:

1. The formula **G F** φ has two consecutive temporal operators
2. The boolean connective $\rightarrow$ is applied to two path formulas

In CTL we **must change the semantics** of **E** and **A** stipulating that they quantify over **fair paths**.

We define a new $\models_F$ semantic satisfaction judgement:

1.	$M, s \models_F p$	$\Leftrightarrow$	there exists a fair path starting from s and $p \in L(s)$.
5.	$M, s \models_F \mathbf{E}(g_1)$	$\Leftrightarrow$	there exists a fair path π starting from s such that $\pi \models_F g_1$.
6.	$M, s \models_F \mathbf{A}(g_1)$	$\Leftrightarrow$	for all fair paths π starting from s, $\pi \models_F g_1$.

Observe that 1. influence indirectly also the semantics of temporal operators **X** or **U**!!!

CTL model checking with fairness

Theorem. The CTL model checking problem **with fairness** can be reduced to:

1. The CTL model checking problem **without fairness,** and
2. The problem of computing $\mathrm{Sat}_{\mathbf{fair}}(\mathbf{E}\,\mathbf{G}\,a)$ for some $a \in AP$.

Proof: This approach is quite straightforward for the propositional logic fragment, for example $\mathrm{Sat}_{\mathrm{fair}}(a)$ if $a \in L(s)$ and there exists a fair path starting in s, that is $\mathcal{M}, s \models_{\mathrm{fair}} \mathbf{E}\,\mathbf{G}\,\mathrm{true}$.

Similarly, of course, for $\mathcal{M}, s \models_{\mathrm{fair}} \mathbf{E}\,\mathbf{X}\,f$: there must be a fair path starting in s such that $s_1 \models f$ and $\mathcal{M}, s_1 \models_{\mathrm{fair}} \mathbf{E}\,\mathbf{G}\,\mathrm{true}$.

As for $\mathcal{M}, s \models_{\mathrm{fair}} \mathbf{E}\,[f_1\,\mathbf{U}\,f_2]$ there must be a fair path starting in s such that there exist $\mathcal{M}, s_n \models_{\mathrm{fair}} f_2$ and $\mathcal{M}, s_i \models_{\mathrm{fair}} f_1$ $\mathcal{M}$, for all $1 \leq i \leq n$ and $s_n \models_{\mathrm{fair}} \mathbf{E}\,\mathbf{G}\,\mathrm{true}$ (**observe that only the infinite suffix is relevant for fairness**).

Obviously, in the iterative CTL algorithm, $\mathcal{M}, s \models_{\mathrm{fair}} \mathbf{E}\,\mathbf{G}\,f$ is applied when f has been processed, and hence the problem is to check $\mathcal{M}, s \models \mathbf{E}\,\mathbf{G}\,a_f$ with a_f atomic proposition. □

Summing up…

Let a_{fair} be a fresh atomic proposition such that:

$a_{\text{fair}} \in L(s)$ if and only if $s \in \text{Sat}_{\mathbf{fair}}(\mathbf{E}\,\mathbf{G}\,\text{true}) \equiv \mathcal{M}, s \models_{\text{fair}} \mathbf{E}\,\mathbf{G}\,\text{true}$

Then:

$$\text{Sat}_{\mathbf{fair}}(\mathbf{E}\,\mathbf{X}\,a) \equiv \text{Sat}(\mathbf{E}\,\mathbf{X}\,a \wedge a_{\text{fair}})$$

$$\text{Sat}_{\mathbf{fair}}(\mathbf{E}\,[a\,\mathbf{U}\,a']) \equiv \text{Sat}(\mathbf{E}\,[a\,\mathbf{U}\,a' \wedge a_{\text{fair}}])$$

And those on the right-hand side are pure CTL formulas that can be computed by the usual CTL algorithm (see lesson **3**).

Therefore, we are left with the problem of computing

$$\text{Sat}_{\mathbf{fair}}(\mathbf{E}\,\mathbf{G}\,a)$$

That, in particular, can be used to compute $a_{\text{fair}} \in L(s) \equiv s \in \text{Sat}_{\mathbf{fair}}(\mathbf{E}\,\mathbf{G}\,\text{true})$.

Checking $\mathcal{M}, s \models_{\text{fair}} EG\ a$ (1)

Lemma. Let sfair $= \bigwedge_{1 \le i \le k} \mathbf{G}\,\mathbf{F}\,a_i \rightarrow \mathbf{G}\,\mathbf{F}\,b_i$ be a fair constraint. Then $\mathcal{M}, s \models_{\text{sfair}} \mathbf{EG}\ a$ if and only if there exists a finite path $s_0 s_1 \ldots s_n$ and a cycle $s'_0 s'_1 \ldots s'_r$ such that:

i. $s = s_0$ and $s'_0 = s'_r$

ii. $s_i \models a$ for all $0 \le i \le n$ and $s'_j \models a$ for all $0 \le j \le r$

iii. For all $0 \le i \le k$ $\text{Sat}(a_i) \cap \{s'_0, s'_1, \ldots, s'_r\} = \varnothing$ or $\text{Sat}(b_i) \cap \{s_0, s_1, \ldots, s_n\} \neq \varnothing$

Proof (**if**): Clearly $s_0 s_1 \ldots s_n (s'_0 s'_1 \ldots s'_r)^\omega$ is a fair path according to sfair satisfying **EG** *a*.

(**Only if**) $\mathcal{M}, s \models_{\text{sfair}} \mathbf{EG}\ a$ implies that there exists an infinite fair path $\pi = s_0 s_1 s_2 \ldots$ such that $\pi \models_{\text{sfair}} \mathbf{G}\ a$ and $\pi \models$ sfair. Two cases:

1. $\pi \models \mathbf{G}\,\mathbf{F}\,a_i$. This implies exists $s' \models b_i$ visited infinitely often in π. Let n and r be the first and second occurrence of s'. Clearly $\{s_0, s_1, \ldots, s_n\}$ and $\{s_n, s_{n+1}, \ldots, s_r\}$ satisfies *iii*.

2. $\pi \not\models \mathbf{G}\,\mathbf{F}\,a_i$. Then there exists m such that $s_m, s_{m+1}, \ldots \notin \text{Sat}(b_i)$. There are finitely many states, there is a cycle $s_n, s_{n+1}, \ldots, s_r$ $(n > m)$ such that $\text{Sat}(a_i) \cap \{s_n, s_{n+1}, \ldots, s_r\} = \varnothing$. □

Checking $\mathcal{M}, s \models_{\text{fair}} EG\ a$ (2)

The previous Lemma can be used as follows. Consider the graph G_a whose nodes $V_a = \{s \mid \mathcal{M}, s \models a\}$ and edges $E_a = \{(s, s') \in R \mid s \in V_a, s' \in V_a\}$.

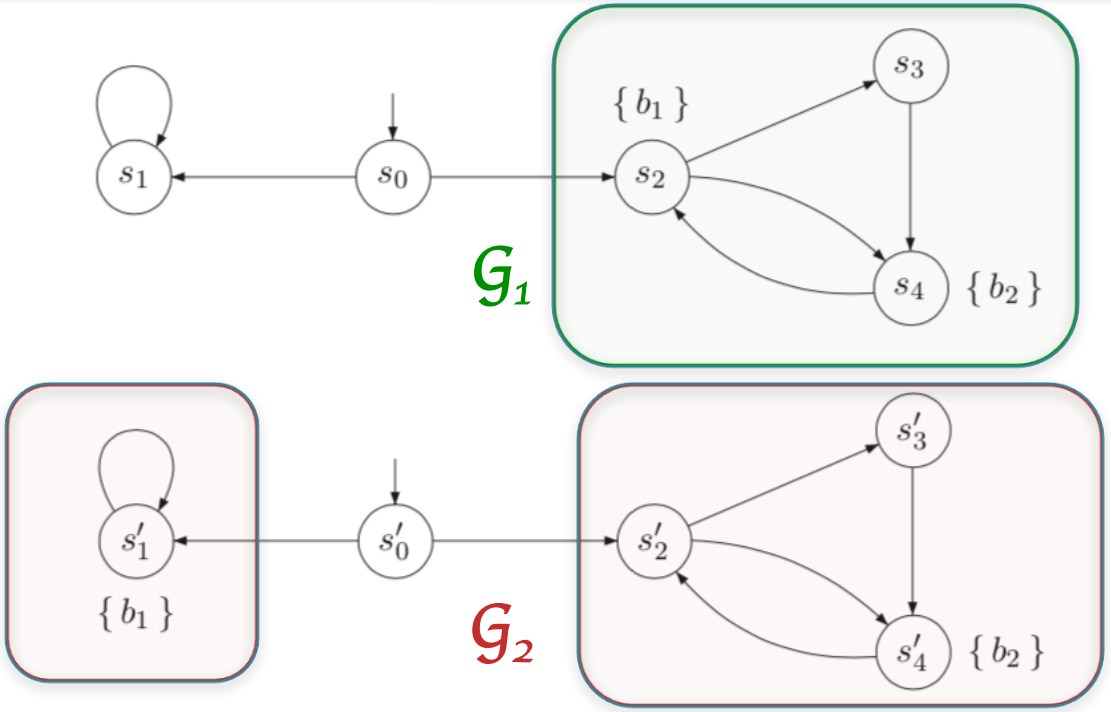
Each infinite path in G_a is a path in $\mathcal{M}$ satisfying **G** a.
Conversely, each path in $\mathcal{M}$ satisfying **G** a is a path in G_a.

$\mathcal{M}, s \models_{\text{sfair}}$ **EG** a if and only if there exists a nontrivial SCC C in G_a reachable from s and a set of nodes $D \subseteq C$ such that for all $0 \leq i \leq k$, $D \cap \text{Sat}(a_i) = \varnothing$ or $D \cap \text{Sat}(b_i) \neq \varnothing$.

$\text{Sat}_{\text{fair}}(\textbf{EG}\ a) = \{s \mid \text{exists } C \text{ reachable from } s \text{ in } G_a\}$

Unconditional Fairness: in this case, a_i is true for all i. Observe that in this case, fair path **corresponds to accepting runs of a Generalised Büchi automaton**.

Example: unconditional fairness



$\mathcal{G}_1$ satisfy unconditional fairness constraint $\mathbf{G}\,\mathbf{F}\,b_1 \wedge \mathbf{G}\,\mathbf{F}\,b_2$ **because there is the SCC $\{s_2, s_3, s_4\}$.**

By contrast, $\mathcal{G}_2$ does not satisfy $\mathbf{G}\,\mathbf{F}\,b_1 \wedge \mathbf{G}\,\mathbf{F}\,b_2$ because there is the SCC $\{s_2, s_3, s_4\}$ that contains b_2 and the SCC $\{s_1\}$ that contains only b_1, but **no one of them contains both** b_1 and b_2.

Lesson 6

That's all Folks…

…Questions?