Fairness

Lecture #7 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling and Verification

E-mail: `katoen@cs.rwth-aachen.de`

November 11, 2008

Overview Lecture #7

⇒ The Importance of Fairness

- Fairness Constraints
- Fairness Assumptions
- Fairness and Safety Properties

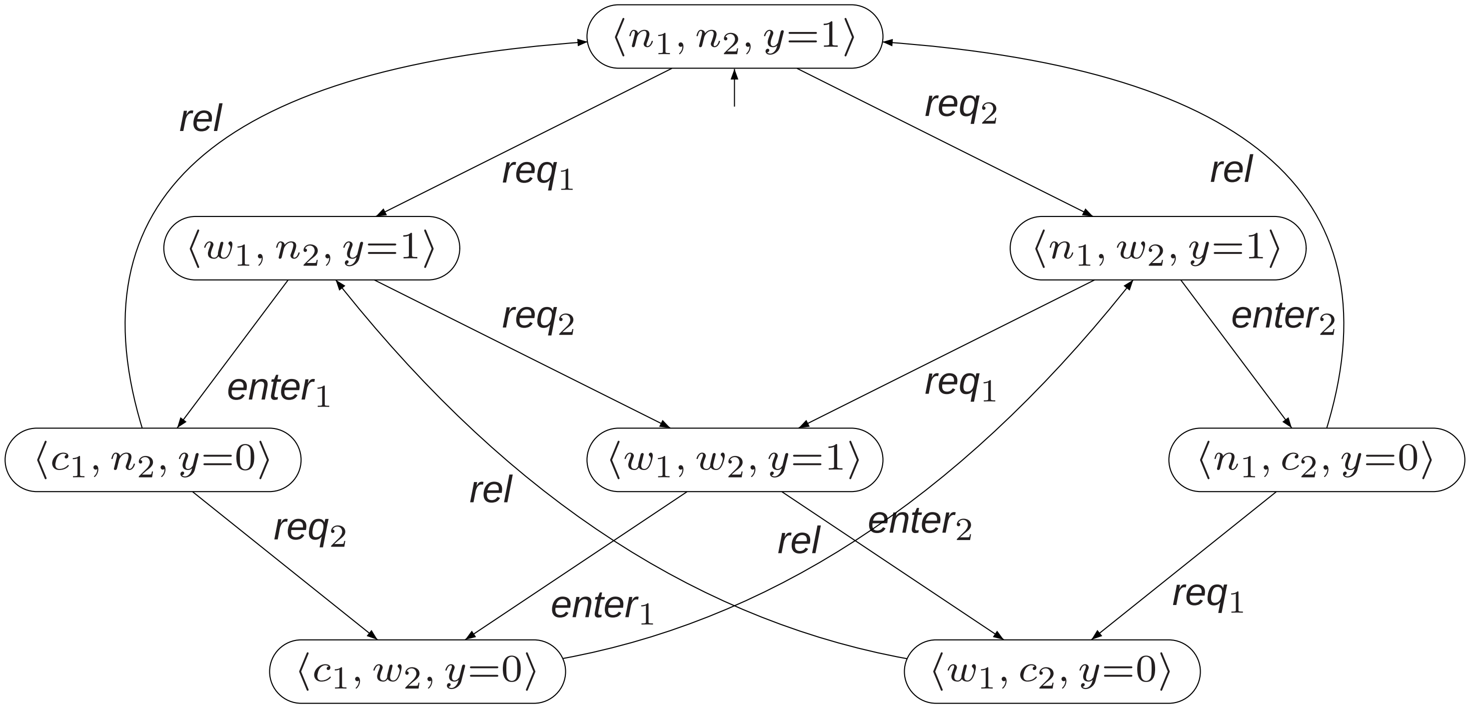
Does this program always terminate?

$$\textsf{Inc} \mid\mid\mid \textsf{Reset}$$

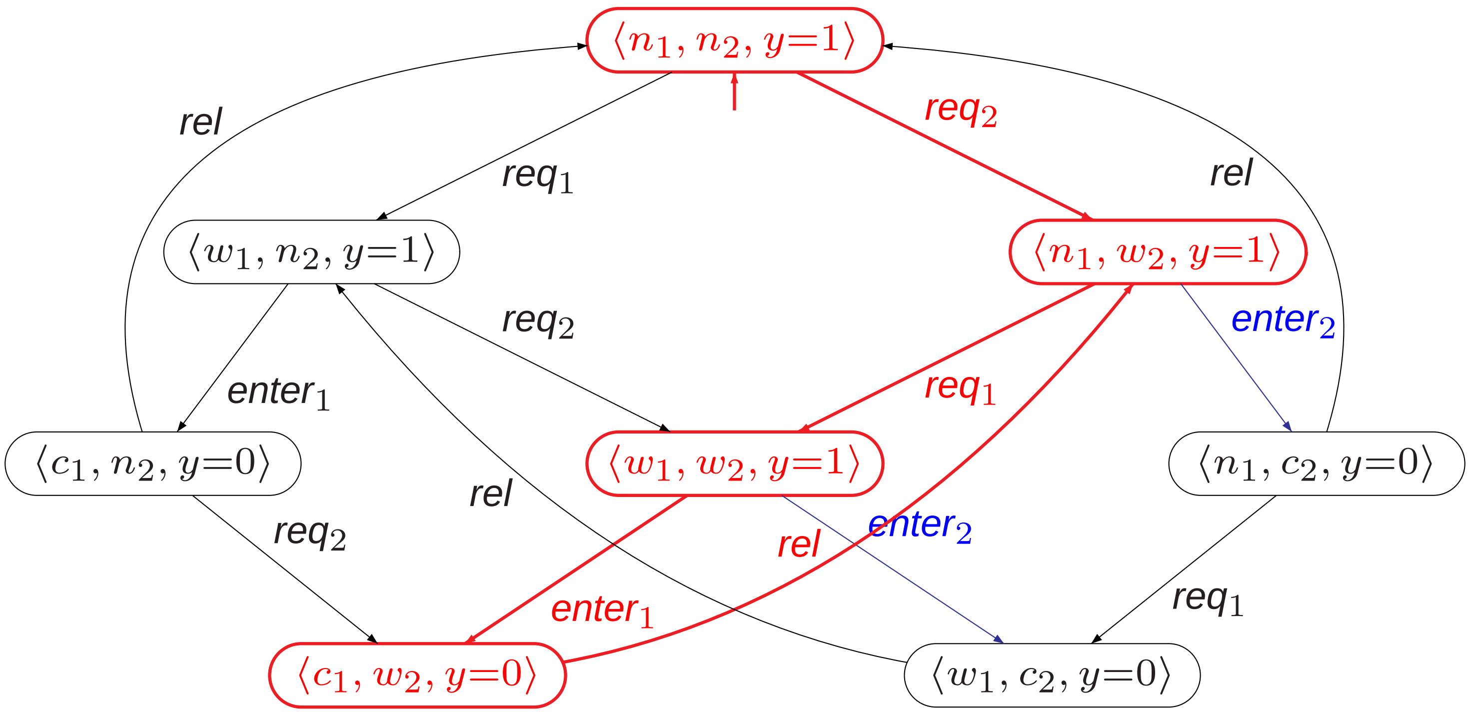
$$\begin{aligned} & where \\ \textbf{proc}\ \textsf{Inc} &= \textbf{while}\ \langle\, x \geqslant 0\ \textbf{do}\ x := x + 1\,\rangle\ \textbf{od} \\ \textbf{proc}\ \textsf{Reset} &= x := -1 \end{aligned}$$

x is a shared integer variable that initially has value 0

Is it possible to starve?

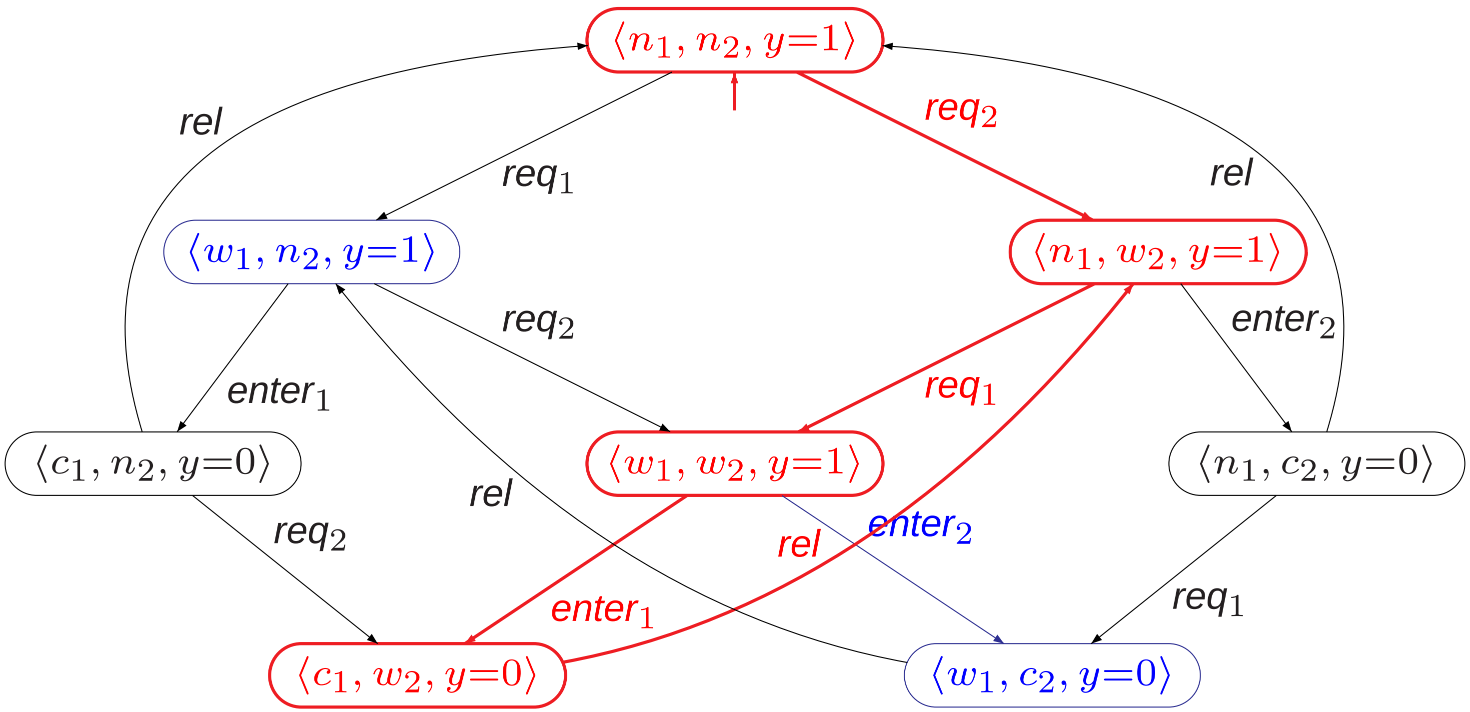


Process two starves



Is it fair that process two has infinitely many possibilities
to enter the critical section, but never enters it?

Process two starves



Is it fair that process two has infinitely many possibilities
to enter the critical section, but only enters it finitely often?

Fairness

- Starvation freedom is often considered under process fairness

  $\Rightarrow$ there is a fair scheduling of the execution of processes

- Fairness is typically needed to prove liveness
  - not for safety properties!
  - to prove some form of progress, progress needs to be possible

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

- 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 in no processor is infinitely 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 process 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 under unconditional (process) fairness:

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

x is a shared integer variable that initially has value 0

Overview Lecture #7

- The Importance of Fairness

$\Rightarrow$ Fairness Constraints

- Fairness Assumptions

- Fairness and Safety Properties

Fairness constraints

- *Unconditional fairness*

  an activity is executed infinitely often

- *Strong fairness*

  if an activity is *infinitely often* enabled (not necessarily always!)
  then it has to be executed infinitely often

- *Weak fairness*

  if an activity is *continuously enabled* (no temporary disabling!)
  then it has to be executed infinitely often

we will use actions to distinguish fair and unfair behaviours

Fairness definition

For $TS = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ without terminal states, $A \subseteq \mathit{Act}$,

and infinite execution fragment $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ of TS:

1. ρ is *unconditionally A-fair* whenever: true $\Longrightarrow \underbrace{\forall k \geqslant 0.\, \exists j \geqslant k.\, \alpha_j \in A}_{\text{infinitely often } A \text{ is taken}}$

2. ρ is *strongly A-fair* whenever:

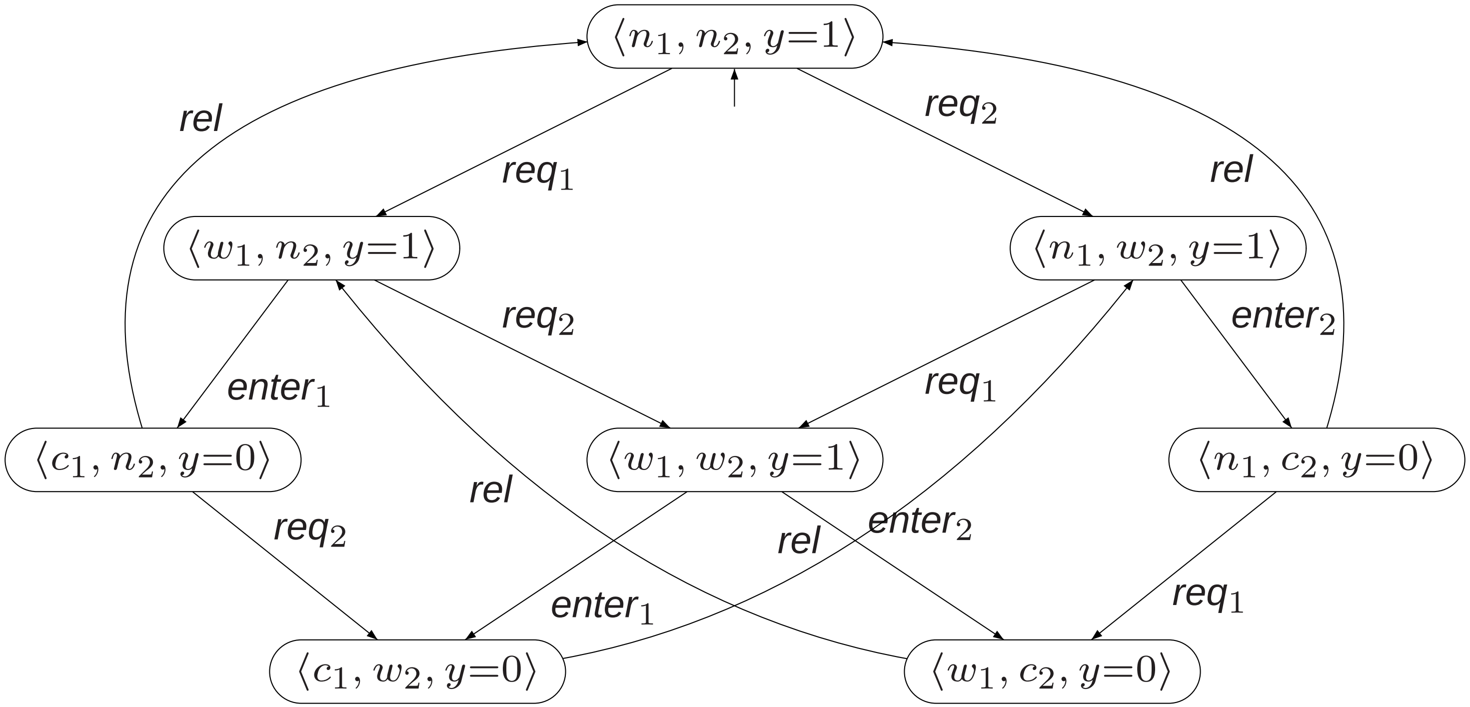
$$\underbrace{(\, \forall k \geqslant 0.\, \exists j \geqslant k.\, \mathit{Act}(s_j) \,\cap\, A \neq \varnothing \,)}_{\text{infinitely often } A \text{ is enabled}} \Longrightarrow \underbrace{\forall k \geqslant 0.\, \exists j \geqslant k.\, \alpha_j \in A}_{\text{infinitely often } A \text{ is taken}}$$

3. ρ is *weakly A-fair* whenever:

$$\underbrace{(\, \exists k \geqslant 0.\, \forall j \geqslant k.\, \mathit{Act}(s_j) \,\cap\, A \neq \varnothing \,)}_{A \text{ is eventually always enabled}} \Longrightarrow \underbrace{\forall k \geqslant 0.\, \exists j \geqslant k.\, \alpha_j \in A}_{\text{infinitely often } A \text{ is taken}}$$

where $\mathit{Act}(s) = \left\{ \alpha \in \mathit{Act} \mid \exists s' \in S.\, s \xrightarrow{\alpha} s' \right\}$

Example (un)fair executions



Which fairness notion to use?

- Fairness constraints aim to rule out "unreasonable" runs
- Too strong? $\Rightarrow$ relevant computations ruled out

  verification yields:
  - "false": error found
  - "true": don't know as some relevant execution may refute it
- Too weak? $\Rightarrow$ too many computations considered

  verification yields:
  - "true": property holds
  - "false": don't know, as refutation maybe due to some unreasonable run

Relation between fairness constraints

unconditional A-fairness $\Longrightarrow$ strong A-fairness $\Longrightarrow$ weak A-fairness

Overview Lecture #7

- The Importance of Fairness
- Fairness Constraints

$\Rightarrow$ Fairness Assumptions

- Fairness and Safety Properties

Fairness assumptions

- Fairness constraints impose a requirement on any $\alpha \in A$
- In practice: different constraints on different action sets needed
- This is realised by *fairness assumptions*

Fairness assumptions

- A *fairness assumption* for *Act* is a triple

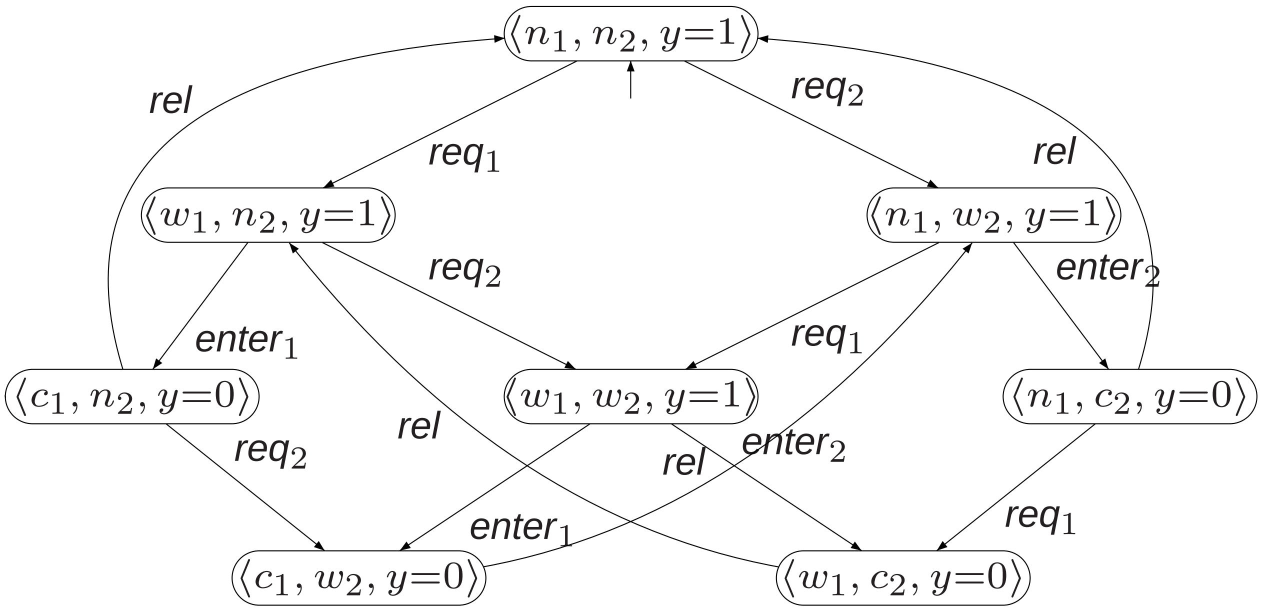
$$\mathcal{F} = (\mathcal{F}_{ucond}, \mathcal{F}_{strong}, \mathcal{F}_{weak})$$

with $\mathcal{F}_{ucond}, \mathcal{F}_{strong}, \mathcal{F}_{weak} \subseteq 2^{Act}$

- Execution ρ is $\mathcal{F}$-fair if:
  - it is unconditionally A-fair for all $A \in \mathcal{F}_{ucond}$, and
  - it is strongly A-fair for all $A \in \mathcal{F}_{strong}$, and
  - it is weakly A-fair for all $A \in \mathcal{F}_{weak}$

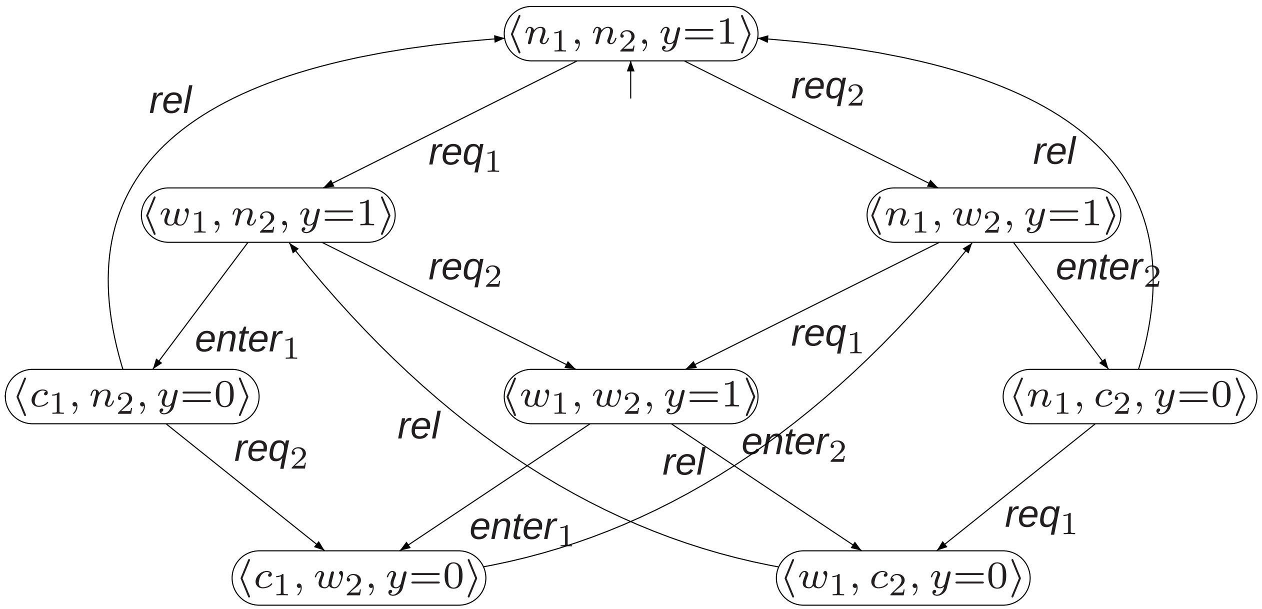
fairness assumption $(\varnothing, \mathcal{F}', \varnothing)$ denotes strong fairness; $(\varnothing, \varnothing, \mathcal{F}')$ weak, etc.

Fairness for mutual exclusion



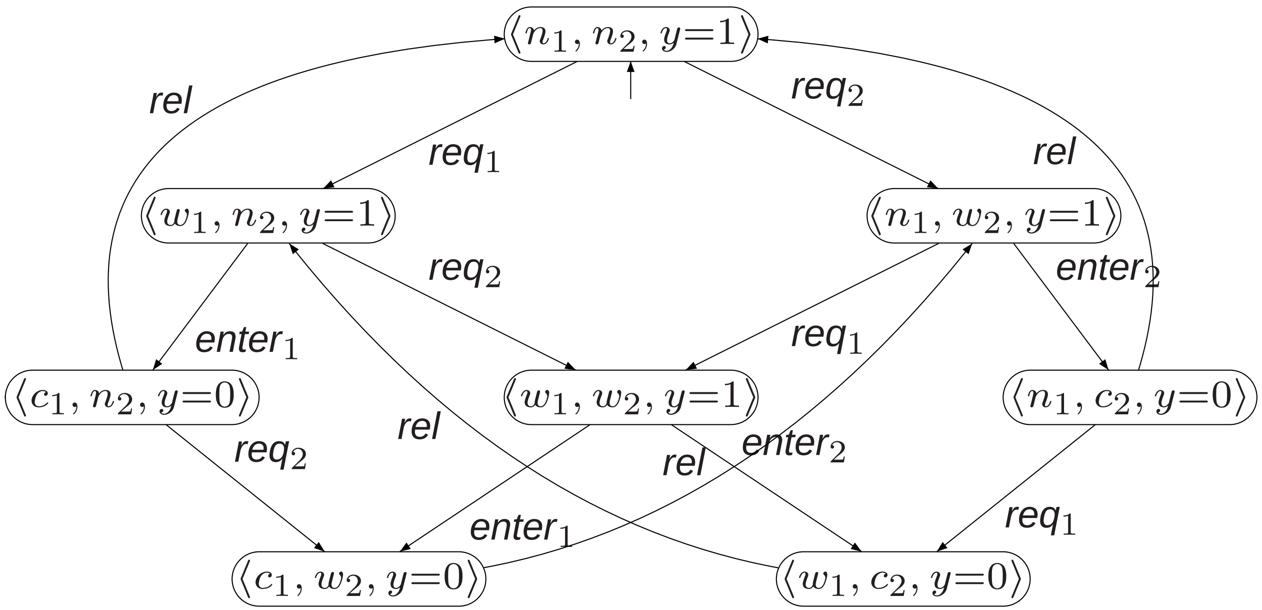
$$\mathcal{F} = (\varnothing, \underbrace{\Big\{ \{ \textit{enter}_1, \textit{enter}_2 \} \Big\}}_{\mathcal{F}_{strong}}, \varnothing)$$

Fairness for mutual exclusion



$$\mathcal{F}' = (\varnothing, \underbrace{\Big\{ \{\, \textit{enter}_1 \,\}, \{\, \textit{enter}_2 \,\} \Big\}}_{\mathcal{F}_{strong}}, \varnothing)$$

Fairness for mutual exclusion



$$\mathcal{F}'' = \left(\varnothing, \underbrace{\Big\{ \{ \mathit{enter}_1 \}, \{ \mathit{enter}_2 \} \Big\}}_{\mathcal{F}_{strong}}, \underbrace{\Big\{ \{ \mathit{req}_1 \}, \{ \mathit{req}_2 \} \Big\}}_{\mathcal{F}_{weak}} \right)$$

in any $\mathcal{F}''$-fair execution each process infinitely often requests access

Fair paths and traces

- Path $s_0 \longrightarrow s_1 \longrightarrow s_2 \ldots$ is *$\mathcal{F}$-fair* if
  - there exists an $\mathcal{F}$-fair execution $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \ldots$
  - $\mathit{FairPaths}_{\mathcal{F}}(s)$ denotes the set of $\mathcal{F}$-fair paths that start in s
  - $\mathit{FairPaths}_{\mathcal{F}}(\mathit{TS}) \; = \; \bigcup_{s \in I} \mathit{FairPaths}_{\mathcal{F}}(s)$

- Trace σ is *$\mathcal{F}$-fair* if there exists an $\mathcal{F}$-fair execution ρ with $\mathit{trace}(\rho) = \sigma$
  - $\mathit{FairTraces}_{\mathcal{F}}(s) = \mathit{trace}(\mathit{FairPaths}_{\mathcal{F}}(s))$
  - $\mathit{FairTraces}_{\mathcal{F}}(\mathit{TS}) = \mathit{trace}(\mathit{FairPaths}_{\mathcal{F}}(\mathit{TS}))$

these notions are only defined for infinite paths and traces; why?

Fair satisfaction

- *TS* *satisfies* LT-property P:

$$\textit{TS} \models P \quad \text{if and only if} \quad \textit{Traces}(\textit{TS}) \subseteq P$$

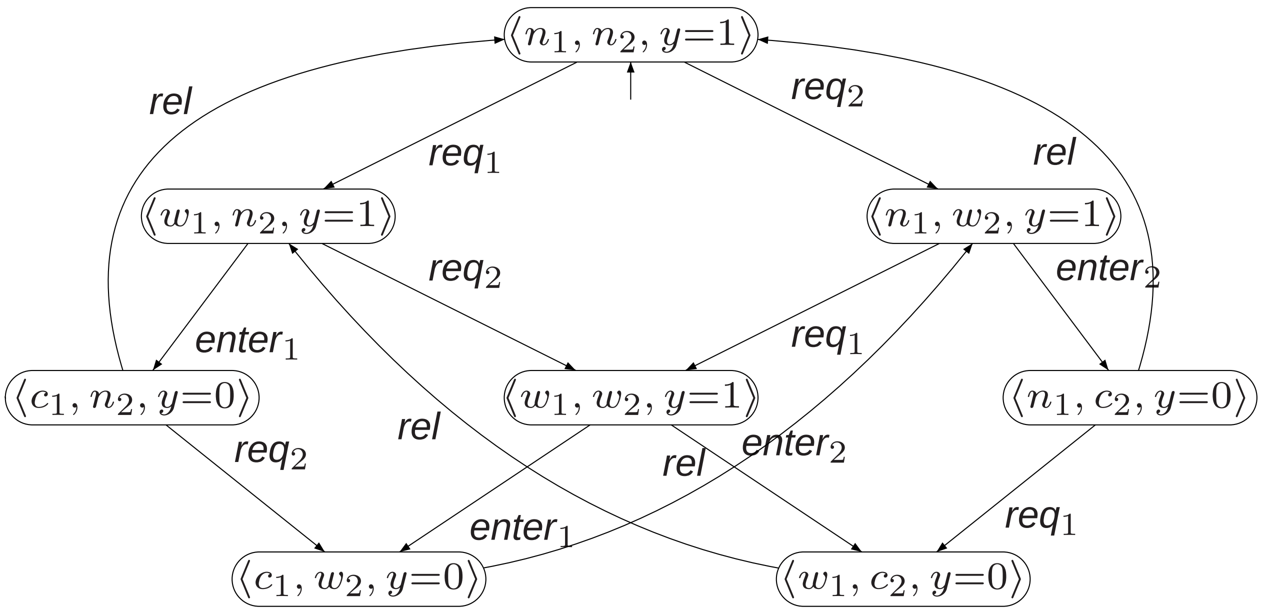
  - *TS* satisfies the LT property P if *all* its observable behaviors are admissible

- *TS* *fairly satisfies* LT-property P wrt. fairness assumption $\mathcal{F}$:

$$\textit{TS} \models_{\mathcal{F}} P \quad \text{if and only if} \quad \textit{FairTraces}_{\mathcal{F}}(\textit{TS}) \subseteq P$$

  - if all paths in *TS* are $\mathcal{F}$-fair, then *TS* $\models_{\mathcal{F}} P$ if and only if *TS* $\models P$
  - if some path in *TS* is not $\mathcal{F}$-fair, then possibly *TS* $\models_{\mathcal{F}} P$ but *TS* $\not\models P$

Fairness for mutual exclusion



TS $\not\models$ "every process enters its critical section infinitely often"

and *TS* $\not\models_{\mathcal{F}'}$ "every . . . often"

but *TS* $\models_{\mathcal{F}''}$ "every . . . often"

Overview Lecture #7

- The Importance of Fairness
- Fairness Constraints
- Fairness Assumptions

$\Rightarrow$ Fairness and Safety Properties

Realizable fairness

For *TS* with set of actions *Act* and fairness assumption $\mathcal{F}$ for *Act*:

$\mathcal{F}$ is *realizable* for *TS* if for any $s \in \mathit{Reach}(\mathit{TS})$: $\mathit{FairPaths}_{\mathcal{F}}(s) \neq \varnothing$

every initial finite execution fragment of TS can be completed to a fair execution

The suffix property

If infinite execution fragment ρ is fair

then all suffixes of ρ are fair.

If infinite execution fragment ρ is fair

then any finite execution fragment continued with ρ is fair.

$$\underbrace{s'_0 \xrightarrow{\beta_1} s'_1 \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} s'_n}_{\text{arbitrary starting fragment}} = \underbrace{s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots}_{\text{fair continuation } \rho}$$

Realizable fairness and safety

For *TS* and safety property P_{safe} (both over *AP*)

and $\mathcal{F}$ a realizable fairness assumption for *TS*:

TS $\models$ P_{safe} if and only if *TS* $\models_{\mathcal{F}}$ P_{safe}

Safety properties are thus preserved by realizable fairness assumptions

Non-realizable fairness may harm safety properties

Summary of fairness

- Fairness constraints rule out unrealistic executions
  - i.e., constraints on the actions that occur along infinite executions
  - important for the verification of liveness properties
- Unconditional, strong, and weak fairness constraints
  - unconditional $\Rightarrow$ strong fair $\Rightarrow$ weak fair
- Fairness assumptions allow distinct constraints on distinct action sets
- (Realizable) fairness assumptions are irrelevant for safety properties