Fairness

Lecture #7 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling and Verification

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

April 24, 2007

Overview Lecture #7

⇒ The Importance of Fairness

- Fairness Constraints
- Fairness Assumptions
- Fair Concurrency
- Fairness and Safety Properties

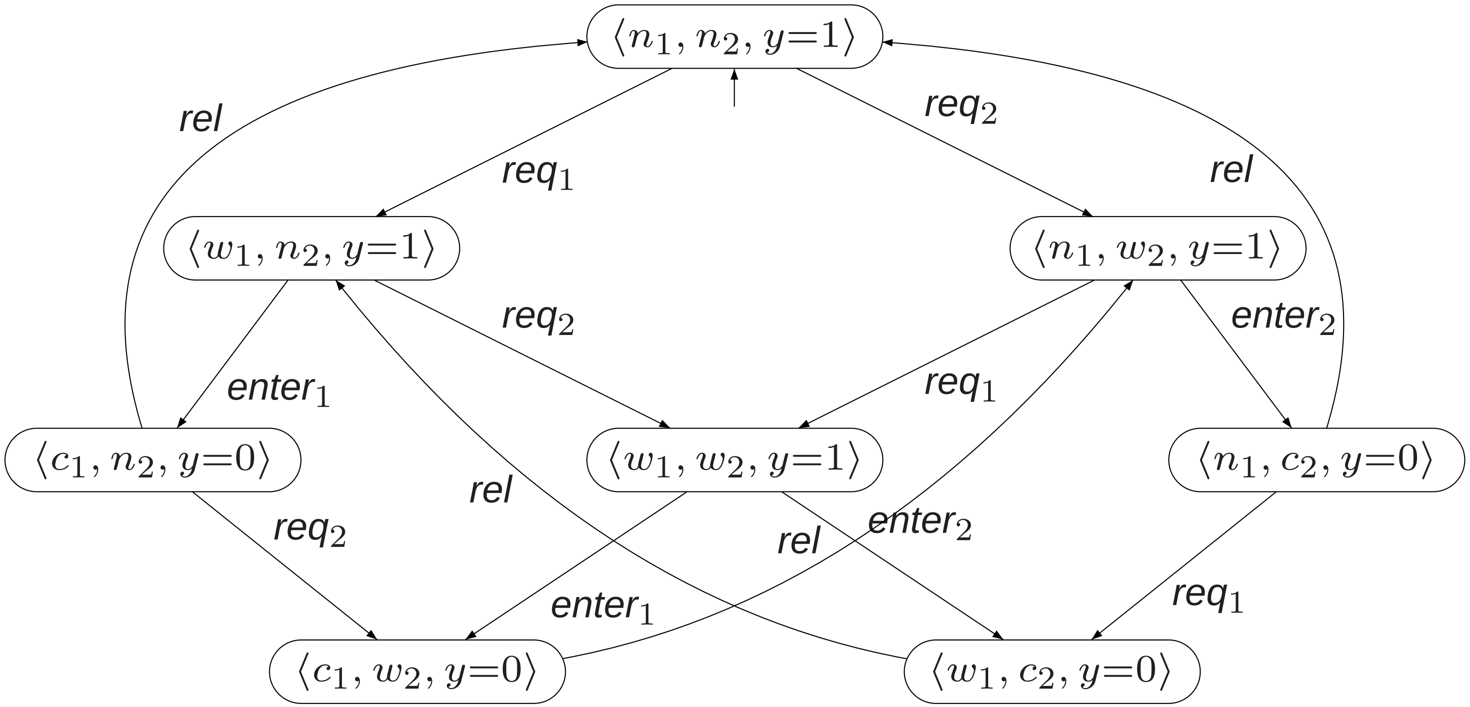
Does this program terminate?

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

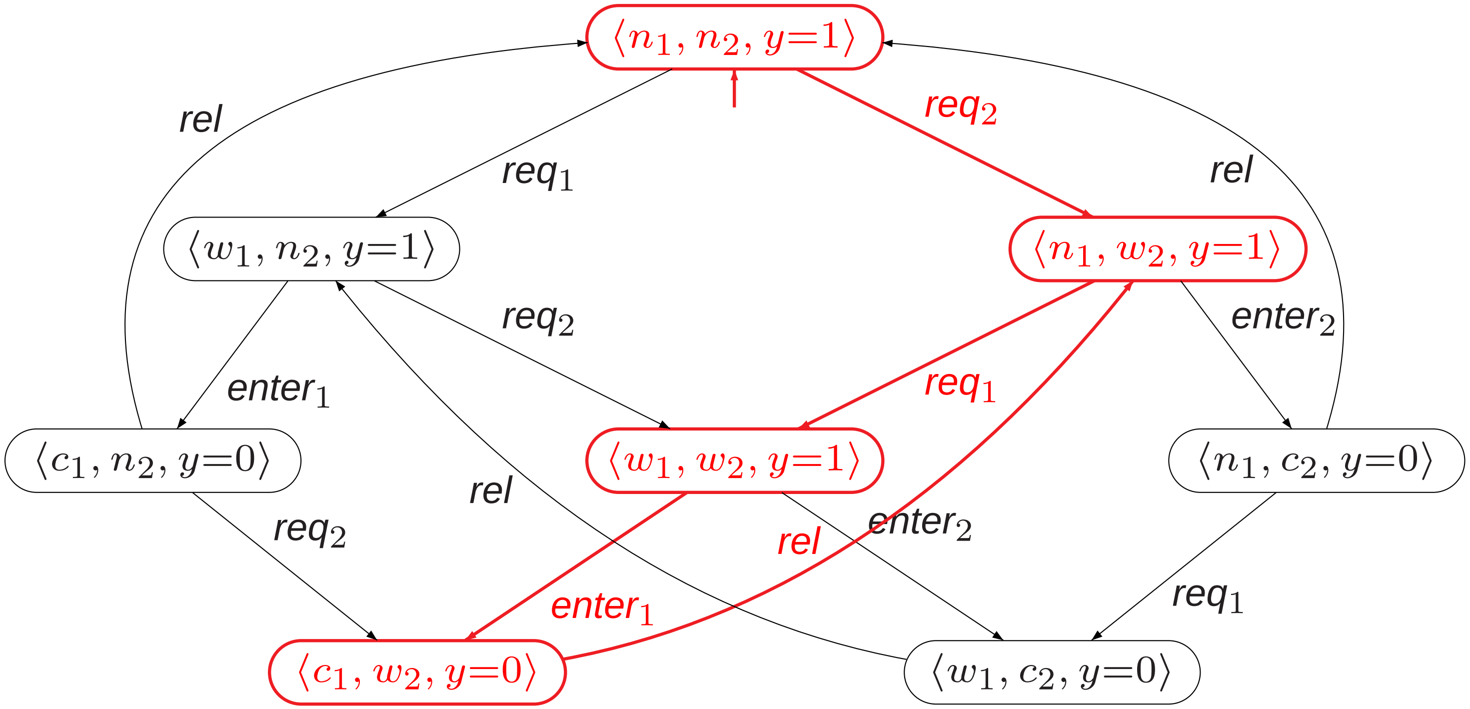
$$
\begin{aligned}
&\mathit{where} \\
\textbf{proc}\ \mathsf{Inc} &= \textbf{while}\ \langle\, x \geqslant 0\ \textbf{do}\ x := x + 1 \,\rangle\ \textbf{od} \\
\textbf{proc}\ \mathsf{Reset} &= x := -1
\end{aligned}
$$

x is a shared integer variable that initially has value 0

Do we starve?

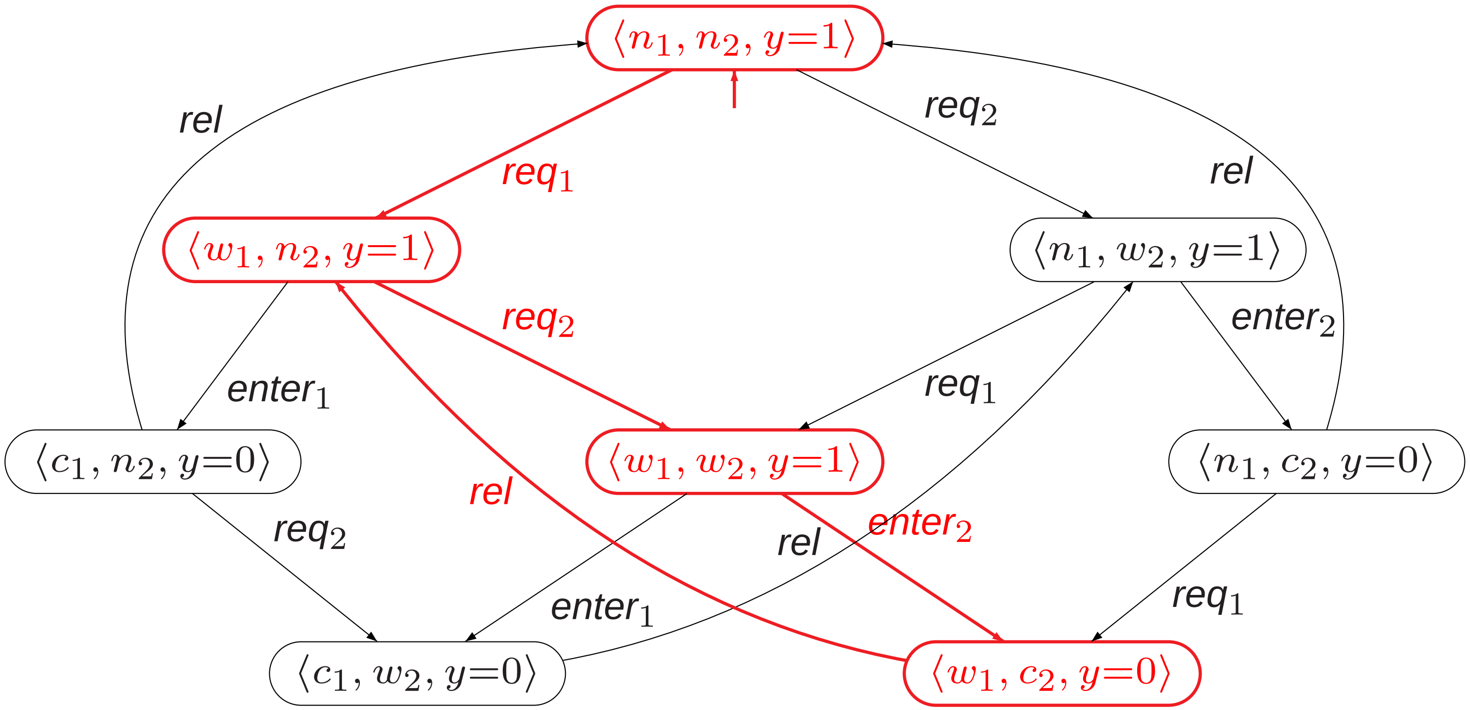

$\langle n_1, n_2, y{=}1 \rangle$
rel
req$_1$
req$_2$
rel
$\langle w_1, n_2, y{=}1 \rangle$
$\langle n_1, w_2, y{=}1 \rangle$
req$_2$
enter$_2$
enter$_1$
req$_1$
$\langle c_1, n_2, y{=}0 \rangle$
$\langle w_1, w_2, y{=}1 \rangle$
$\langle n_1, c_2, y{=}0 \rangle$
rel
req$_2$
enter$_2$
rel
enter$_1$
req$_1$
$\langle c_1, w_2, y{=}0 \rangle$
$\langle w_1, c_2, y{=}0 \rangle$

Process two starves



process two finitely many times in critical section remains unfair

Process one starves



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" runs 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 fairness:

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

x is a shared integer variable that initially has value 0

Overview Lecture #7

- The Importance of Fairness

$\Rightarrow$ Fairness Constraints

- Fairness Assumptions
- Fair Concurrency
- 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* $= (S, \mathit{Act}, \longrightarrow, I, \mathit{AP}, L)$ 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 $\implies \underbrace{\forall k \geqslant 0.\, \exists j \geqslant k.\, \alpha_j \in A}_{\text{infinitely often } A \text{ is taken}}$

2. ρ is *strongly A-fair* whenever:

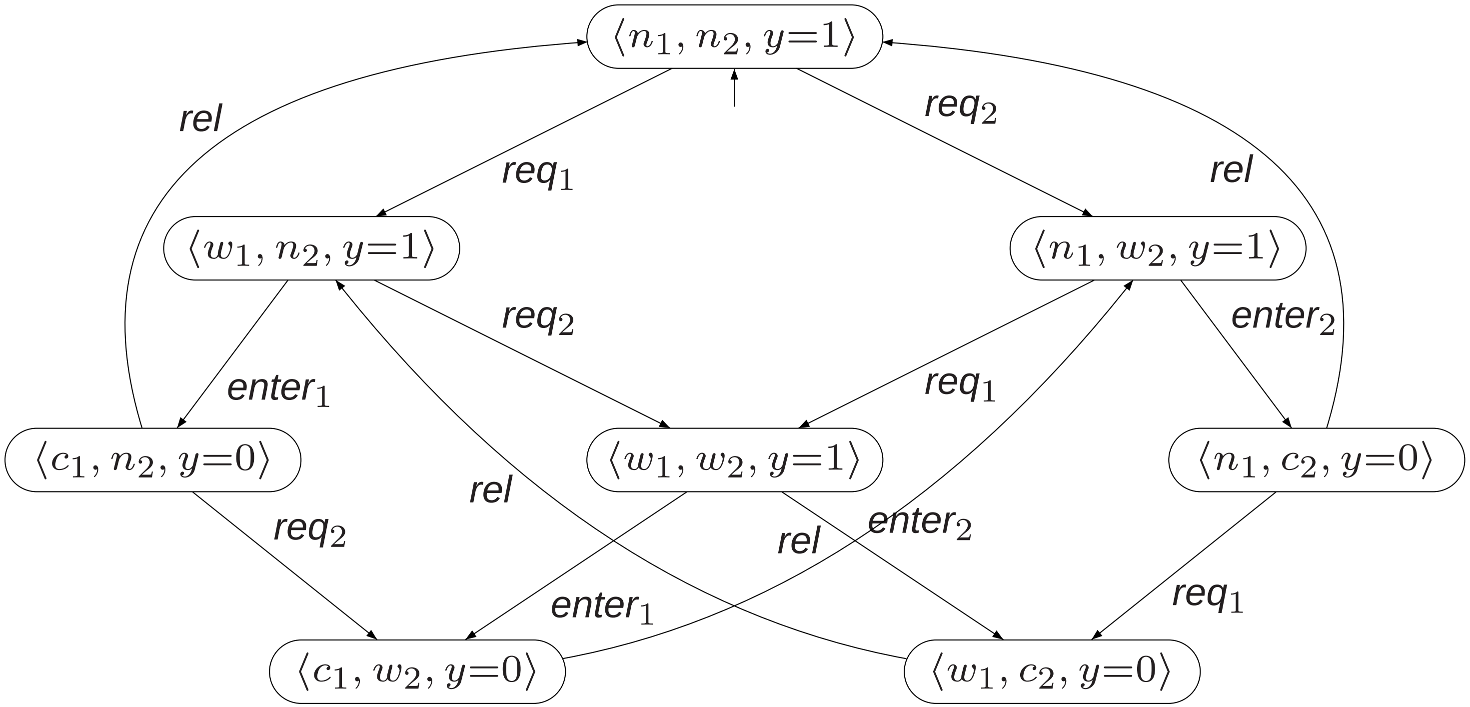
$$\underbrace{\left(\forall k \geqslant 0.\, \exists j \geqslant k.\, \mathit{Act}(s_j) \cap A \neq \varnothing\right)}_{\text{infinitely often } A \text{ is enabled}} \implies \underbrace{\left(\forall k \geqslant 0.\, \exists j \geqslant k.\, \alpha_j \in A\right)}_{\text{infinitely often } A \text{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

3. ρ is *weakly A-fair* whenever:

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

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

- Fair Concurrency
- 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

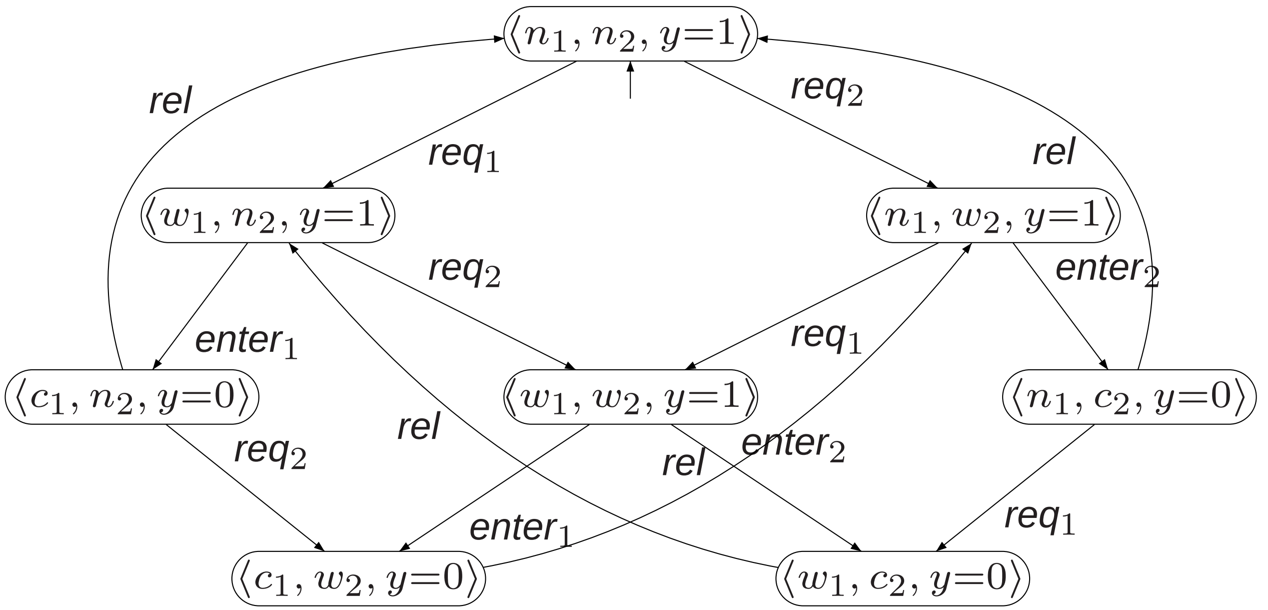
$$\mathcal{F} = (\mathcal{F}_{ucond}, \mathcal{F}_{strong}, \mathcal{F}_{weak})$$

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

- 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 $A \in \mathcal{F}_{weak}$

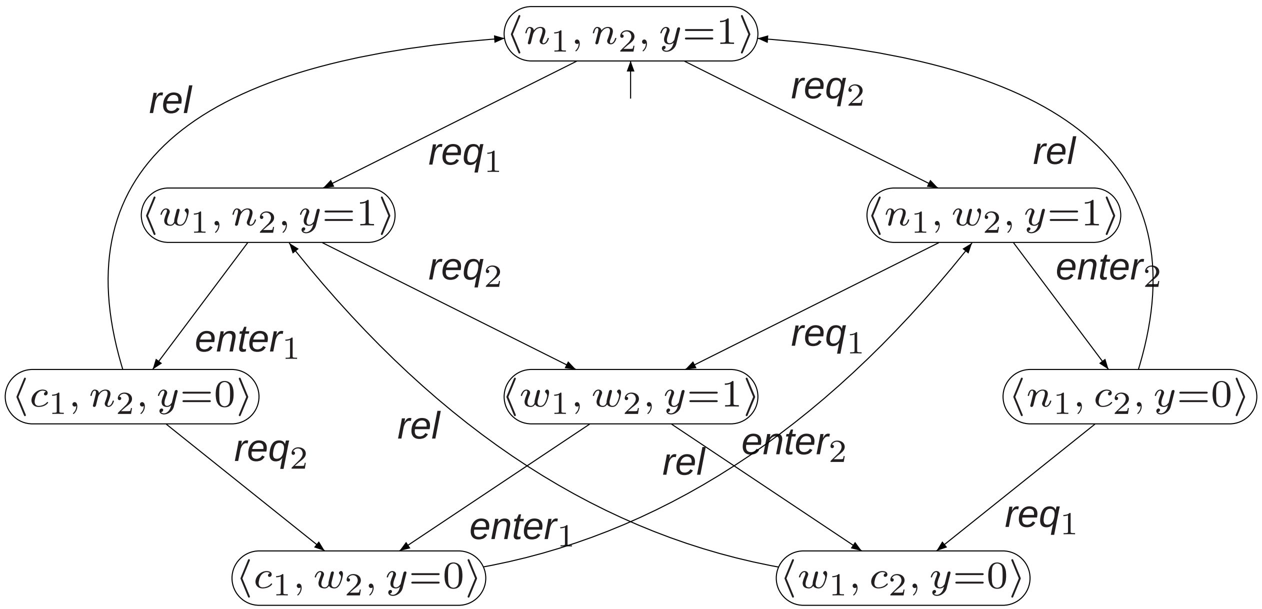
fairness assumption $(\varnothing, \mathcal{F}', \varnothing)$ denotes strong fairness; $(\varnothing, \varnothing, \mathcal{F}')$ weak, etc.

Fairness for mutual exclusion



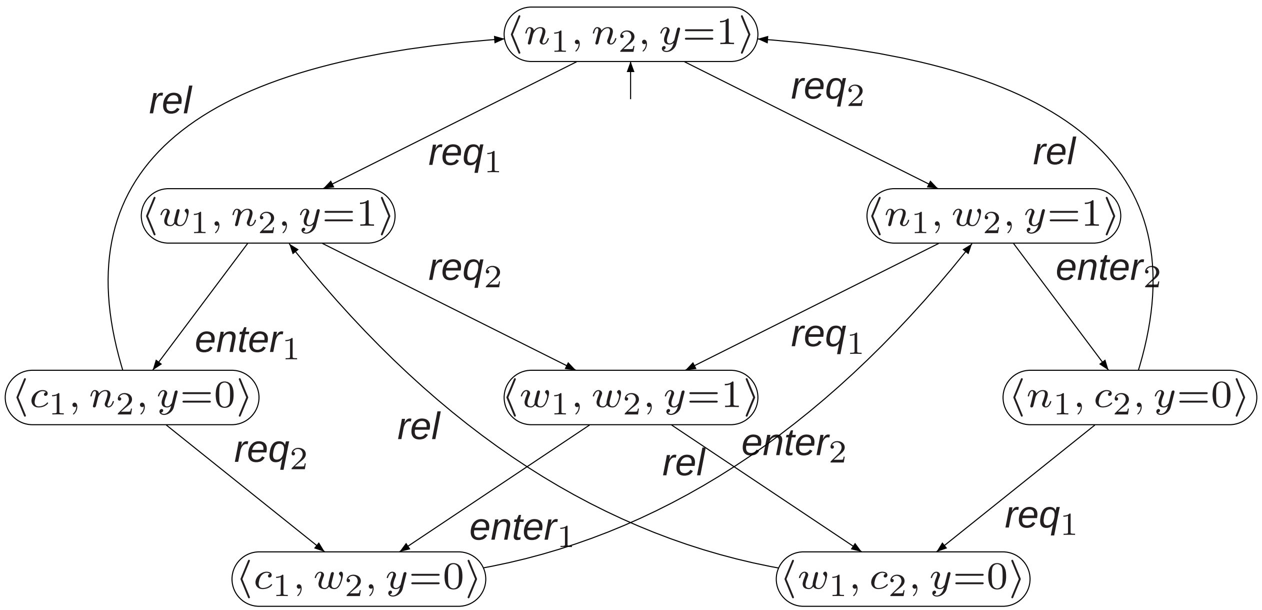
$$\mathcal{F} = (\varnothing, \underbrace{\left\{ \{ \textit{enter}_1, \textit{enter}_2 \} \right\}}_{\mathcal{F}_{strong}}, \varnothing)$$

Fairness for mutual exclusion



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

Fairness for mutual exclusion



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

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:

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

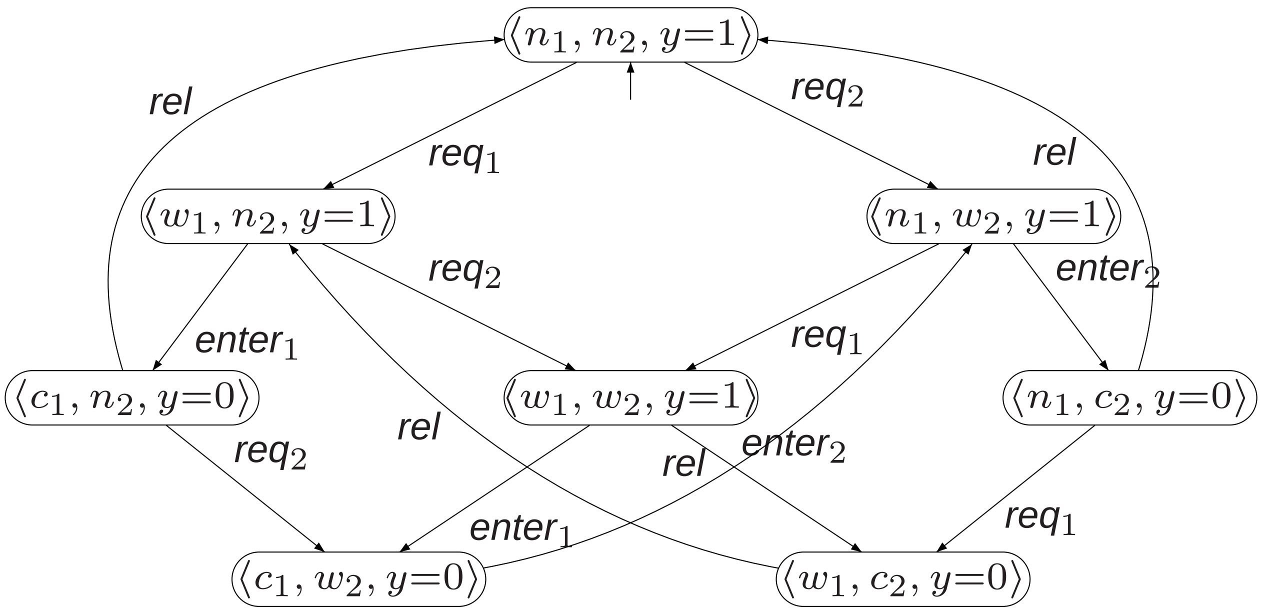
  - *TS* satisfies the LT property P if *all* its observable behaviors are admissible

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

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

  - 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

$\Rightarrow$ Fair Concurrency

- Fairness and Safety Properties

Fair concurrency with synchronization

$\mathit{TS}_i = (S_i, \mathit{Act}_i, \rightarrow_i, I_i, \mathit{AP}_i, L_i)$, for $1 \leqslant i \leqslant n$, has no terminal states

$$\mathit{TS} = \mathit{TS}_1 \parallel \mathit{TS}_2 \parallel \ldots \parallel \mathit{TS}_n$$

TS_i and TS_j $(i \neq j)$ synchronize on their common actions:

$$\mathit{Syn}_{i,j} = \mathit{Act}_i \cap \mathit{Act}_j$$

$\mathit{Syn}_{i,j} \cap \mathit{Act}_k = \varnothing$ for any $k \neq i, j$

For simplicity, it is assumed that *TS* has no terminal states

how to establish a fair communication mechanism?

Asynchronous concurrent systems

concurrency = interleaving (i.e., nondeterminism) + fairness

Some fairness assumptions

- Strong fairness constraint: $\{Act_1, Act_2, \ldots, Act_n\}$
  - TS_i executes an action (not necessarily a sync!) infinitely often provided TS is infinitely often in a (global) state with a transition of TS_i enabled
- Strong fairness constraint: $\{\, \{\, \alpha \,\} \mid \alpha \in Syn_{i,j}, 0 < i < j \leqslant n \,\}$
  - every individual synchronization is forced to happen infinitely often
- Strong fairness constraint: $\{\, Syn_{i,j} \mid 0 < i < j \leqslant n \,\}$
  - every pair of processes is forced to synchronize infinitely often
- Strong fairness constraint: $\{\, \bigcup_{0<i<j\leqslant n} Syn_{i,j} \,\}$
  - a synchronization (possibly the same) takes place infinitely often

Overview Lecture #7

- The Importance of Fairness
- Fairness Constraints
- Fairness Assumptions
- Fair Concurrency

$\Rightarrow$ 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

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

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}

Summary of fairness

- Fairness constraints rule out unrealistic traces
  - 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