Verification

Lecture 7

Martin Zimmermann

UNIVERSITÄT
DES
SAARLANDES

Plan for today

- Linear-time properties
  - Safety
  - Liveness
  - Fairness
  - Regular Properties
    - Finite automata
    - Checking regular safety properties

Summary LT properties

- LT properties are sets of infinite words over 2^{AP} (= traces)
- An invariant requires a condition Φ to hold in any reachable state
- Each trace refuting a safety property has a finite prefix causing this
  - invariants are safety properties with bad prefix $\Phi^*(\neg\Phi)$

  $\Rightarrow$ safety properties constrain finite behaviors
- A liveness property does not rule out finite behaviour

  $\Rightarrow$ liveness properties constrain infinite behaviors
- Any LT property is equivalent to a conjunction of a safety and a liveness property

Fairness

Does this program terminate?

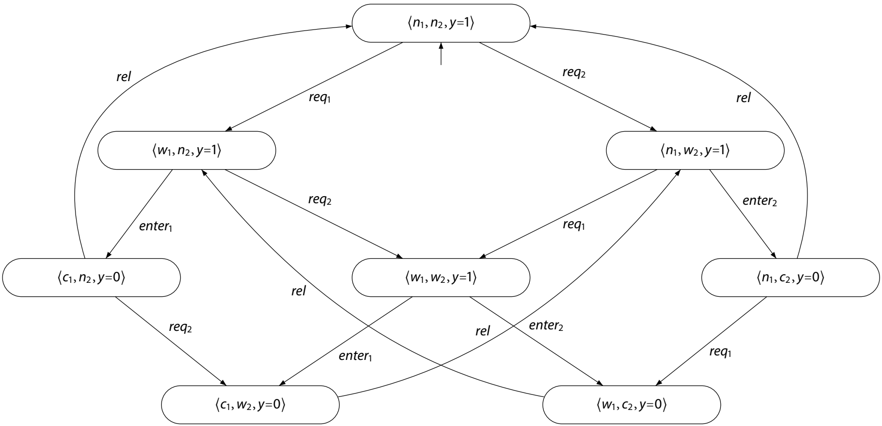
$$\text{Inc} \;|||\; \text{Reset}$$

where

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

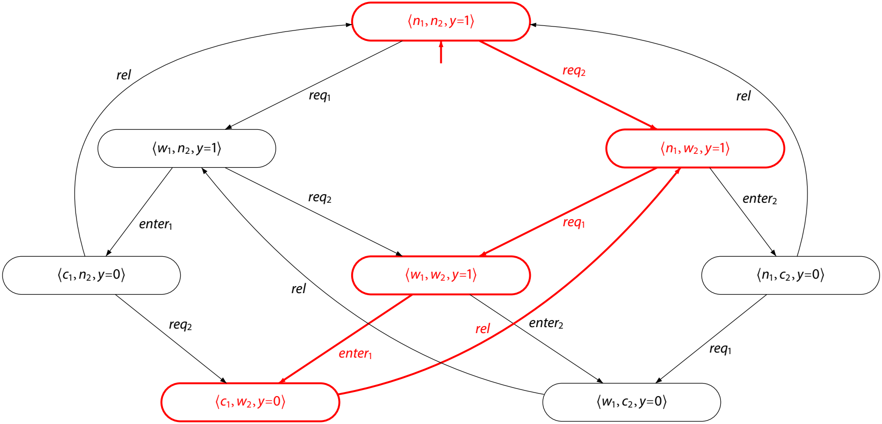
x is a shared integer variable that initially has value 0

Do we starve?



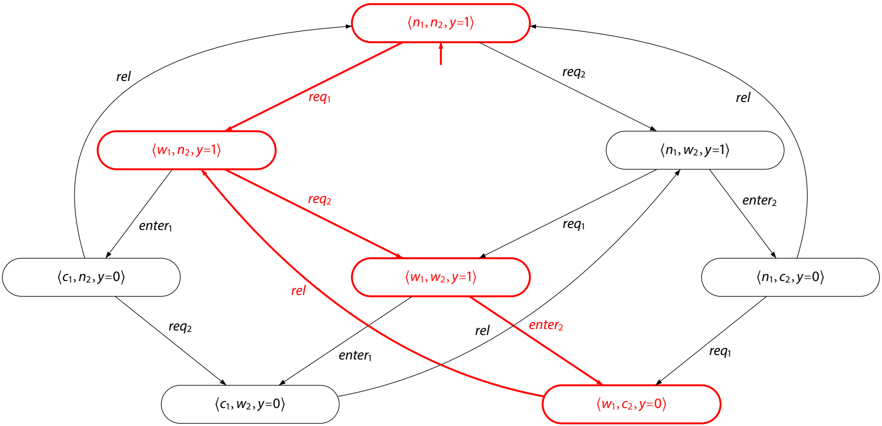
$\langle n_1, n_2, y=1 \rangle$
$\langle w_1, n_2, y=1 \rangle$
$\langle n_1, w_2, y=1 \rangle$
$\langle c_1, n_2, y=0 \rangle$
$\langle w_1, w_2, y=1 \rangle$
$\langle n_1, c_2, y=0 \rangle$
$\langle c_1, w_2, y=0 \rangle$
$\langle w_1, c_2, y=0 \rangle$
req_1
req_2
$enter_1$
$enter_2$
rel

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 reality no processor is infinitely faster than another
- 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:

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

x is a shared integer variable that initially has value 0

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

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

1. ρ is unconditionally A-fair whenever:

$$\text{true} \implies \underbrace{\forall k \geq 0.\, \exists j \geq k.\, \alpha_j \in A}_{\text{infinitely often } A \text{ is taken}}$$

2. ρ is strongly A-fair whenever:

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

3. ρ is weakly A-fair whenever:

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

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

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

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

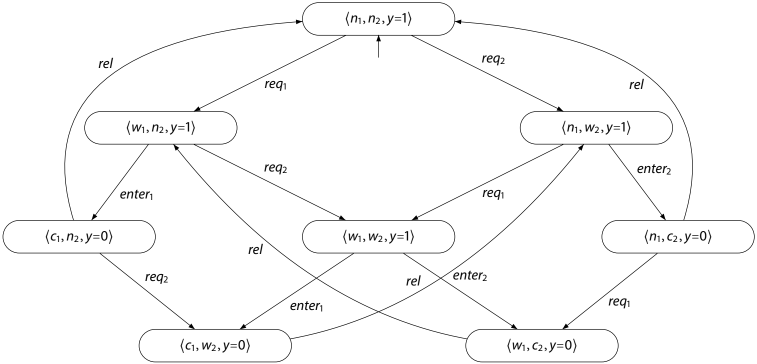
- A fairness assumption for *Act* is a triple

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

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

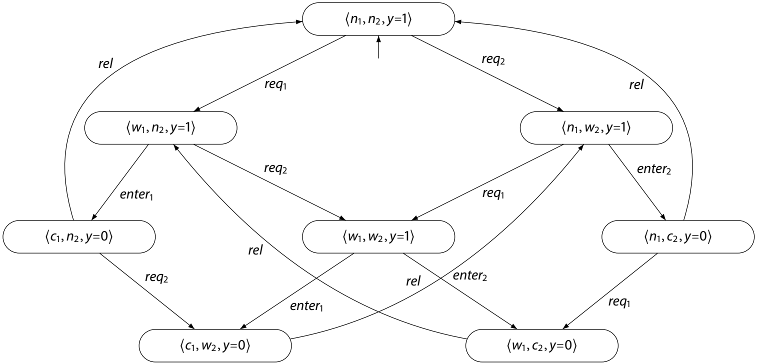
fairness assumption $(\varnothing, \mathcal{F}', \varnothing)$ denotes strong fairness; $(\varnothing, \varnothing, \mathcal{F}')$ weak, etc.

Fairness for mutual exclusion



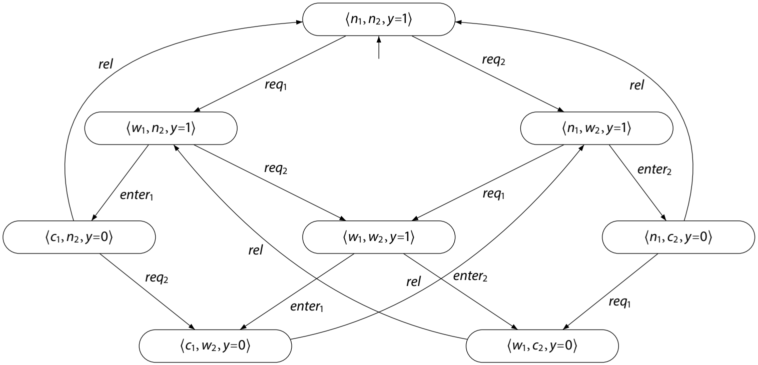
$$\mathcal{F} = (\varnothing, \underbrace{\{\{\, enter_1, enter_2 \,\}\}}_{\mathcal{F}_{strong}}, \varnothing)$$

Fairness for mutual exclusion



$$\mathcal{F} = (\varnothing, \underbrace{\{\{\, enter_1 \,\}, \{\, enter_2 \,\}\}}_{\mathcal{F}_{strong}}, \varnothing)$$

Fairness for mutual exclusion



$$\mathcal{F}' = \Big(\varnothing, \underbrace{\{\{\,enter_1\,\},\{\,enter_2\,\}\}}_{\mathcal{F}_{strong}}, \underbrace{\{\{\,req_1\,\},\{\,req_2\,\}\}}_{\mathcal{F}_{weak}}\Big)$$

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

Fair paths and traces

- Path $s_0 \to s_1 \to s_2 \dots$ is $\mathcal{F}$-fair if
  - there exists an $\mathcal{F}$-fair execution $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \dots$
  - $\mathit{FairPaths}_{\mathcal{F}}(s)$ denotes the set of $\mathcal{F}$-fair paths that start in s
  - $\mathit{FairPaths}_{\mathcal{F}}(TS) = \bigcup_{s \in I} \mathit{FairPaths}_{\mathcal{F}}(s)$
- Trace σ is $\mathcal{F}$-fair if there exists an $\mathcal{F}$-fair execution ρ with $\mathit{trace}(\rho) = \sigma$
  - $\mathit{FairTraces}_{\mathcal{F}}(s) = \mathit{trace}(\mathit{FairPaths}_{\mathcal{F}}(s))$
  - $\mathit{FairTraces}_{\mathcal{F}}(TS) = \mathit{trace}(\mathit{FairPaths}_{\mathcal{F}}(TS))$

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

Fair satisfaction

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

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

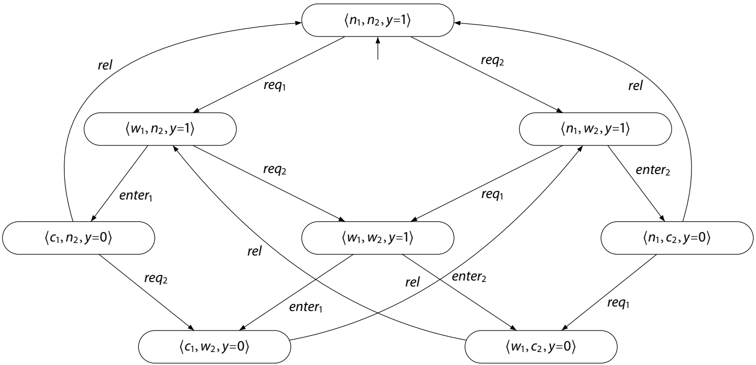
  - *TS* satisfies the LT property *P* if all its observable behaviors are admissible

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

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

  - 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"

Fair concurrency with synchronization

- $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$, for $1 \leq i \leq n$, has no terminal states
- TS_i and TS_j ($i \neq j$) synchronize on their common actions:

$$Syn_{i,j} = Act_i \cap Act_j$$

State space of $TS_1 \| \ldots \| TS_n$ is the Cartesian product of those of TS_i

- for $\alpha \in Act_i \setminus \Big(\bigcup\limits_{\substack{0<j\leq n \\ i\neq j}} Syn_{i,j} \Big)$ and $0 < i \leq n$:

$$\frac{s_i \xrightarrow{\alpha}_i s_i'}{\langle s_1, \ldots, s_i, \ldots, s_n \rangle \xrightarrow{\alpha} \langle s_1, \ldots, s_i', \ldots s_n \rangle}$$

- for $\alpha \in Syn_{i,j}$ and $0 < i < j \leq n$:

$$\frac{s_i \xrightarrow{\alpha}_i s_i' \quad \wedge \quad s_j \xrightarrow{\alpha}_j s_j'}{\langle s_1, \ldots, s_i, \ldots, s_j, \ldots, s_n \rangle \xrightarrow{\alpha} \langle s_1, \ldots, s_i', \ldots, s_j', \ldots, s_n \rangle}$$

Asynchronous concurrent systems

$$\boxed{\text{concurrency} \;=\; \text{interleaving (i.e., nondeterminism)} \;+\; \text{fairness}}$$

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: $\{\, \{\, \alpha \,\} \mid \alpha \in Syn_{i,j}, 0 < i < j \leq n \,\}$
  - every individual synchronization is forced to happen infinitely often
- Strong fairness constraint: $\{\, Syn_{i,j} \mid 0 < i < j \leq n \,\}$
  - every pair of processes is forced to synchronize infinitely often
- Strong fairness constraint: $\{\, \bigcup_{0<i<j\leq n} Syn_{i,j} \,\}$
  - a synchronization (possibly the same) takes place infinitely often

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 $s \in Reach(TS)$: $FairPaths_{\mathcal{F}}(s) \neq \varnothing$

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

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

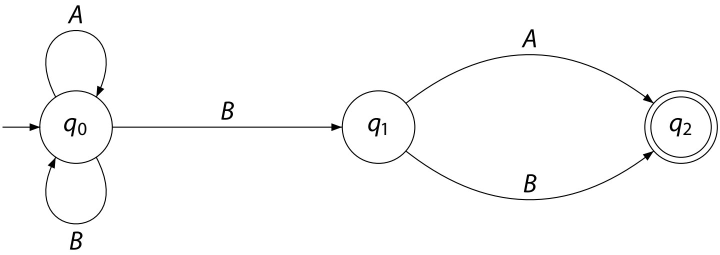
- Fairness constraints rule out unrealistic traces
  - i.e., constraints on the actions that occur along infinite executions
  - important for the verification of liveness properties
- Unconditional, strong, and weak fairness constraints
  - unconditional $\Rightarrow$ strong fair $\Rightarrow$ weak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

Regular properties

Finite automata

A nondeterministic finite automaton (NFA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, Q_0, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function
- $Q_0 \subseteq Q$ a set of initial states
- $F \subseteq Q$ is a set of accept (or: final) states



Size of an NFA

The size of $\mathcal{A}$, denoted $|\mathcal{A}|$, is the number of states and transitions in $\mathcal{A}$:

$$|\mathcal{A}| = |Q| + \sum_{q \in Q} \sum_{A \in \Sigma} |\,\delta(q, A)\,|$$

Language of an automaton

- NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ and word $w = A_1 \dots A_n \in \Sigma^*$
- A *run* for w in $\mathcal{A}$ is a finite sequence $q_0\, q_1\, \dots\, q_n$ such that:
  - $q_0 \in Q_0$ and $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leq i < n$
- Run $q_0\, q_1\, \dots\, q_n$ is accepting if $q_n \in F$
- $w \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if there exists an accepting run for w
- The accepted language of $\mathcal{A}$:

$$\mathcal{L}(\mathcal{A}) = \left\{ w \in \Sigma^* \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \right\}$$

- NFA $\mathcal{A}$ and $\mathcal{A}'$ are equivalent if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$

Accepted language revisited

Extend the transition function δ to $\delta^* : Q \times \Sigma^* \to 2^Q$ by:

$$\delta^*(q, \varepsilon) = \{\, q \,\} \quad \text{and} \quad \delta^*(q, A) = \delta(q, A)$$

$$\delta^*(q, A_1 A_2 \ldots A_n) = \bigcup_{p \in \delta(q, A_1)} \delta^*(p, A_2 \ldots A_n)$$

$\delta^*(q, w)$ = set of states reachable from q for the word w

Then: $\mathcal{L}(\mathcal{A}) = \left\{ w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \varnothing \text{ for some } q_0 \in Q_0 \right\}$

The class of languages accepted by NFA (over Σ)

= the class of regular languages (over Σ)