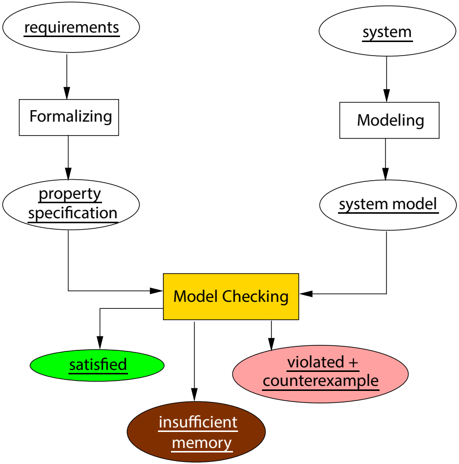
Verification

Lecture 6

Martin Zimmermann

UNIVERSITÄT
DES
SAARLANDES

REVIEW: model checking



Plan for today

- Linear-time properties
  - Safety
  - Liveness
  - Fairness

Linear-Time Properties

REVIEW: executions

- A finite execution fragment ρ of TS is an alternating sequence of states and actions ending with a state:

$$\rho \ = \ s_0 \, \alpha_1 \, s_1 \, \alpha_2 \, \ldots \alpha_n \, s_n \text{ such that } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ for all } 0 \leq i < n.$$

- An infinite execution fragment ρ of TS is an infinite, alternating sequence of states and actions:

$$\rho \ = \ s_0 \, \alpha_1 \, s_1 \, \alpha_2 \, s_2 \, \alpha_3 \ldots \text{ such that } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ for all } 0 \leq i.$$

- An execution of TS is an initial, maximal execution fragment
  - a maximal execution fragment is either finite ending in a terminal state, or infinite
  - an execution fragment is initial if $s_0 \in I$

State graph

- The state graph of TS, notation $G(TS)$, is the digraph (V, E)

  with vertices $V = S$ and edges $E = \{(s, s') \in S \times S \mid s' \in Post(s)\}$

  $\Rightarrow$ omit all state and transition labels in TS and ignore being initial

- $Post^*(s)$ is the set of states reachable in $G(TS)$ from s

$$Post^*(C) = \bigcup_{s \in C} Post^*(s) \quad \text{for } C \subseteq S$$

- The notations $Pre^*(s)$ and $Pre^*(C)$ have analogous meaning
- The set of reachable states: $Reach(TS) = Post^*(I)$

Path fragments

- A path fragment is an execution fragment without actions
- A finite path fragment $\widehat{\pi}$ of TS is a state sequence:

$$\widehat{\pi} = s_0\, s_1 \ldots s_n \quad \text{such that} \quad s_{i+1} \in Post(s_i) \text{ for all } 0 \leq i < n \text{ where } n \geq 0$$

- An infinite path fragment π of TS is an infinite state sequence:

$$\pi = s_0\, s_1\, s_2 \ldots \quad \text{such that } s_{i+1} \in Post(s_i) \text{ for all } i \geq 0$$

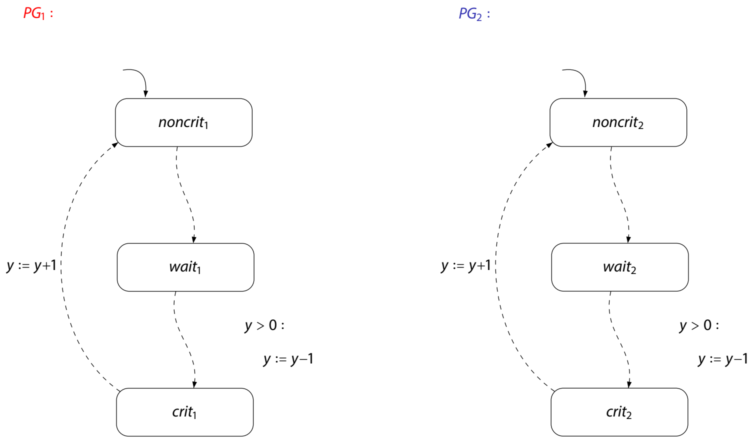
- A path of TS is an initial, maximal path fragment
  - a maximal path fragment is either finite ending in a terminal state, or infinite
  - a path fragment is initial if $s_0 \in I$
  - $Paths(s)$ is the set of maximal path fragments π with $first(\pi) = s$

Traces

States themselves are not "observable", but just their atomic propositions

- Let transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states
  - all maximal paths (and excutions) are infinite
- The trace of path fragment $\pi = s_0\, s_1 \ldots$ is $trace(\pi) = L(s_0)\, L(s_1) \ldots$
  - the trace of $\widehat{\pi} = s_0\, s_1 \ldots s_n$ is $trace(\widehat{\pi}) = L(s_0)\, L(s_1) \ldots L(s_n)$
- The set of traces of a set Π of paths: $trace(\Pi) = \{\, trace(\pi) \mid \pi \in \Pi \,\}$
- $Traces(s) = trace(Paths(s))$ $\qquad Traces(TS) = \bigcup_{s \in I} Traces(s)$
- $Traces_{fin}(s) = trace(Paths_{fin}(s))$ $\qquad Traces_{fin}(TS) = \bigcup_{s \in I} Traces_{fin}(s)$

Semaphore-based mutual exclusion



y=0 means "lock is currently possessed"; y=1 means "lock is free"

Interleaving of transition systems

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$ i=1, 2, be two transition systems.

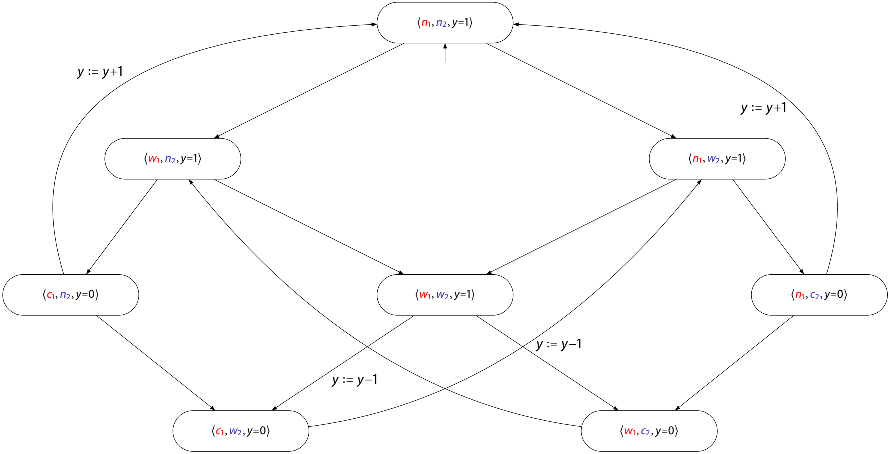
Transition system

$$TS_1 \mathbin{|||} TS_2 \;=\; (S_1 \times S_2, Act_1 \uplus Act_2, \rightarrow, I_1 \times I_2, AP_1 \uplus AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and the transition relation $\rightarrow$ is defined by the rules:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \text{and} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Transition system $TS(PG_1 \,|||\, PG_2)$



Example traces

Let $AP = \{ crit_1, crit_2 \}$
Example path:

$$\begin{aligned} \pi \quad = \quad & \langle n_1, n_2, y = 1 \rangle \rightarrow \langle w_1, n_2, y = 1 \rangle \rightarrow \langle c_1, n_2, y = 0 \rangle \rightarrow \\ & \langle n_1, n_2, y = 1 \rangle \rightarrow \langle n_1, w_2, y = 1 \rangle \rightarrow \langle n_1, c_2, y = 0 \rangle \rightarrow \ldots \end{aligned}$$

The trace of this path is the infinite word:

$$trace(\pi) \;=\; \varnothing\varnothing\{ crit_1 \}\varnothing\varnothing\{ crit_2 \}\varnothing\varnothing\{ crit_1 \}\varnothing\varnothing\{ crit_2 \}\ldots$$

The trace of the finite path fragment:

$$\begin{aligned} \widehat{\pi} \quad = \quad & \langle n_1, n_2, y = 1 \rangle \rightarrow \langle w_1, n_2, y = 1 \rangle \rightarrow \langle w_1, w_2, y = 1 \rangle \rightarrow \\ & \langle w_1, c_2, y = 0 \rangle \rightarrow \langle w_1, n_2, y = 1 \rangle \rightarrow \langle c_1, n_2, y = 0 \rangle \end{aligned}$$

is:

$$trace(\widehat{\pi}) \;=\; \varnothing\varnothing\varnothing\{ crit_2 \}\varnothing\{ crit_1 \}$$

Linear-time properties

- Linear-time properties specify the traces that a TS may exhibit
  - LT-property specifies the admissible behaviour of system under consideration

    later, a logic will be introduced for specifying LT properties
- A linear-time property (LT property) over *AP* is a subset of $\left(2^{AP}\right)^{\omega}$
  - finite words are not needed, as it is assumed that there are no terminal states
- *TS* (over *AP*) satisfies LT property *P* (over *AP*):

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

  - *TS* satisfies the LT property *P* if all its "observable" behaviors are admissible
  - state $s \in S$ satisfies *P*, notation $s \models P$, whenever $Traces(s) \subseteq P$

How to specify mutual exclusion?

"Always at most one process is in its critical section"

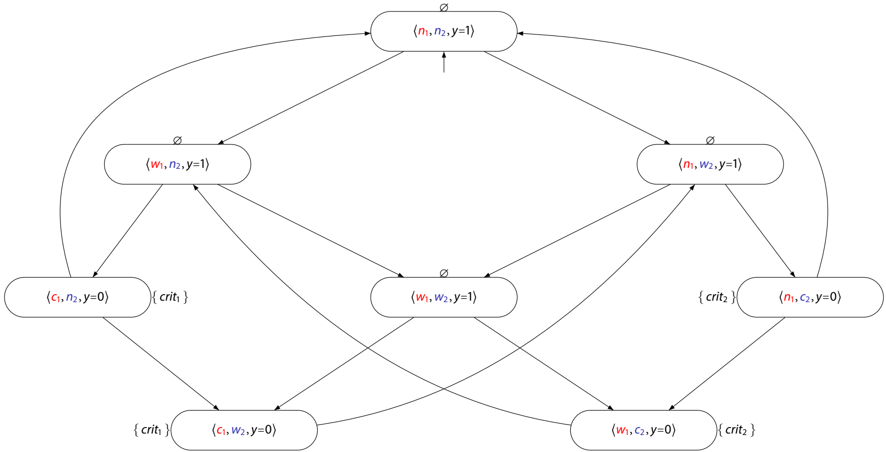
- Let $AP = \{ crit_1, crit_2 \}$
  - other atomic propositions are not of any relevance for this property
- Formalization as LT property

$$P_{mutex} = \text{set of infinite words } A_0 \, A_1 \, A_2 \ldots \text{ with } \{ crit_1, crit_2 \} \not\subseteq A_i \text{ for all } 0 \leq i$$

- Contained in P_{mutex} are e.g., the infinite words:
  - $(\{ crit_1 \} \{ crit_2 \})^\omega$ and $\{ crit_1 \} \{ crit_1 \} \{ crit_1 \} \ldots$ and $\varnothing \varnothing \varnothing \ldots$
  - but not $\{ crit_1 \} \varnothing \{ crit_1, crit_2 \} \ldots$ or $\varnothing \{ crit_1 \}, \varnothing \varnothing \{ crit_1, crit_2 \} \varnothing \ldots$

Does the semaphore-based algorithm satisfy P_{mutex}?

Does the semaphore-based algorithm satisfy P_{mutex}?



Yes as there is no reachable state labeled with $\{ \, crit_1, crit_2 \, \}$

How to specify starvation freedom?

"A process that wants to enter the critical section is eventually able to do so'"

- Let $AP = \{ wait_1, crit_1, wait_2, crit_2 \}$
- Formalization as LT-property

$P_{nostarve}$ = set of infinite words $A_0\, A_1\, A_2 \ldots$ such that:

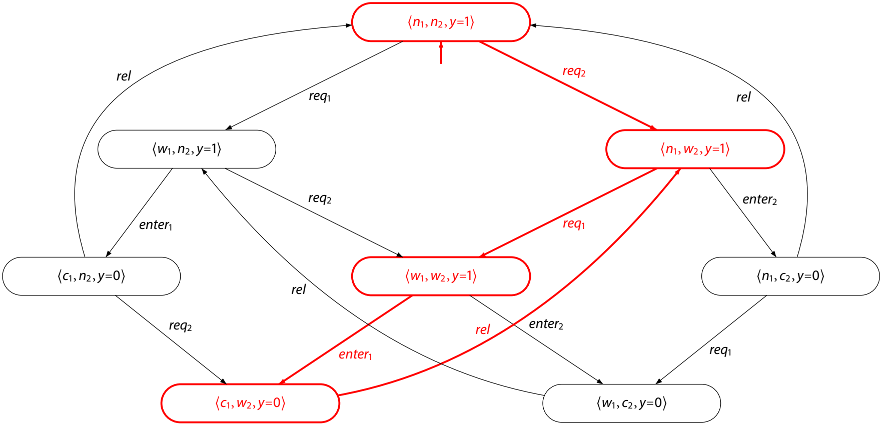
$$\left(\overset{\infty}{\exists}\, j.\, wait_i \in A_j \right) \Rightarrow \left(\overset{\infty}{\exists}\, j.\, crit_i \in A_j \right) \quad \text{for each } i \in \{1, 2\}$$

there exist infinitely many:

$$\left(\overset{\infty}{\exists}\, j.\, wait_i \in A_j \right) \equiv (\forall k \geq 0.\, \exists j > k.\, wait_i \in A_j)$$

Does the semaphore-based algorithm satisfy $P_{nostarve}$?

Does the semaphore-based algorithm satisfy $P_{nostarve}$?



No. Trace $\varnothing \; (\{ wait_2 \} \{ wait_1, wait_2 \} \{ crit_1, wait_2 \})^{\omega} \in Traces(TS)$, but $\notin P_{nostarve}$

Trace equivalence and LT properties

Let TS and TS' be transition systems (over AP) without terminal states:

$Traces(TS) \subseteq Traces(TS')$

if and only if

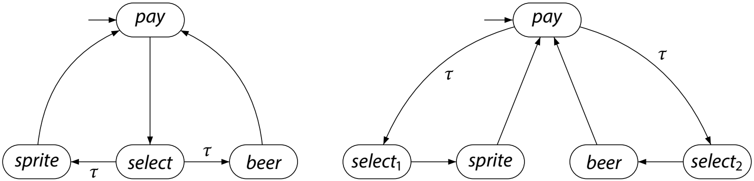
for any LT property P: $TS' \models P$ implies $TS \models P$

$Traces(TS) = Traces(TS')$

if and only if

TS and TS' satisfy the same LT properties

Two beverage vending machines



$$AP = \{ \, pay, sprite, beer \, \}$$

there is no LT-property that can distinguish between these machines

Invariants

- Safety properties ≈ "nothing bad should happen" [Lamport 1977]
- Typical safety property: mutual exclusion property
  - the bad thing (having > 1 process in the critical section) never occurs
- Another typical safety property is deadlock freedom

⇒ These properties are in fact invariants

- An invariant is an LT property
  - that is given by a condition Φ for the states
  - and requires that Φ holds for all reachable states
  - e.g., for mutex property $\Phi \equiv \neg crit_1 \ \vee \ \neg crit_2$

Invariants

- An LT property P_{inv} over AP is an invariant if there is a propositional logic formula Φ over AP such that:

$$P_{inv} = \left\{ A_0 A_1 A_2 \ldots \in \left(2^{AP}\right)^\omega \;\middle|\; \forall j \geq 0.\, A_j \models \Phi \right\}$$

  - Φ is called an invariant condition of P_{inv}
- Note that

$$\begin{array}{lll} TS \models P_{inv} & \text{iff} & \mathit{trace}(\pi) \in P_{inv} \text{ for all paths } \pi \text{ in } TS \\ & \text{iff} & L(s) \models \Phi \text{ for all states } s \text{ that belong to a path of } TS \\ & \text{iff} & L(s) \models \Phi \text{ for all states } s \in \mathit{Reach}(TS) \end{array}$$

- Φ has to be fulfilled by all initial states and
  - satisfaction of Φ is invariant under all transitions in the reachable fragment of TS

Checking an invariant

- Checking an invariant for the propositional formula Φ
  - $=$ check the validity of Φ in every reachable state
  - $\Rightarrow$ use a slight modification of standard graph traversal algorithms (DFS and BFS)
  - provided the given transition system *TS* is finite
- Perform a forward depth-first search
  - at least one state *s* is found with $s \not\models \Phi \Rightarrow$ the invariance of Φ is violated
- Alternative: backward search
  - starts with all states where Φ does not hold
  - calculates (by a DFS or BFS) the set $\bigcup_{s \in S, s \not\models \Phi} Pre^*(s)$
- The time complexity for invariant checking is $\mathcal{O}(N * (1 + |\Phi|) + M)$
  - where *N* denotes the number of reachable states, and
  - $M = \sum_{s \in S} |Post(s)|$ the number of transitions in the reachable fragment of *TS*

Safety properties

- Safety properties may impose requirements on finite path fragments
  - and cannot be verified by considering the reachable states only
- A safety property which is not an invariant:
  - consider a cash dispenser, also known as automated teller machine (ATM)
  - property “money can only be withdrawn once a correct PIN has been provided”

  $\Rightarrow$ not an invariant, since it is not a state property
- But a safety property:
  - any infinite run violating the property has a finite prefix that is “bad”
  - i.e., in which money is withdrawn without issuing a PIN before

Safety properties

- LT property P_{safe} over AP is a safety property if
  - for all $\sigma \in \left(2^{AP}\right)^{\omega} \setminus P_{safe}$ there exists a finite prefix $\widehat{\sigma}$ of σ such that:
    $$P_{safe} \cap \underbrace{\left\{\sigma' \in \left(2^{AP}\right)^{\omega} \mid \widehat{\sigma} \text{ is a prefix of } \sigma'\right\}}_{\text{all possible extensions of } \widehat{\sigma}} = \varnothing$$
  - any such finite word $\widehat{\sigma}$ is called a bad prefix for P_{safe}
- Minimal bad prefix for P_{safe}:
  - is a bad prefix $\widehat{\sigma}$ for P_{safe} for which no proper prefix of $\widehat{\sigma}$ is a bad prefix for P_{safe}

  $\Rightarrow$ minimal bad prefixes are bad prefixes of minimal length

Safety properties and finite traces

For transition system TS without terminal states

and safety property P_{safe}:

$$TS \models P_{safe} \text{ if and only if } Traces_{fin}(TS) \cap BadPref(P_{safe}) = \varnothing$$

where $BadPref(P_{safe})$ is the set of bad prefixes of P_{safe}

Finite trace equivalence and safety properties

For TS and TS' be transition systems (over AP) without terminal states:

$$Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$$

if and only if

for any safety property P_{safe} : $TS' \models P_{safe} \Rightarrow TS \models P_{safe}$

$$Traces_{fin}(TS) = Traces_{fin}(TS')$$

if and only if

TS and TS' satisfy the same safety properties

Why liveness?

- Safety properties specify that “something bad never happens”
- Doing nothing easily fulfills a safety property
  - as this will never lead to a “bad” situation

$\Rightarrow$ Safety properties are complemented by liveness properties

  - that require some progress
- Liveness properties assert that:
  - ”something good” will happen eventually [Lamport 1977]

Liveness properties

LT property P_{live} over AP is a liveness property whenever

$$pref(P_{live}) = \left(2^{AP}\right)^*$$

- A liveness property is an LT property
  - that does not rule out any prefix
- Liveness properties are violated in "infinite time"
  - whereas safety properties are violated in finite time
  - finite traces are of no use to decide whether P holds or not
  - any finite prefix can be extended such that the resulting infinite trace satisfies P

Example liveness properties

- “If the tank is empty, the outlet valve will eventually be closed”
- “If the outlet valve is open and the request signal disappears, the outlet valve will eventually be closed”
- “If the tank is full and a request is present, the outlet valve will eventually be opened”
- “The program terminates within 31 computational steps”
  - ⇒ a finite trace may violate this; this is a safety property!
- “The program eventually terminates”

Liveness properties for mutual exclusion

- Eventually:
  - each process will eventually enter its critical section
- Repeated eventually:
  - each process will enter ist critical section infinitely often
- Starvation freedom:
  - each waiting process will eventually enter its critical section

how to formalize these properties?

Liveness properties for mutual exclusion

$P = \{ A_0 A_1 A_2 \ldots \mid A_j \subseteq AP \ \& \ \ldots \}$ and $AP = \{wait_1, crit_1, wait_2, crit_2\}$

- Eventually:

$$(\exists j \geq 0.\ crit_1 \in A_j) \ \wedge \ (\exists j \geq 0.\ crit_2 \in A_j)$$

- Repeated eventually:

$$\left(\overset{\infty}{\exists}\ j \geq 0.\ crit_1 \in A_j\right) \ \wedge \ \left(\overset{\infty}{\exists}\ j \geq 0.\ crit_2 \in A_j\right)$$

- Starvation freedom:

$$\forall j \geq 0.\ (wait_1 \in A_j \ \Rightarrow \ (\exists k > j.\ crit_1 \in A_k)) \ \wedge$$
$$\forall j \geq 0.\ (wait_2 \in A_j \ \Rightarrow \ (\exists k > j.\ crit_2 \in A_k))$$

Safety vs. liveness

- Are safety and liveness properties disjoint? Only $\left(2^{AP}\right)^{\omega}$ is both.
- Is every linear-time property a safety or liveness property? No.

"the machine provides infinitely often beer
after initially providing sprite three times in a row"

- This property consists of two parts:
  - it requires beer to be provided infinitely often

  $\Rightarrow$ as any finite trace fulfills this, it is a liveness property
  - the first three drinks it provides should all be sprite

  $\Rightarrow$ bad prefix = one of first three drinks is beer; this is a safety property
- Property is thus a conjunction of a safety and a liveness property

does this apply to all properties?

Decomposition theorem

For any LT property P over AP there exists

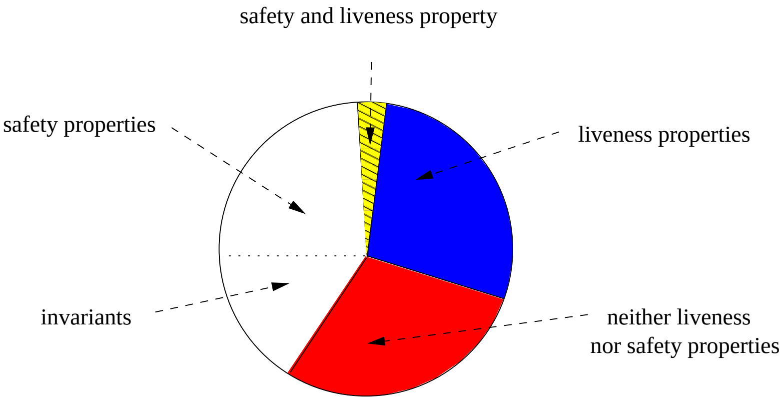
a safety property P_{safe} and a liveness property P_{live}

(both over AP) such that:

$$P = P_{safe} \cap P_{live}$$

$\Rightarrow$ safety and liveness provide an essential characterization of LT properties

Classification of LT 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