Model Checking

Lecture #3: Linear-Time Properties

[Baier & Katoen, Chapter 3]

Joost-Pieter Katoen and Tim Quatmann

& Jip Spel

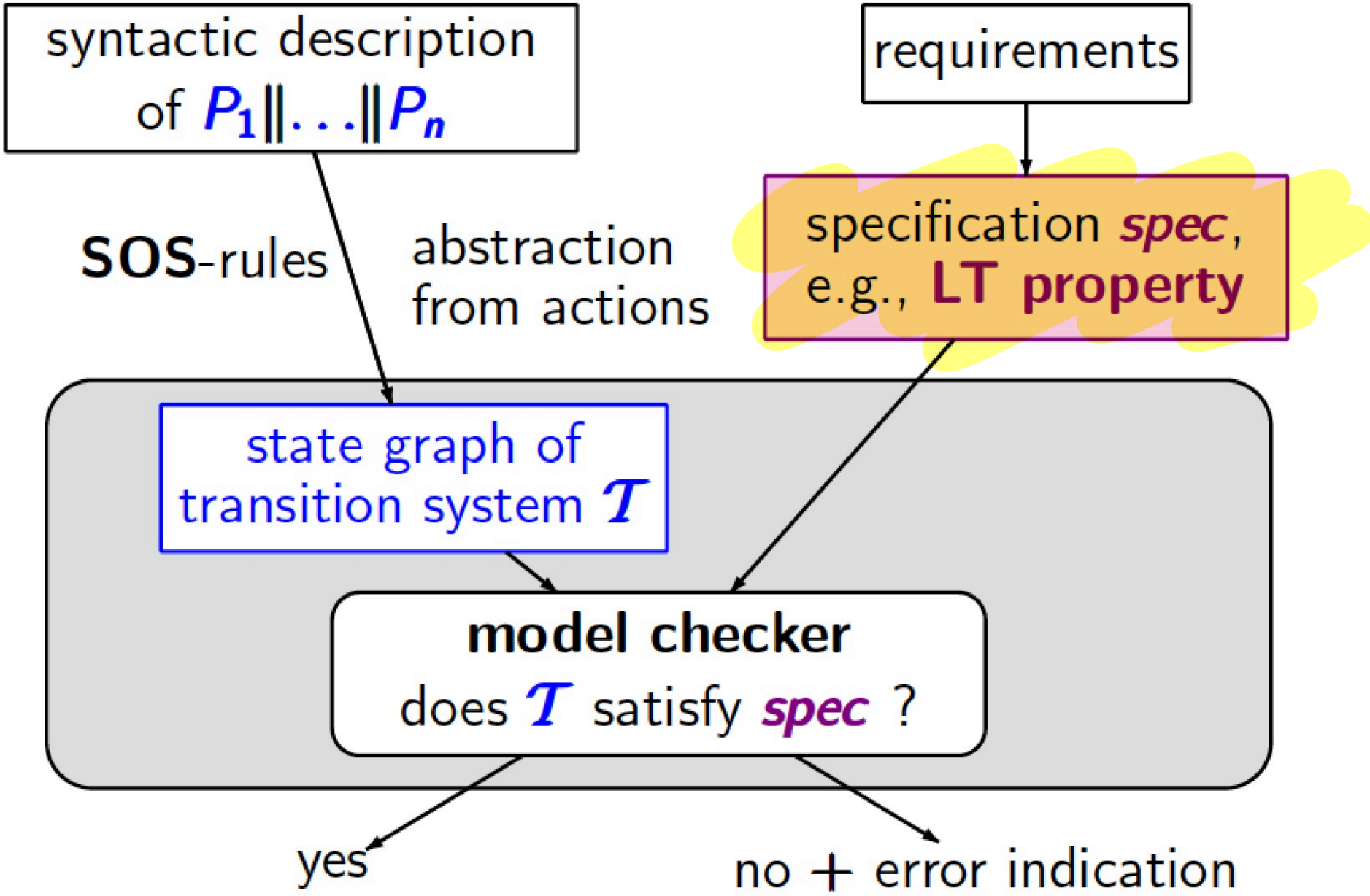
Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, SoSe 2022

Overview

"nothing bad happens"

"eventually something good will happen"


syntactic description
of $P_1 \| \ldots \| P_n$
requirements
SOS-rules
abstraction
from actions
specification *spec*,
e.g., LT property
state graph of
transition system T
model checker
does T satisfy *spec* ?
yes
no + error indication

Overview

Traces

- Actions are mainly used to model the (possibility of) interaction
  synchronous or asynchronous communication
- Here, focus on the states that are visited during executions
  the states themselves are not "observable", but just their atomic propositions
- Traces are sequences of the form $L(s_0)\,L(s_1)\,L(s_2)\ldots$
  record the (sets of) atomic propositions along an execution
- For transition systems without terminal states[1]:
  traces are infinite words over the alphabet 2^{AP}, i.e., they are in $\left(2^{AP}\right)^{\omega}$

[1] This is an assumption commonly used throughout this lecture.

Traces

Definition: Traces

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be transition system without terminal states.

- The trace of execution

$$\rho = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$$

is the infinite word $trace(\rho) = L(s_0)\,L(s_1)\,L(s_2)\ldots$ over $\left(2^{AP}\right)^{\omega}$.
Prefixes of traces are finite traces.

$\rho \in (S \times Act)^{\omega}$

- initial : $s_0 \in I$
- maximal : infinite

Traces

Definition: Traces

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be transition system without terminal states.

- The trace of execution

$$\rho \;=\; s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$$

  is the infinite word $trace(\rho) \;=\; L(s_0)\,L(s_1)\,L(s_2)\ldots$ over $\left(2^{AP}\right)^{\omega}$.
  Prefixes of traces are finite traces.

- The traces of a set Π of executions (or paths) is defined by:

$$trace(\Pi) \;=\; \{\, trace(\pi) \mid \pi \in \Pi \,\}.$$

Traces

Definition: Traces

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be transition system without terminal states.

- The trace of execution

$$\rho \;=\; s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$$

  is the infinite word $trace(\rho) = L(s_0)\,L(s_1)\,L(s_2)\ldots$ over $\left(2^{AP}\right)^{\omega}$.
  Prefixes of traces are finite traces.

- The traces of a set Π of executions (or paths) is defined by:

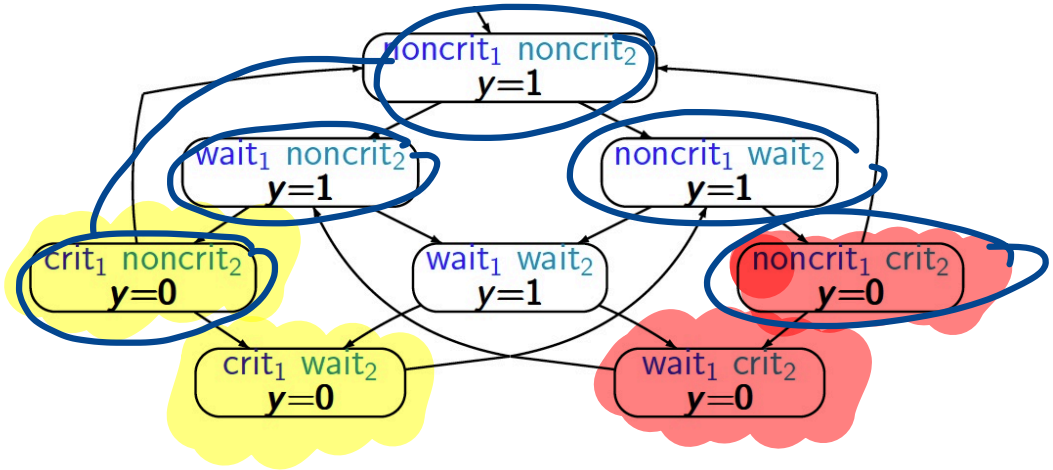
$$trace(\Pi) \;=\; \{\, trace(\pi) \mid \pi \in \Pi \,\}.$$

- The traces of state s are $Traces(s) = trace(Paths(s))$.
- The traces of transition system TS: $Traces(TS) = \bigcup_{s \in I} Traces(s)$.

Example

Consider the mutex transition system. Let $AP = \{ crit_1, crit_2 \}$.
The trace of the path:

$$\pi = \underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle w_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle c_1, n_2, y = 0 \rangle}_{L=\{ crit_1 \}} \to$$
$$\underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, w_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, c_2, y = 0 \rangle}_{L=\{ crit_2 \}} \to \ldots$$



Example

Consider the mutex transition system. Let $AP = \{ crit_1, crit_2 \}$.
The trace of the path:

$$\begin{aligned} \pi \quad = \quad & \underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle w_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle c_1, n_2, y = 0 \rangle}_{L=\{ crit_1 \}} \to \\ & \underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, w_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, c_2, y = 0 \rangle}_{L=\{ crit_2 \}} \to \ldots \end{aligned}$$

is:

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

Example

Consider the mutex transition system. Let $AP = \{ crit_1, crit_2 \}$.
The trace of the path:

$$\begin{aligned}\pi \quad = \quad & \underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle w_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle c_1, n_2, y = 0 \rangle}_{L=\{ crit_1 \}} \to \\ & \underbrace{\langle n_1, n_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, w_2, y = 1 \rangle}_{L=\varnothing} \to \underbrace{\langle n_1, c_2, y = 0 \rangle}_{L=\{ crit_2 \}} \to \ldots\end{aligned}$$

is:

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

Or expressed using ω-regular expressions:

$$trace(\pi) \;=\; \big(\varnothing\,\varnothing\,\{ crit_1 \}\,\varnothing\,\varnothing\,\{ crit_2 \}\big)^{\omega}$$

Regular Expressions

- Let Σ be an alphabet, i.e. countable set of symbols, with $A \in \Sigma$

Regular Expressions

- Let Σ be an alphabet, i.e. countable set of symbols, with $A \in \Sigma$
- Regular expressions over Σ have syntax:

$$E ::= \underline{\varnothing} \mid \underline{\varepsilon} \mid \underline{A} \mid E + E' \mid E.E' \mid E^*$$

empty word

empty language

Regular Expressions

- Let Σ be an alphabet, i.e. countable set of symbols, with $A \in \Sigma$
- Regular expressions over Σ have syntax:

$$E ::= \underline{\varnothing} \mid \underline{\varepsilon} \mid \underline{A} \mid E + E' \mid E.E' \mid E^*$$

- The semantics of regular expression E is a language $\mathfrak{L}(E) \subseteq \Sigma^*$:

$$\mathfrak{L}(\underline{\varnothing}) = \varnothing, \quad \mathfrak{L}(\underline{\varepsilon}) = \{ \varepsilon \}, \quad \mathfrak{L}(\underline{A}) = \{ A \}$$

$$\mathfrak{L}(E+E') = \mathfrak{L}(E) \cup \mathfrak{L}(E') \quad \mathfrak{L}(E.E') = \mathfrak{L}(E).\mathfrak{L}(E') \quad \mathfrak{L}(E^*) = \mathfrak{L}(E)^*$$

Regular Expressions

- Let Σ be an alphabet, i.e. countable set of symbols, with $A \in \Sigma$
- Regular expressions over Σ have syntax:

$$E ::= \underline{\varnothing} \mid \underline{\varepsilon} \mid \underline{A} \mid E + E' \mid E.E' \mid E^*$$

- The semantics of regular expression E is a language $\mathfrak{L}(E) \subseteq \Sigma^*$:

$$\mathfrak{L}(\underline{\varnothing}) = \varnothing, \quad \mathfrak{L}(\underline{\varepsilon}) = \{ \varepsilon \}, \quad \mathfrak{L}(\underline{A}) = \{ A \}$$

$$\mathfrak{L}(E+E') = \mathfrak{L}(E) \cup \mathfrak{L}(E') \quad \mathfrak{L}(E.E') = \mathfrak{L}(E).\mathfrak{L}(E') \quad \mathfrak{L}(E^*) = \mathfrak{L}(E)^*$$

- Regular expressions denote languages of finite words

ω-Regular Expressions: Syntax

Definition: ω-regular expression

An ω-regular expression G over the alphabet Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(F_i)$.

ω-Regular Expressions: Syntax

Definition: ω-regular expression

An ω-regular expression G over the alphabet Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(F_i)$.

- ω-Regular expressions denote languages of infinite words

ω-Regular Expressions: Syntax

Definition: ω-regular expression

An ω-regular expression G over the alphabet Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(F_i)$.

- ω-Regular expressions denote languages of infinite words
- Examples over the alphabet $\Sigma = \{ A, B \}$:
  - language of all words with infinitely many As: $(B^*.A)^\omega$

ω-Regular Expressions: Syntax

Definition: ω-regular expression

An ω-regular expression G over the alphabet Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(F_i)$.

- ω-Regular expressions denote languages of infinite words
- Examples over the alphabet $\Sigma = \{ A, B \}$:
  - language of all words with infinitely many As: $(B^*.A)^\omega$
  - language of all words with finitely many As: $(A + B)^*.B^\omega$

ω-Regular Expressions: Syntax

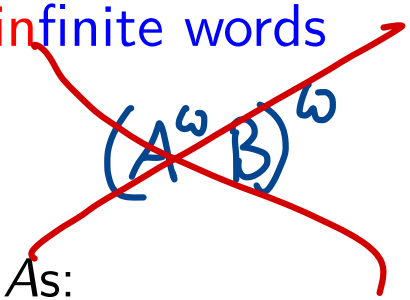
Definition: ω-regular expression

An ω-regular expression G over the alphabet Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(F_i)$.

- ω-Regular expressions denote languages of infinite words



- Examples over the alphabet $\Sigma = \{A, B\}$:
  - language of all words with infinitely many As: $(B^*.A)^\omega$
  - language of all words with finitely many As: $(A + B)^*.B^\omega$
  - the empty language $\varnothing^\omega$

ω-Regular Expressions: Semantics

Definition: semantics of ω-regular expressions

The semantics of ω-regular expression $G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega$ is the language $\mathfrak{L}(G) \subseteq \Sigma^\omega$ defined by:

$$\mathfrak{L}_\omega(G) = \mathfrak{L}(E_1).\mathfrak{L}(F_1)^\omega \cup \ldots \cup \mathfrak{L}(E_n).\mathfrak{L}(F_n)^\omega.$$

where for $\mathfrak{L} \subseteq \Sigma^*$, we have $\mathfrak{L}^\omega = \{ w_1 w_2 w_3 \ldots \mid \forall i \geq 0.\, w_i \in \mathfrak{L} \}$.

ω-Regular Expressions: Semantics

Definition: semantics of ω-regular expressions

The semantics of ω-regular expression $G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega$ is the language $\mathfrak{L}(G) \subseteq \Sigma^\omega$ defined by:

$$\mathfrak{L}_\omega(G) = \mathfrak{L}(E_1).\mathfrak{L}(F_1)^\omega \cup \ldots \cup \mathfrak{L}(E_n).\mathfrak{L}(F_n)^\omega.$$

where for $\mathfrak{L} \subseteq \Sigma^*$, we have $\mathfrak{L}^\omega = \{ w_1 w_2 w_3 \ldots \mid \forall i \geq 0.\, w_i \in \mathfrak{L} \}$.

The ω-regular expression G_1 and G_2 are equivalent,

denoted $G_1 \equiv G_2$, if $\mathfrak{L}_\omega(G_1) = \mathfrak{L}_\omega(G_2)$.

Overview

Linear-Time Properties

Definition: Linear-Time Property

A linear-time property (LT property) over AP is a subset of $\left(2^{AP}\right)^{\omega}$.

language of infinite words over 2^{AP}

Linear-Time Properties

Definition: Linear-Time Property

A linear-time property (LT property) over AP is a subset of $(2^{AP})^{\omega}$.

- Linear-time properties specify desirable traces of a transition system
- They are infinite words $A_0\, A_1\, A_2 \ldots$ with $A_i \subseteq AP$, i.e. traces
- No finite words, as TS is assumed to have no terminal states
- TS satisfies property P if all its "observable" behaviours are admitted by P

$$TS \models P \quad \text{iff} \quad Traces(TS) \subseteq P$$

Linear-Time Properties

Definition: Linear-Time Property

A linear-time property (LT property) over AP is a subset of $(2^{AP})^{\omega}$.

- Linear-time properties specify desirable traces of a transition system
- They are infinite words $A_0\, A_1\, A_2 \ldots$ with $A_i \subseteq AP$, i.e. traces
- No finite words, as TS is assumed to have no terminal states
- TS satisfies property P if all its "observable" behaviours are admitted by P

Satisfaction relation for LT properties

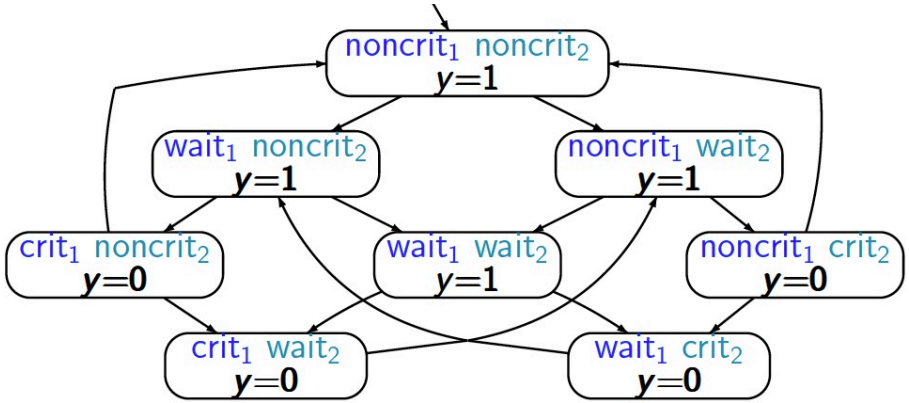
Transition system TS (over AP) satisfies LT property P (over AP):

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

Mutual Exclusion as LT Property

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

$AP = \{crit_1, crit_2\}$



Mutual Exclusion as LT Property

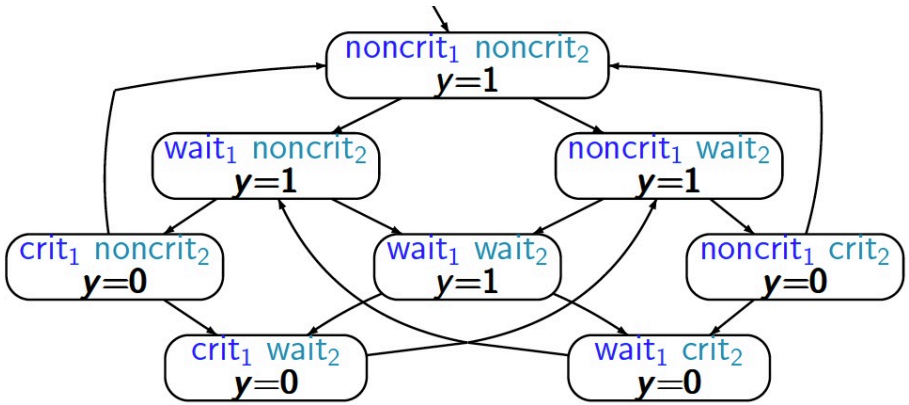
"Always at most one thread is in its critical section"

- Let $AP = \{ crit_1, crit_2 \}$

  other atomic propositions are not of any relevance for this property

→ Formalize:

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



Mutual Exclusion as LT Property

"Always at most one thread 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

$$P_{mutex} = \text{set of infinite words } A_0 A_1 A_2 \ldots$$

$$\text{with } \{ crit_1, crit_2 \} \not\subseteq A_i \text{ for all } 0 \leq i$$

Mutual Exclusion as LT Property

"Always at most one thread 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} = set of infinite words $A_0 A_1 A_2 \ldots$

  with $\{ crit_1, crit_2 \} \not\subseteq A_i$ for all $0 \leq i$

- Contained in P_{mutex} are e.g., the infinite words:
  - $(\{ crit_1 \}\{ crit_2 \})^\omega$ and $(\{ crit_1 \})^\omega$ and $\varnothing^\omega$

not same set

Mutual Exclusion as LT Property

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

- Let $AP = \{ crit_1, crit_2 \}$

  other atomic propositions are not of any relevance for this property

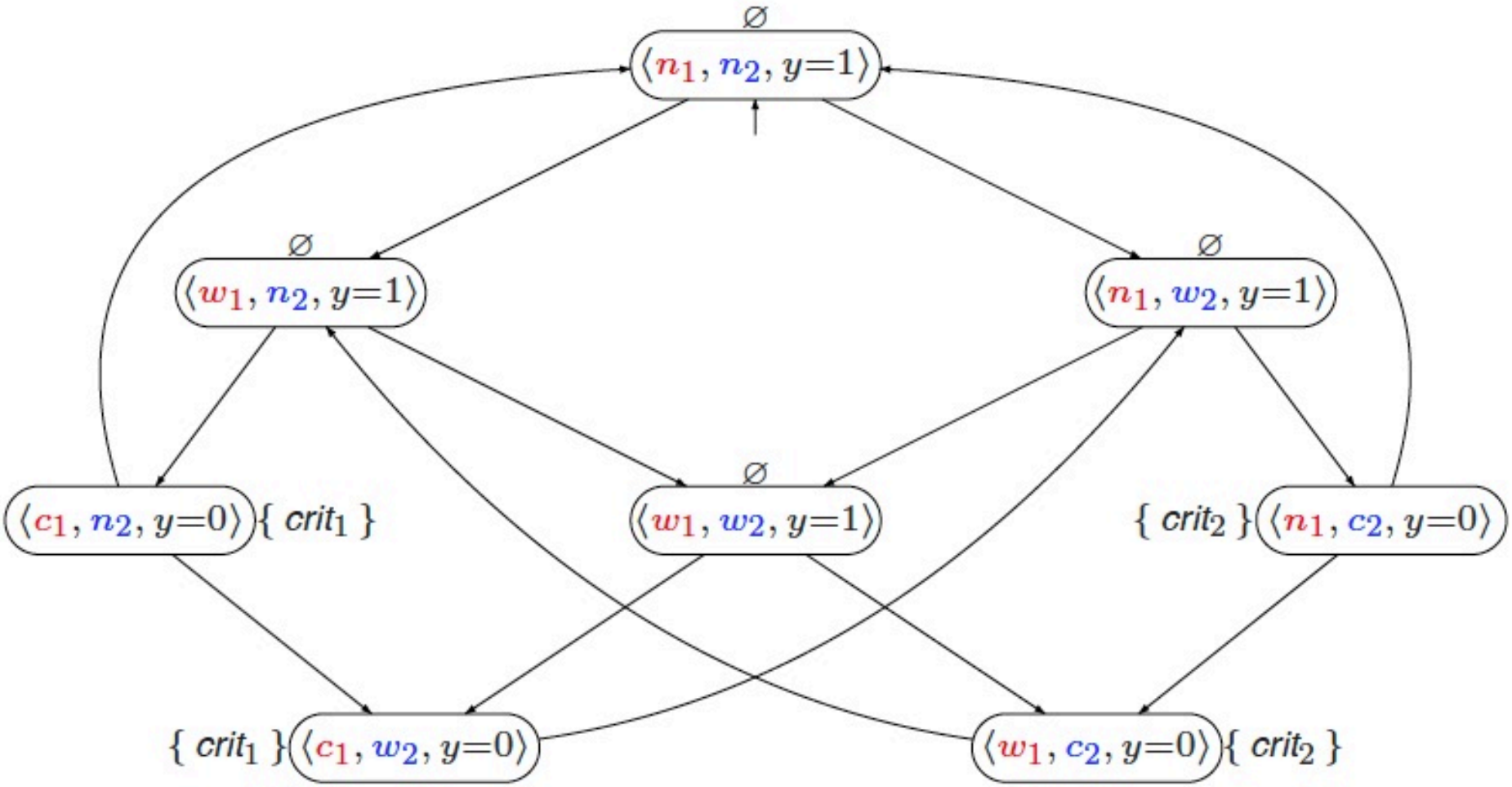
- Formalization as LT property

  P_{mutex} = set of infinite words $A_0 A_1 A_2 \ldots$

  with $\{ crit_1, crit_2 \} \not\subseteq A_i$ for all $0 \leq i$

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

Mutual Exclusion by Semaphores



{crit1, crit2} occur at a state? no

Yes, the semaphore-based algorithm satisfies P_{mutex}.

Starvation Freedom as LT Property

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

$AP = \{wait_1, wait_2, crit_1, crit_2\}$

$P_{nostarve}$ = set of infinite words $A_0 A_1 A_2 \ldots$ s.t.

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

see $wait_i$ infinitely often

$$\forall k \geq 0.\ \exists j \geq k$$

Starvation Freedom as LT Property

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

- Let $AP = \{ wait_1, crit_1, wait_2, crit_2 \}$

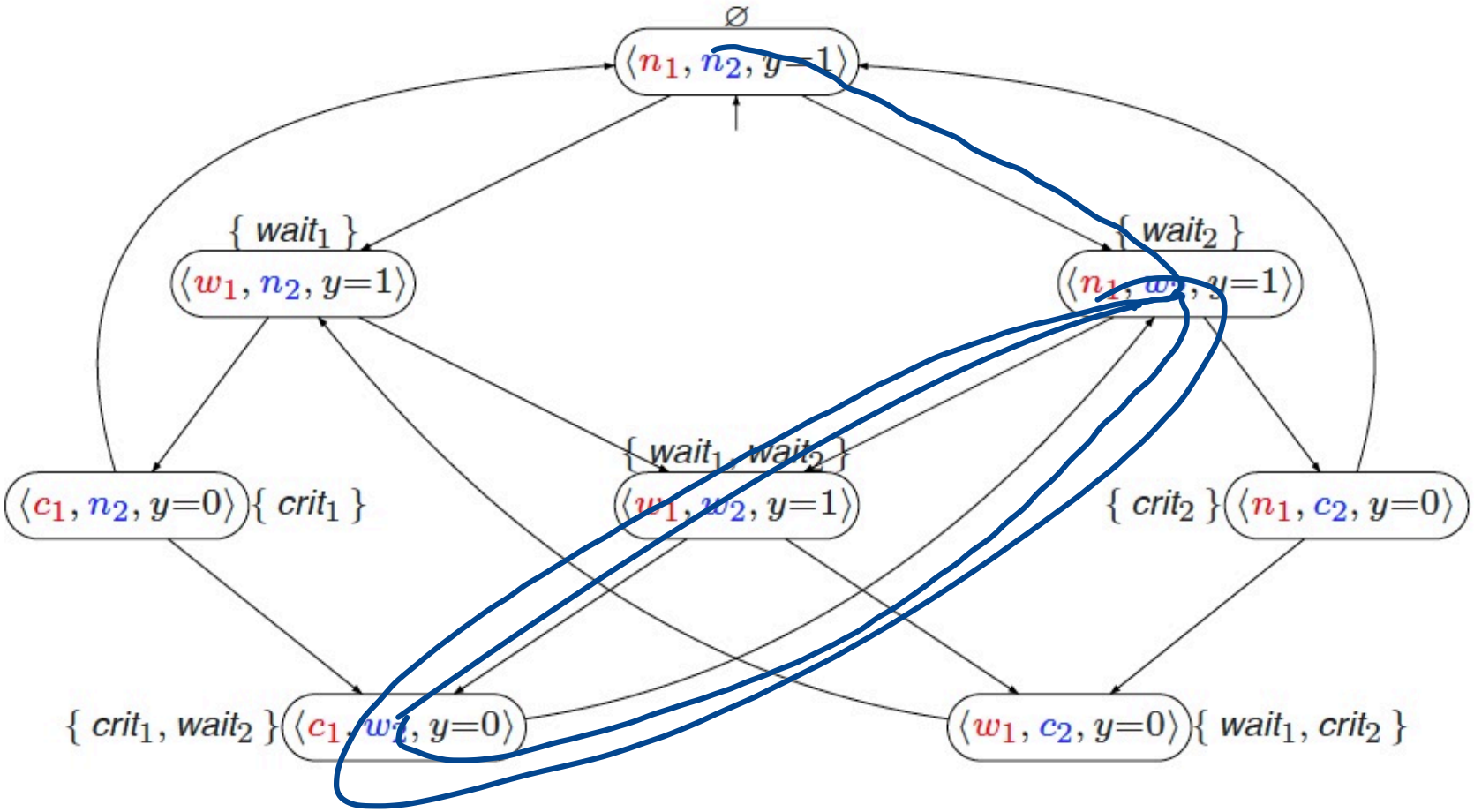
- Formalization as LT-property

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

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

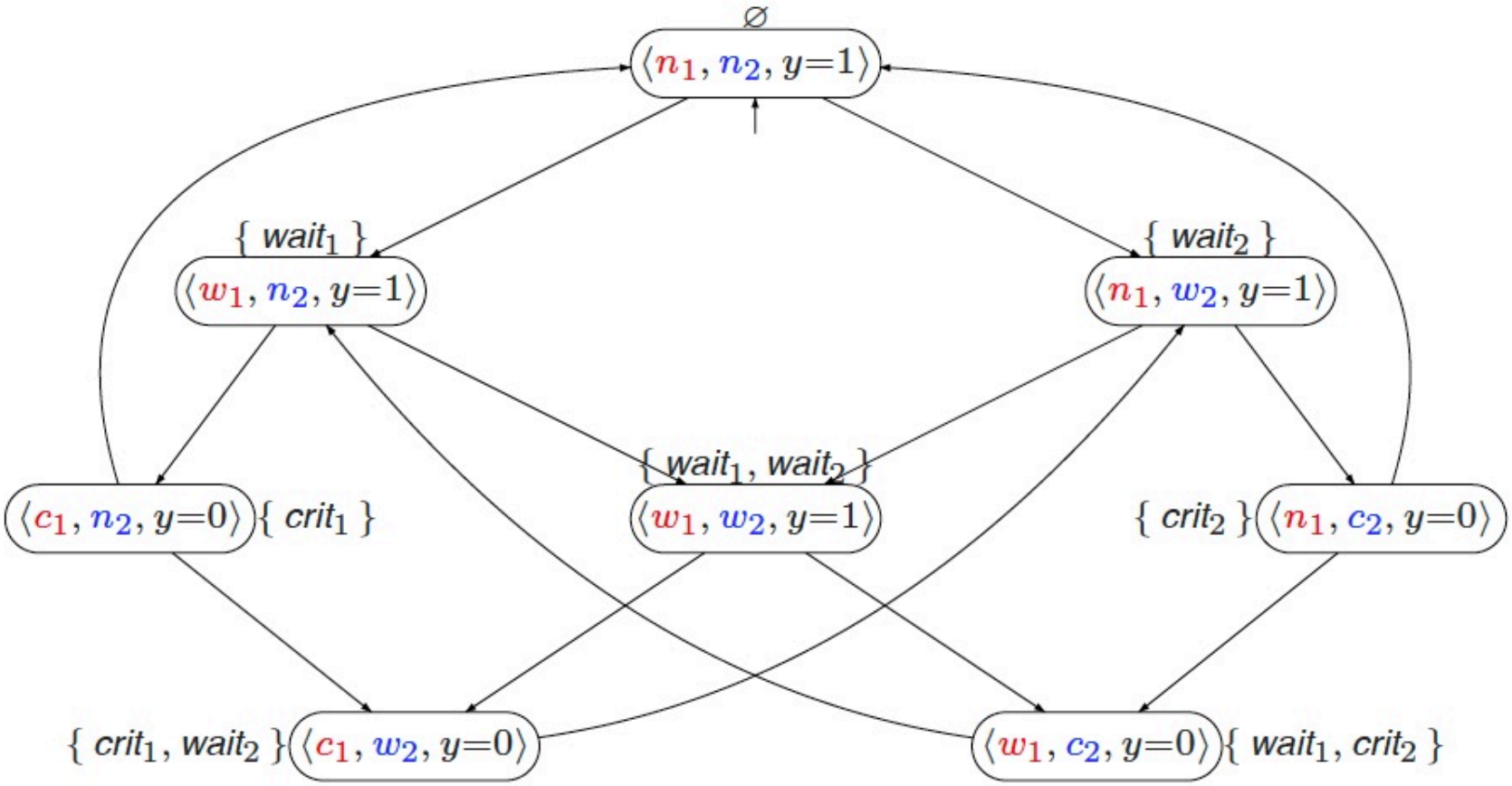
where: $\left(\overset{\infty}{\exists} j.\ wait_i \in A_j\right)$ abbreviates $(\forall k \geq 0.\ \exists j > k.\ wait_i \in A_j)$

Starvation Freedom by Semaphores



Can we see e.g. $wait_2$ inf. often and see $crit_2$ fin. often?

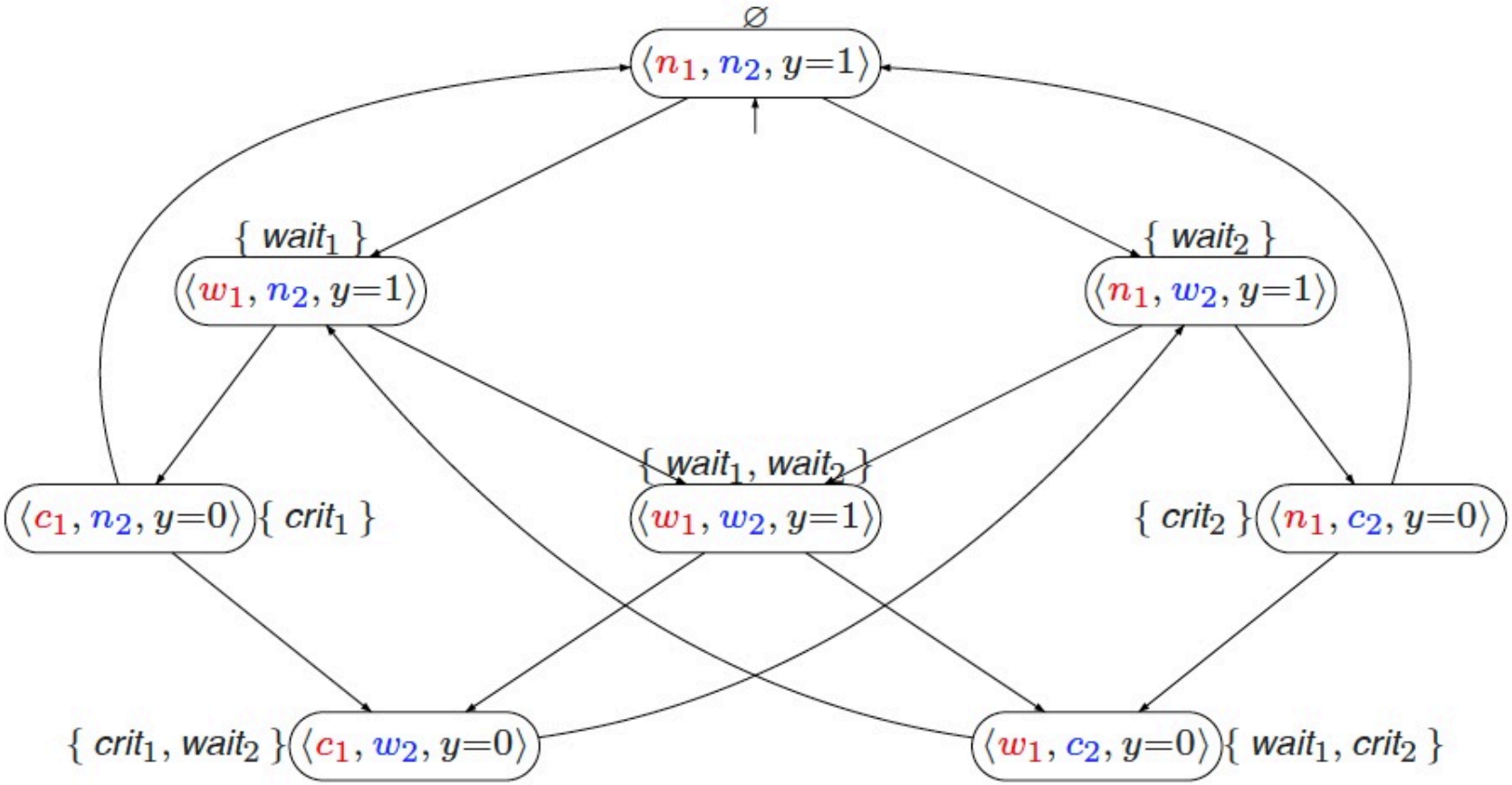
Starvation Freedom by Semaphores



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

TS ⊨ P iff Traces(TS) ⊆ P no

Starvation Freedom by Semaphores



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 Inclusion and LT Properties

For 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 $TS \models P$.

Trace Inclusion and LT Properties

For 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 $TS \models P$.

"$\Rightarrow$": Let $Traces(TS) \subseteq Traces(TS')$ ① and P s.t. $TS' \models P$

$Traces(TS') \subseteq P$ ②

Show $TS \models P$ so show $Traces(TS) \subseteq P$

$Traces(TS) \subseteq Traces(TS') \subseteq P$ ↗ ①

Trace Inclusion and LT Properties

For 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 $TS \models P$.

"$\Leftarrow$": Assume for all LT property P: $TS' \models P \Rightarrow^{*} TS \models P$

Assume $\nexists$ $\sigma \in Traces(TS)$. $\sigma \notin Traces(TS')$

$P = Traces(TS')$, so $\sigma \notin P$ but then $\sigma \in Traces(TS)$

so $Traces(TS) \not\subseteq P$ $\lightning^{*}$

so our assumption $\nexists$ does not hold so

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

Trace Inclusion and LT Properties

For 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 $TS \models P$.

Corollary

$Traces(TS) = Traces(TS')$ iff TS and TS' satisfy the same LT properties.

Overview

Invariants

- LT property E_{inv} over AP is an invariant if it has the form:

$$E_{inv} = \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \forall j \geq 0.\ A_j \vDash \Phi \}$$

where (invariant condition) Φ is a propositional logic formula over AP

$\varphi := a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$ for $a \in AP$

e.g. $AP = \{a, b\}$ $\varphi = a \vee b$

$A_0 \vDash \varphi$ iff $A_0 \in \{\{a\}, \{b\}, \{a, b\}\}$

Invariants

- LT property E_{inv} over AP is an invariant if it has the form:

$$E_{inv} = \left\{ A_0 A_1 A_2 \ldots \in \left(2^{AP}\right)^{\omega} \mid \forall j \geq 0.\ A_j \vDash \Phi \right\}$$

where (invariant condition) Φ is a propositional logic formula over AP

- Note that

$$\begin{aligned} TS \vDash E_{inv} \quad & \text{iff} \quad trace(\pi) \in E_{inv} \text{ for all paths } \pi \text{ in } TS \\ & \text{iff} \quad L(s) \vDash \Phi \text{ for all states } s \text{ that belong to a path of } TS \\ & \text{iff} \quad L(s) \vDash \Phi \text{ for all states } s \in Reach(TS) \end{aligned}$$

Invariants

- LT property E_{inv} over AP is an invariant if it has the form:

$$E_{inv} = \left\{ A_0 A_1 A_2 \ldots \in \left(2^{AP}\right)^{\omega} \mid \forall j \geq 0.\ A_j \vDash \Phi \right\}$$

  where (invariant condition) Φ is a propositional logic formula over AP

- Note that

$$\begin{aligned} TS \vDash E_{inv} \quad & \text{iff} \quad trace(\pi) \in E_{inv} \text{ for all paths } \pi \text{ in } TS \\ & \text{iff} \quad L(s) \vDash \Phi \text{ for all states } s \text{ that belong to a path of } TS \\ & \text{iff} \quad L(s) \vDash \Phi \text{ for all states } s \in Reach(TS) \end{aligned}$$

- all initial states fulfil Φ and all transitions in the reachable fragment of TS preserve Φ

Example Invariants

mutual exclusion (safety):

$$MUTEX = \text{set of all infinite words } A_0 A_1 A_2 \ldots \text{ s.t. } \forall i \in \mathbb{N}.\ \mathbf{crit}_1 \notin A_i \text{ or } \mathbf{crit}_2 \notin A_i$$

invariant condition: $\Phi = \neg\mathbf{crit}_1 \vee \neg\mathbf{crit}_2$

deadlock freedom for 5 dining philosophers:

$$DF = \text{set of all infinite words } A_0 A_1 A_2 \ldots \text{ s.t. } \forall i \in \mathbb{N}\ \exists j \in \{0,1,2,3,4\}.\ \mathbf{wait}_j \notin A_i$$

invariant condition:

$$\Phi = \neg\mathbf{wait}_0 \vee \neg\mathbf{wait}_1 \vee \neg\mathbf{wait}_2 \vee \neg\mathbf{wait}_3 \vee \neg\mathbf{wait}_4$$

here: $AP = \{\mathbf{wait}_j : 0 \leq j \leq 4\} \cup \{\ldots\}$

Safety Properties

- Safety properties may impose requirements on finite path fragments
  - and cannot be verified by considering the reachable states individually

Safety Properties

- Safety properties may impose requirements on finite path fragments
  - and cannot be verified by considering the reachable states individually
- Every invariant is a safety property, but not the reverse

Safety Properties

- Safety properties may impose requirements on finite path fragments
  - and cannot be verified by considering the reachable states individually
- Every invariant is a safety property, but not the reverse
- A safety property which is not an invariant:
  - consider a cash dispenser, aka: 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

Safety Properties

- Safety properties may impose requirements on finite path fragments
  - and cannot be verified by considering the reachable states individually
- Every invariant is a safety property, but not the reverse
- A safety property which is not an invariant:
  - consider a cash dispenser, aka: 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

bad prefix ✗

Safety Properties

Definition: Safety Property

LT property E_{safe} over AP is a safety property if for all $\sigma \in \left(2^{AP}\right)^{\omega} \setminus E_{safe}$:

$$E_{safe} \cap \left\{\sigma' \in \left(2^{AP}\right)^{\omega} \mid \hat{\sigma} \text{ is a prefix of } \sigma'\right\} = \varnothing.$$

for some prefix $\hat{\sigma}$ of σ.

Intersection between all traces in E_{safe} and any infinite trace that has a bad prefix is empty

Safety Properties

Definition: Safety Property

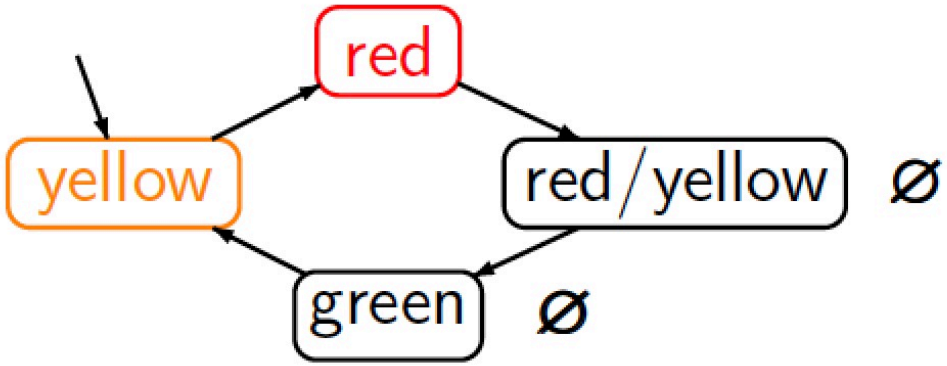
LT property E_{safe} over AP is a safety property if for all $\sigma \in (2^{AP})^{\omega} \setminus E_{safe}$:

$$E_{safe} \cap \left\{\sigma' \in (2^{AP})^{\omega} \mid \hat{\sigma} \text{ is a prefix of } \sigma'\right\} = \varnothing.$$

for some prefix $\hat{\sigma}$ of σ.

- Path fragment $\hat{\sigma}$ is called a bad prefix of E_{safe}
- Let $BadPref(E_{safe})$ denote the set of bad prefixes of E_{safe}
- $\hat{\sigma} \in E_{safe}$ is minimal if no proper prefix of it is in $BadPref(E_{safe})$

Examples



(1) "every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$ (4)

E = set of all infinite words $A_0 A_1 A_2 \ldots$ (2)
over 2^{AP} such that for all $i \in \mathbb{N}$:
$\mathit{red} \in A_i \implies i \geq 1$ and $\mathit{yellow} \in A_{i-1}$

is a safety property over $AP = \{\mathit{red}, \mathit{yellow}\}$ with

(3) $\mathit{BadPref}$ = set of all finite words $A_0 A_1 \ldots A_n$
over 2^{AP} s.t. for some $i \in \{0, \ldots, n\}$:
$\mathit{red} \in A_i \wedge (i{=}0 \vee \mathit{yellow} \notin A_{i-1})$

Safety Properties and Finite Traces

For transition system TS without terminal states

and safety property E_{safe}:

$TS \models E_{safe}$ if and only if $Traces_{fin}(TS) \cap BadPref(E_{safe}) = \emptyset$.

Finite Prefixes

- For trace $\sigma \in (2^{AP})^{\omega}$, let $pref(\sigma)$ be the set of finite prefixes of σ:

$$pref(\sigma) = \{\, \hat{\sigma} \in (2^{AP})^{*} \mid \hat{\sigma} \text{ is a finite prefix of } \sigma \,\}$$

  - if $\sigma = A_0\, A_1\, \ldots$ then $pref(\sigma) = \{\varepsilon, A_0, A_0 A_1, A_0 A_1 A_2, \ldots\}$

Finite Prefixes

- For trace $\sigma \in \left(2^{AP}\right)^{\omega}$, let $\mathit{pref}(\sigma)$ be the set of finite prefixes of σ:

$$\mathit{pref}(\sigma) = \{\, \hat{\sigma} \in \left(2^{AP}\right)^{*} \mid \hat{\sigma} \text{ is a finite prefix of } \sigma \,\}$$

  - if $\sigma = A_0\, A_1\, \ldots$ then $\mathit{pref}(\sigma) = \{\varepsilon, A_0, A_0 A_1, A_0 A_1 A_2, \ldots\}$

- For property P this is lifted as follows:

$$\mathit{pref}(P) = \bigcup_{\sigma \in P} \mathit{pref}(\sigma)$$

Finite Prefixes

- For trace $\sigma \in \left(2^{AP}\right)^{\omega}$, let $\mathit{pref}(\sigma)$ be the set of finite prefixes of σ:

$$\mathit{pref}(\sigma) = \{\, \hat{\sigma} \in \left(2^{AP}\right)^{*} \mid \hat{\sigma} \text{ is a finite prefix of } \sigma \,\}$$

  - if $\sigma = A_0\, A_1\, \ldots$ then $\mathit{pref}(\sigma) = \{\varepsilon, A_0, A_0 A_1, A_0 A_1 A_2, \ldots\}$

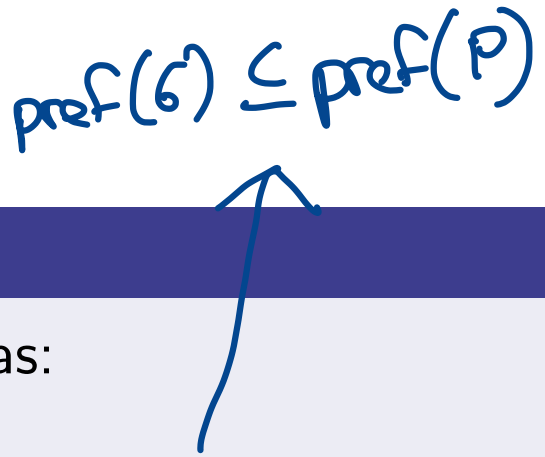
- For property P this is lifted as follows:

$$\mathit{pref}(P) = \bigcup_{\sigma \in P} \mathit{pref}(\sigma)$$

- For transition system TS we have:

$$\mathit{Traces}_{\mathit{fin}}(TS) = \mathit{pref}(\mathit{Traces}(TS))$$

Closure



Definition: closure of a property

The closure of LT property P is defined as:

$$cl(P) = \{\sigma \in (2^{AP})^{\omega} \mid \text{every prefix of } \sigma \text{ is a prefix of } P\}$$

- $cl(P)$ contains the set of infinite traces whose finite prefixes are also prefixes of P, or equivalently
- infinite traces in the closure of P do not have a prefix that is not a prefix of P

Safety Properties and Closure

For any LT property P over AP:

P is a safety property if and only if $cl(P) = P$.

Safety Properties and Clo

Definition: Safety Property

LT property E_{safe} over AP is a safety property if for all $\sigma \in (2^{AP})^\omega \setminus E_{safe}$:

① $$E_{safe} \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a prefix of } \sigma'\} = \varnothing.$$

for some prefix $\hat{\sigma}$ of σ.

② $$cl(P) = \{\sigma \in (2^{AP})^\omega \mid \text{every prefix of } \sigma \text{ is a prefix of } P\}$$

For any LT property P over AP:

P is a safety property if and only if $cl(P) = P$.

pref(6) ⊆ pref(P)

"⇒": Let P be a safety property. Show cl(P)=P

By ② we know P⊆cl(P) so show cl(P)⊆P by contradiction

Assume ∃ 6 ∈ cl(P)\P, 6 = A₀A₁A₂...

As P is a safety property, and 6 ∉ P ①: ∃ $\hat{6}$ ∈ pref(6) which is a bad prefix of P.

6 ∈ cl(P) pref(6) ⊆ pref(P) ②

But then $\hat{6}$ ∈ pref(P) ↯ with P is a safety property

so cl(P) ⊆ P □

Safety Properties and Clo

Definition: Safety Property

LT property E_{safe} over AP is a safety property if for all $\sigma \in (2^{AP})^\omega \setminus E_{safe}$:

(1) $$E_{safe} \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a prefix of } \sigma'\} = \varnothing.$$

for some prefix $\hat{\sigma}$ of σ.

(2) $$cl(P) = \{\sigma \in (2^{AP})^\omega \mid \text{every prefix of } \sigma \text{ is a prefix of } P\}$$

For any LT property P over AP:

P is a safety property if and only if $cl(P) = P$.

"⇐": Let $cl(P) = P$, show P is a safety property (1)

Let $\sigma \in (2^{AP})^\omega \setminus P$ $\quad \sigma \notin P$ so $\sigma \notin cl(P)$

Thus σ has a prefix $\hat{\sigma}$ that is not a prefix in P.

so none of the words $\sigma' \in (2^{AP})^\omega$ with $\hat{\sigma} \in pref(\sigma')$ belong to P

Hence $P \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \in pref(\sigma')\} = \emptyset$

for all $\sigma \in (2^{AP})^\omega \setminus P$ and some $\hat{\sigma} \in pref(\sigma)$

Safety Properties and Finite Trace Equivalence

Let 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 E_{safe} : $TS' \models E_{safe} \Rightarrow TS \models E_{safe}$.

Exercise Sheet 02

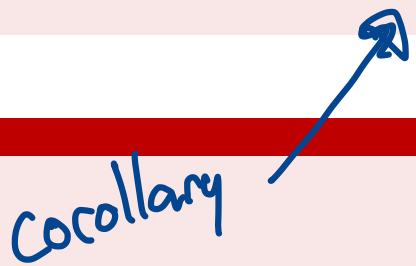
Safety Properties and Finite Trace Equivalence

Let 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 E_{safe} : $TS' \models E_{safe} \Rightarrow TS \models E_{safe}$.



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

if and only if

TS and TS' satisfy the same safety properties.

Finite versus Infinite Traces

For TS without terminal states and finite TS':

$$Traces(TS) \subseteq Traces(TS') \quad \text{iff} \quad Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$$

[2] Transition systems in which each state has finitely many direct successors.

Finite versus Infinite Traces

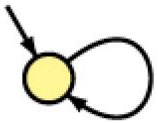
For TS without terminal states and finite TS':

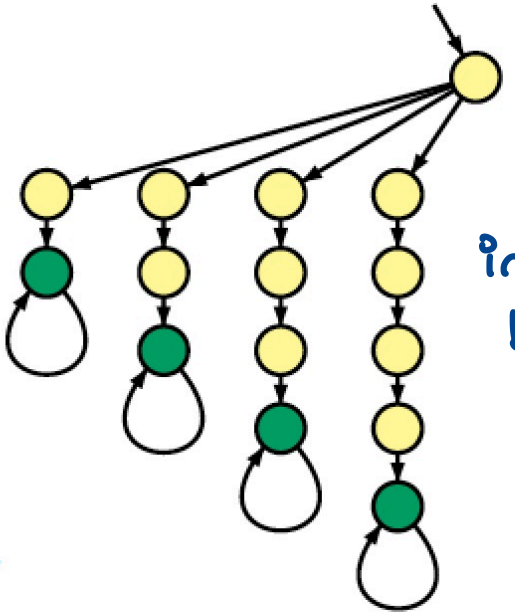
$$Traces(TS) \subseteq Traces(TS') \quad \text{iff} \quad Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$$

this does not hold for infinite TS' (cf. next slide)
but also holds for image-finite TS'.[2]

[2] Transition systems in which each state has finitely many direct successors.

Trace Equivalence ≠ Finite Trace Equivalence

$\mathcal{T}$ 

$\mathcal{T}'$ 

$$
\begin{aligned}
\mathit{Traces}(\mathcal{T}) &= \{\varnothing^\omega\} \\
\mathit{Traces}_{\mathit{fin}}(\mathcal{T}) &= \{\varnothing^n : n \geq 0\} \\
\mathit{Traces}(\mathcal{T}') &= \{\varnothing^n \{b\}^\omega : n \geq 2\} \\
\mathit{Traces}_{\mathit{fin}}(\mathcal{T}') &= \{\varnothing^n : n \geq 0\} \cup \\
&\quad \{\varnothing^n \{b\}^m : n \geq 2 \wedge m \geq 1\}
\end{aligned}
$$

$\mathit{Traces}(\mathcal{T}) \not\subseteq \mathit{Traces}(\mathcal{T}')$, but
$\mathit{Traces}_{\mathit{fin}}(\mathcal{T}) \subseteq \mathit{Traces}_{\mathit{fin}}(\mathcal{T}')$

LT property
$E \mathrel{\widehat{=}}$ "eventually b"

$\mathcal{T} \not\models E$, $\mathcal{T}' \models E$

Overview

Why Liveness?

- Safety properties specify that:

  "something bad never happens" [Lamport 1977]

Why Liveness?

- Safety properties specify that:
  "something bad never happens" [Lamport 1977]
- Doing nothing easily fulfils a safety property
  as this will never lead to a "bad" situation

Why Liveness?

- Safety properties specify that:
  "something bad never happens" [Lamport 1977]

- Doing nothing easily fulfils a safety property
  as this will never lead to a "bad" situation

$\Rightarrow$ Safety properties are complemented by liveness properties
that require some progress

Why Liveness?

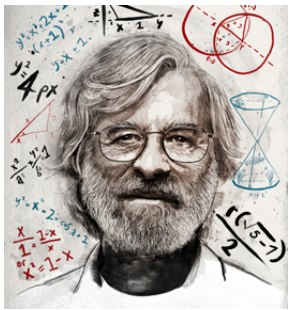
- Safety properties specify that:
  "something bad never happens" [Lamport 1977]

- Doing nothing easily fulfils a safety property
  as this will never lead to a "bad" situation

⇒ Safety properties are complemented by liveness properties
  that require some progress

- Liveness properties assert that:
  "something good" will happen eventually [Lamport 1977]

The Meaning of Liveness



The question of whether a real system satisfies a liveness property is meaningless; it can be answered only by observing the system for an infinite length of time, and real systems don't run forever.

Liveness is always an approximation to the property we really care about. We want a program to terminate within 100 years, but proving that it does would require addition of distracting timing assumptions.

So, we prove the weaker condition that the program eventually terminates. This doesn't prove that the program will terminate within our lifetimes, but it does demonstrate the absence of infinite loops.

[Lamport 2000]

Liveness Properties

Definition: Liveness property

LT property E_{live} over AP is a *liveness* property whenever

$$pref(E_{live}) = (2^{AP})^*.$$

$\forall w \in (2^{AP})^* . \exists \sigma \in (2^{AP})^\omega$ s.t. $w\sigma \in E_{live}$

Liveness Properties

Definition: Liveness property

LT property E_{live} over AP is a *liveness* property whenever

$$pref(E_{live}) = (2^{AP})^*.$$

- A liveness property 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 E_{live} holds or not
  - any finite prefix can be extended such that the resulting infinite trace satisfies E_{live}

Liveness Properties

Definition: Liveness property

LT property E_{live} over AP is a *liveness* property whenever

$$pref(E_{live}) = (2^{AP})^*.$$

- A liveness property 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 E_{live} holds or not
  - any finite prefix can be extended such that the resulting infinite trace satisfies E_{live}
- Equivalently, E_{live} is a liveness property iff $cl(E_{live}) = (2^{AP})^{\omega}$

Example Liveness Properties for Mutual Exclusion

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

Example Liveness Properties for Mutual Exclusion

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

- Any thread eventually is in its critical section:

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

Example Liveness Properties for Mutual Exclusion

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

- Any thread eventually is in its critical section:

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

- Any thread is infinitely often in its critical section:

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

$\forall k \geq 0.\ \exists j \geq k$

Example Liveness Properties for Mutual Exclusion

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

- Any thread eventually is in its critical section:

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

- Any thread is infinitely often in its critical section:

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

- Starvation freedom — no thread is "starving":

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

Overview

Safety versus Liveness

- Are safety and liveness properties disjoint? Yes, almost

Safety versus Liveness

- Are safety and liveness properties disjoint? Yes, almost
- The property $(2^{AP})^{\omega}$ is both a safety and a liveness property

Safety versus Liveness

- Are safety and liveness properties disjoint? Yes, almost
- The property $\left(2^{AP}\right)^{\omega}$ is both a safety and a liveness property
- Is any linear-time property a safety or liveness property? No

Safety versus Liveness

- Are safety and liveness properties disjoint? Yes, almost
- The property $\left(2^{AP}\right)^{\omega}$ is both a safety and a liveness property
- Is any linear-time property a safety or liveness property? No
- But:

  for any LT property P there exists an equivalent LT property P'

  which is a conjunction of a safety and a liveness property

Safety versus Liveness

- Are safety and liveness properties disjoint? Yes, almost
- The property $\left(2^{AP}\right)^{\omega}$ is both a safety and a liveness property
- Is any linear-time property a safety or liveness property? No
- But:

  for any LT property P there exists an equivalent LT property P'

  which is a conjunction of a safety and a liveness property

$\Rightarrow$ safety and liveness provide a complete characterization of LT properties

Alpern-Schneider Characterisation

Bowen Alpern

Fred B. Schneider

Neither Safe nor Live

liveness

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

↳ safety prop → badprefix

Neither Safe nor Live

"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

Neither Safe nor Live

"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

- This property is thus a conjunction of a safety and a liveness property

does this apply to all such properties?

Decomposition Theorem

Decomposition theorem for LT properties

For any LT property P over AP there exists a safety property E_{safe} and a liveness property E_{live} (both over AP) such that:

$$P = E_{safe} \cap E_{live}.$$

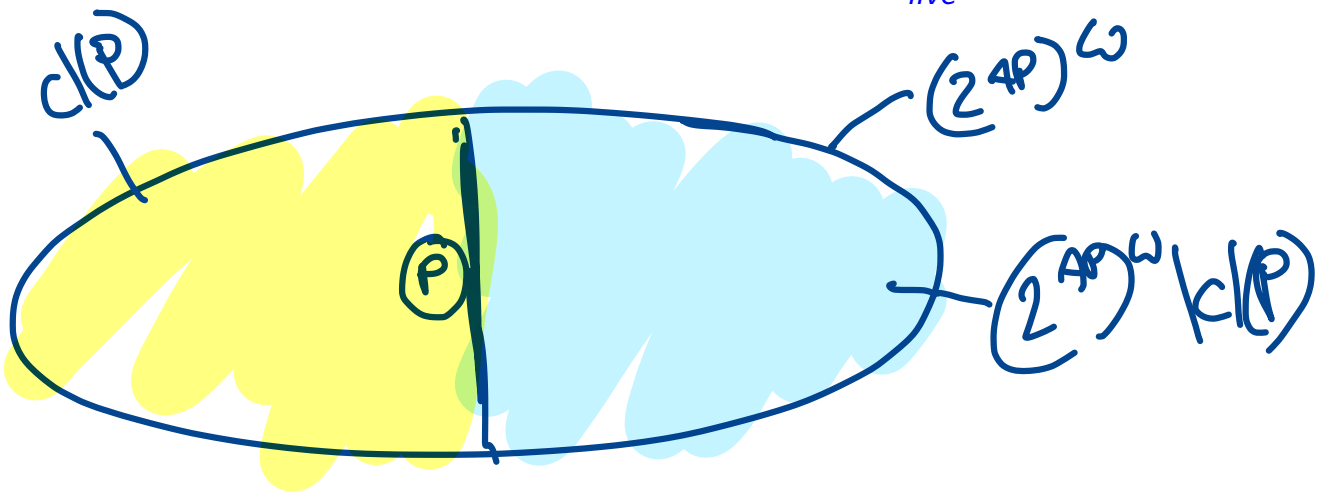
Decomposition Theorem

Decomposition theorem for LT properties

For any LT property P over AP there exists a safety property E_{safe} and a liveness property E_{live} (both over AP) such that:

$$P = E_{safe} \cap E_{live}.$$

Proposal: $P = \underbrace{cl(P)}_{=E_{safe}} \cap \underbrace{\left(P \cup \left((2^{AP})^{\omega} \setminus cl(P)\right)\right)}_{=E_{live}}$



Proof

Let P be an LT Property over AP.
Show:

1) $P = cl(P) \cap \left(P \cup \left(\left(2^{AP}\right)^{\omega} \setminus cl(P)\right)\right)$

2) $cl(P)$ is a safety property

3) $\left(P \cup \left(\left(2^{AP}\right)^{\omega} \setminus cl(P)\right)\right)$ is a liveness property

Proof

Let P be an LT Property over AP.
Show: $P = cl(P) \cap \left(P \cup \left((2^{AP})^{\omega} \setminus cl(P)\right)\right)$

By def. $P \subseteq cl(P)$ so $P = cl(P) \cap P$

$(2^{AP})^{\omega} \setminus cl(P) \cap cl(P) = \emptyset$

$P = cl(P) \cap P = cl(P) \cap \left(P \cup (2^{AP})^{\omega} \setminus cl(P)\right)$ □

Proof

Let P be an LT Property over AP.
Show: $cl(P)$ is a safety property

$E_{safe} = cl(P)$

$cl(E_{safe}) = cl(P)$

□

Slide 27:

For any LT property P over AP:

P is a safety property if and only if $cl(P) = P$.

Proof

Let P be an LT Property over AP.
Show: $(P \cup ((2^{AP})^{\omega} \setminus cl(P)))$ is a liveness property

$= (2^{AP})^{\omega}$

$cl(P \cup ((2^{AP})^{\omega} \setminus cl(P))) = cl(P) \cup cl((2^{AP})^{\omega} \setminus cl(P))$

$\supseteq cl(P) \cup (2^{AP})^{\omega} \setminus cl(P)$

$= (2^{AP})^{\omega}$

□

Slide 34: $cl(E_{live}) = (2^{AP})^{\omega}$

"Sharpest" Decomposition

Let P be an LT property and $P = E_{safe} \cap E_{live}$ where E_{safe} is a safety property and E_{live} a liveness property.

Then:

1. $cl(P) \subseteq E_{safe}$, and
2. $E_{live} \subseteq P \cup \left((2^{AP})^{\omega} \setminus cl(P)\right)$.

strongest safety property

weakest liveness property

"Sharpest" Decomposition

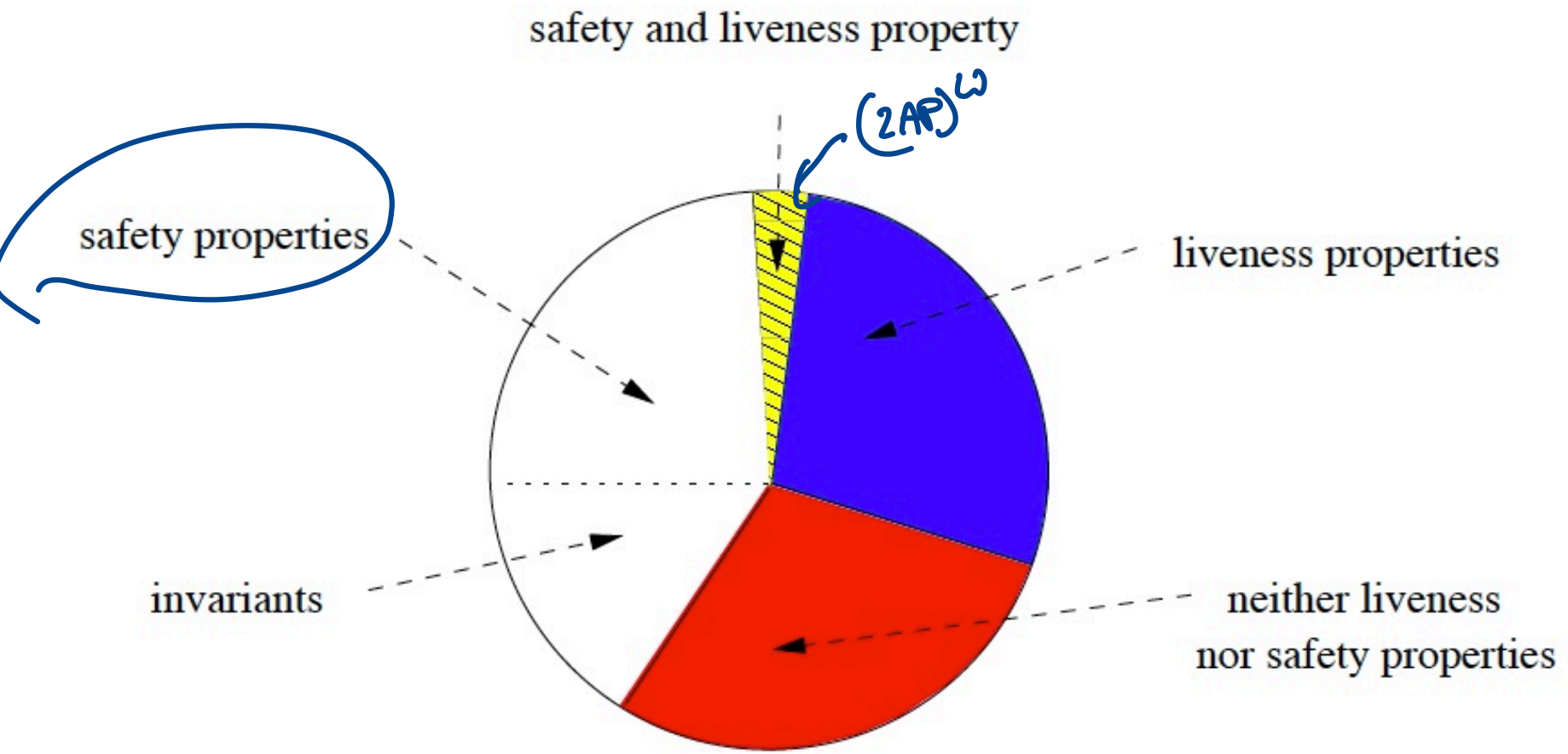
Let P be an LT property and $P = E_{safe} \cap E_{live}$ where E_{safe} is a safety property and E_{live} a liveness property.

Then:

1. $cl(P) \subseteq E_{safe}$, and
2. $E_{live} \subseteq P \cup \big((2^{AP})^{\omega} \setminus cl(P)\big)$.

$cl(P)$ is the strongest safety property and
$\big((2^{AP})^{\omega} \setminus cl(P)\big)$ the weakest liveness property

Classification of LT Properties



[Alpern and Schneider, 1987]

Summary

- LT properties are finite 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 behaviours
- A liveness property does not rule out any finite behaviour

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

Next Lecture

Monday April 25, 10:30