# Model Checking

Lecture #3: Safety and Liveness Properties

[Baier & Katoen, Chapter 3]

Joost-Pieter Katoen

Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020



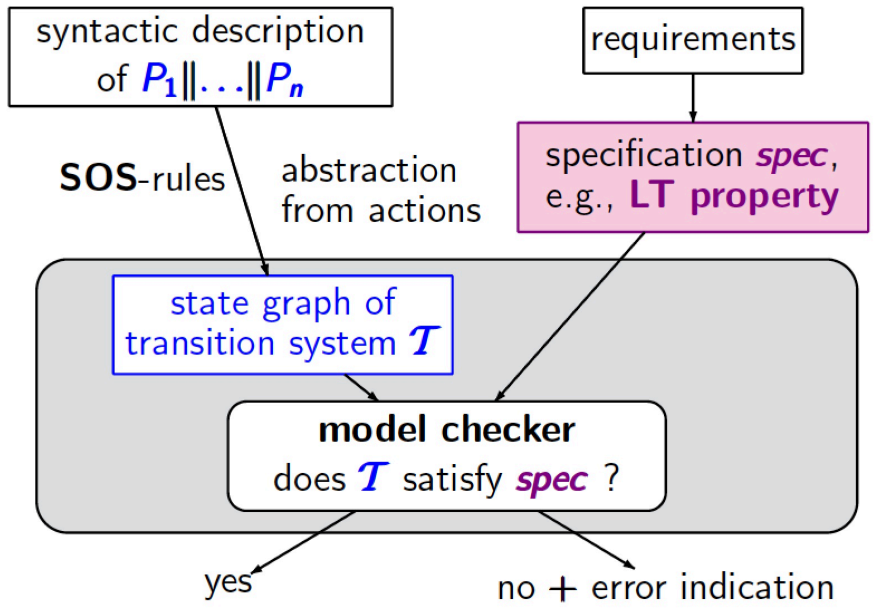


## Overview














## 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.
- The traces of a set $\Pi$ 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:

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

is:

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

Or expressed using $\omega$-regular expressions:

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





## Regular Expressions

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

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

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

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

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

- Regular expressions denote languages of finite words

## $\omega$-Regular Expressions: Syntax

**Definition: $\omega$-regular expression**

An $\omega$-regular expression G over the alphabet $\Sigma$ 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 $\Sigma$ with $\varepsilon \notin \mathfrak{L}(F_i)$.

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





## $\omega$-Regular Expressions: Semantics

**Definition: semantics of $\omega$-regular expressions**

The semantics of $\omega$-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 $\omega$-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 $\left(2^{AP}\right)^\omega$.

- 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 $AP$):

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

# 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

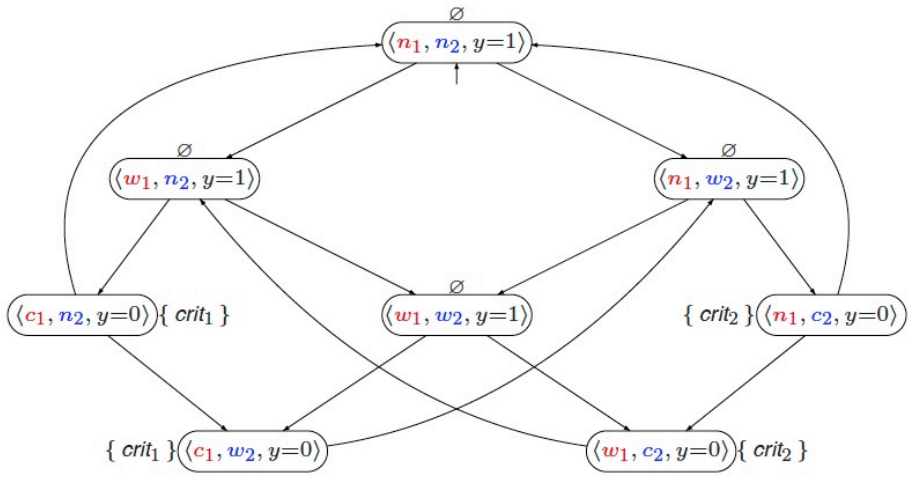
$$P_{mutex} \;=\; \text{set of infinite words } A_0\, A_1\, A_2 \ldots \\ \text{with } \{ crit_1, crit_2 \} \not\subseteq A_i \text{ for all } 0 \leq i$$

- 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




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'"*

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

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

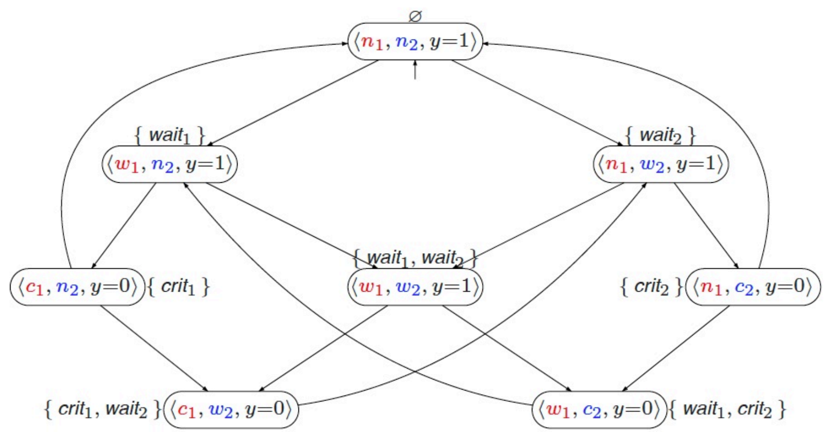
$$\left( \overset{\infty}{\exists}\, j.\ wait_i \in A_j \right) \;\Rightarrow\; \left( \overset{\infty}{\exists}\, j.\ crit_i \in A_j \right) \qquad \text{for each } i \in \{ 1, 2 \}$$

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




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 $TS \models P$.

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





## Overview







## Invariants

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

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

  where (invariant condition) $\Phi$ is a propositional logic formula over $AP$

- Note that

| | | |
|---|---|---|
| $TS \models P_{inv}$ | iff | $trace(\pi) \in P_{inv}$ for all paths $\pi$ in $TS$ |
| | iff | $L(s) \models \Phi$ for all states $s$ that belong to a path of $TS$ |
| | iff | $L(s) \models \Phi$ for all states $s \in Reach(TS)$ |

- all initial states fulfil $\Phi$ and all transitions in the reachable fragment of $TS$ preserve $\Phi$





## Example Invariants

mutual exclusion (safety):

$$MUTEX = \begin{array}{c}\text{set of all infinite words } A_0\, A_1\, A_2 \ldots \text{ s.t.}\\ \forall i \in \mathbb{N}.\ \ crit_1 \notin A_i \ \text{ or } \ crit_2 \notin A_i\end{array}$$

invariant condition: $\Phi = \neg crit_1 \vee \neg crit_2$

deadlock freedom for 5 dining philosophers:

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

invariant condition:

$$\Phi = \neg wait_0 \vee \neg wait_1 \vee \neg wait_2 \vee \neg wait_3 \vee \neg wait_4$$

here: $AP = \{wait_j : 0 \leq j \leq 4\} \cup \{\ldots\}$

## Safety Properties

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





## Safety Properties

**Definition: Safety Property**

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

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

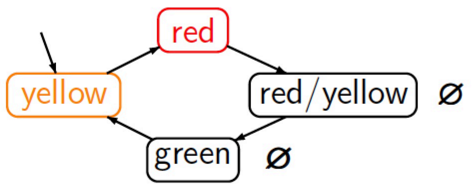
for some prefix $\widehat{\sigma}$ of $\sigma$.

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





## Examples




"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

$$\boxed{\begin{aligned} E \;=\; & \text{set of all infinite words } A_0 A_1 A_2 \ldots \\ & \text{over } 2^{AP} \text{ such that for all } i \in \mathbb{N}: \\ & red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1} \end{aligned}}$$

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

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





## Safety Properties and Finite Traces

For transition system $TS$ without terminal states and safety property $P_{safe}$:

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

## Closure

**Definition: closure of a property**

The closure of LT property $P$ is defined as:

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

- $closure(P)$ 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 $closure(P) = P$.





## 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 $P_{safe}$ : $TS' \models P_{safe} \Rightarrow TS \models P_{safe}$.

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

if and only if

$TS$ and $TS'$ satisfy the same safety properties.





## Finite versus Infinite Traces

For $TS$ without terminal states and finite $TS'$:

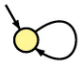
$$Traces(TS) \subseteq Traces(TS') \quad \text{iff} \quad Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$$

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

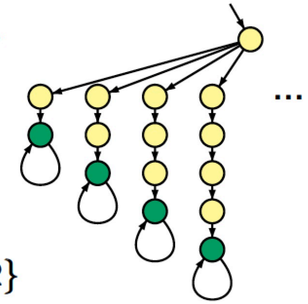
[2] Transition systems in which each state has finitely many direct successors.

## Trace Equivalence ≠ Finite Trace Equivalence

$\mathcal{T}$



$\mathcal{T}'$



$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$
$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$
$$Traces(\mathcal{T}') = \{\varnothing^n \{b\}^\omega : n \geq 2\}$$
$$Traces_{fin}(\mathcal{T}') = \{\varnothing^n : n \geq 0\} \cup \{\varnothing^n \{b\}^m : n \geq 2 \wedge m \geq 1\}$$

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

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

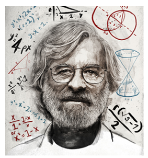
⇒ Safety properties are complemented by liveness properties
  that require some progress

- 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 $P_{live}$ over $AP$ is a *liveness* property whenever

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

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





## Example Liveness Properties for Mutual Exclusion

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

- 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
- 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 an essential characterization of LT properties

## 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 $P_{safe}$ and a liveness property $P_{live}$ (both over $AP$) such that:

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

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





## Proof





## "Sharpest" Decomposition

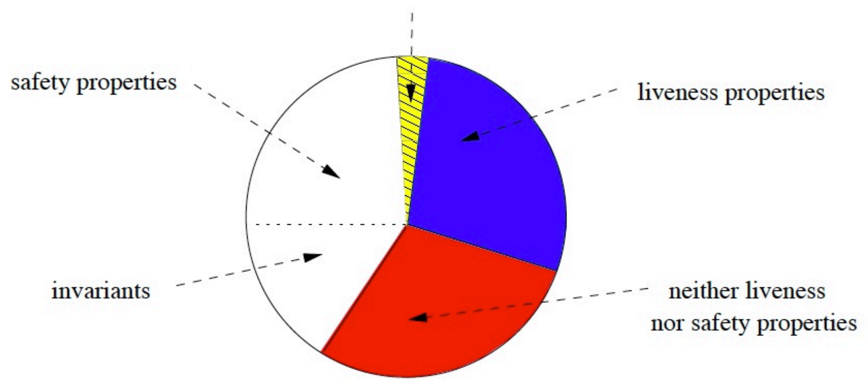
Let $P$ be an LT property and $P = P_{safe} \cap P_{live}$ where $P_{safe}$ is a safety property and $P_{live}$ a liveness property.

Then:

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

$closure(P)$ is the strongest safety property and
$\left(\left(2^{AP}\right)^{\omega} \setminus closure(P)\right)$ 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 $\Phi$ 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

Thursday October 24, 10:30