Model Checking

Linear-Time Properties

[Baier & Katoen, Chapter 3]

Prof. Dr. Nils Jansen

Radboud University, 2025

Outline of this lecture

1. Models: Transition Systems
2. **Properties**

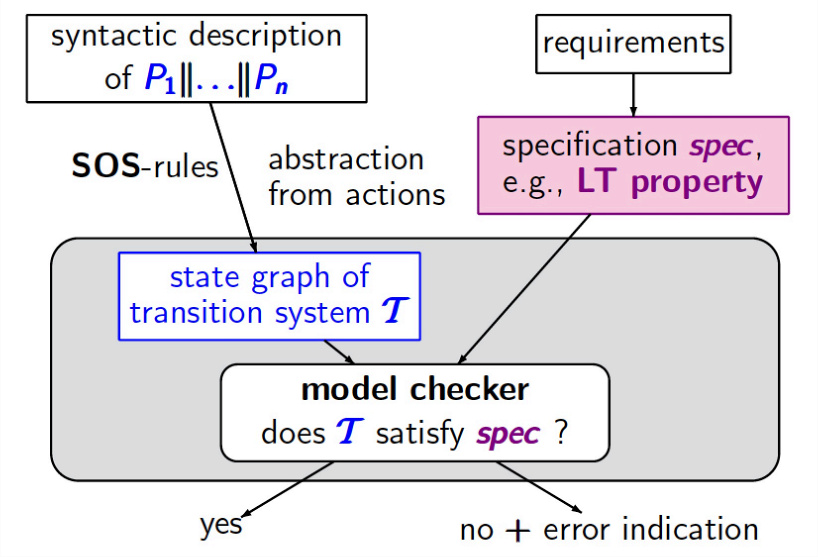
Second

How do we specify properties?

Overview

1. Recap: Traces
2. Linear-Time Properties
3. Safety Properties

Our Goal: Model Checking



Overview

1. Recap: Traces
2. Linear-Time Properties
3. Safety Properties

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.

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: $Traces(TS) = \bigcup_{s \in I} Traces(s)$.

Recap: Regular Expressions

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

$\Sigma = \{a, b\}$

Recap: Regular Expressions

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

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

$(a+b)$

$a.b$

a^*

reg expressions

$(a+b)a^*b$

a aaa b

b b

Recap: Regular Expressions

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

- Regular expressions over Σ have syntax:

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

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

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

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

Recap: Regular Expressions

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

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

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

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

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

- Regular expressions denote languages of finite words

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

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

$(a+b)^* a\, b^\omega$

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

$a a b a b\ a\ b \ldots$

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:

$$\mathsf{G} \;=\; \mathsf{E}_1.\mathsf{F}_1^{\omega} + \ldots + \mathsf{E}_n.\mathsf{F}_n^{\omega} \quad \text{for } n \in \mathbb{N}_{>0}$$

where $\mathsf{E}_i, \mathsf{F}_i$ are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(\mathsf{F}_i)$.

- 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:

$$\mathsf{G} = \mathsf{E}_1.\mathsf{F}_1^\omega + \ldots + \mathsf{E}_n.\mathsf{F}_n^\omega \quad \text{for } n \in \mathbb{N}_{>0}$$

where $\mathsf{E}_i, \mathsf{F}_i$ are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(\mathsf{F}_i)$.

- 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:

$$\mathsf{G} \;=\; \mathsf{E}_1.\mathsf{F}_1^{\omega} + \ldots + \mathsf{E}_n.\mathsf{F}_n^{\omega} \quad \text{for } n \in \mathbb{N}_{>0}$$

where $\mathsf{E}_i, \mathsf{F}_i$ are regular expressions over Σ with $\varepsilon \notin \mathfrak{L}(\mathsf{F}_i)$.

- 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 $\varnothing^{\underline{\omega}}$

Definition: semantics of ω-regular expressions
The semantics of ω-regular expression $\mathsf{G} = \mathsf{E}_1.\mathsf{F}_1^\omega + \ldots + \mathsf{E}_n.\mathsf{F}_n^\omega$ is the language $\mathfrak{L}_\omega(\mathsf{G}) \subseteq \Sigma^\omega$ defined by:

$$\mathfrak{L}_\omega(\mathsf{G}) = \mathfrak{L}(\mathsf{E}_1).\mathfrak{L}(\mathsf{F}_1)^\omega \cup \ldots \cup \mathfrak{L}(\mathsf{E}_n).\mathfrak{L}(\mathsf{F}_n)^\omega.$$

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

Definition: semantics of ω-regular expressions
The semantics of ω-regular expression $\mathsf{G} = \mathsf{E}_1.\mathsf{F}_1^\omega + \ldots + \mathsf{E}_n.\mathsf{F}_n^\omega$ is the language $\mathfrak{L}_\omega(\mathsf{G}) \subseteq \Sigma^\omega$ defined by:

$$\mathfrak{L}_\omega(\mathsf{G}) = \mathfrak{L}(\mathsf{E}_1).\mathfrak{L}(\mathsf{F}_1)^\omega \cup \ldots \cup \mathfrak{L}(\mathsf{E}_n).\mathfrak{L}(\mathsf{F}_n)^\omega.$$

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 $\mathsf{G}_1 \equiv \mathsf{G}_2$, if $\mathfrak{L}_\omega(\mathsf{G}_1) = \mathfrak{L}_\omega(\mathsf{G}_2)$.

$(a+b)^* c^\omega \equiv a^*(a+b)^* b^* c^\omega$

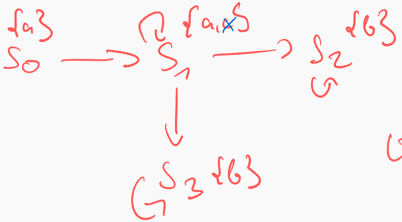
Overview

Linear-Time Properties

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

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. What does that mean?



LT property: $a^* b^\omega$

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. What does that mean?

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"

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

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

  2^{AP}

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

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

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

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

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

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

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

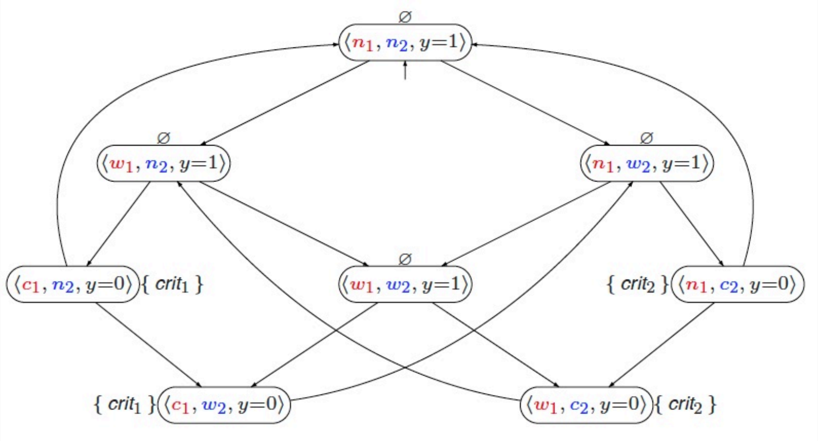
- Formalization as LT property

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

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

- 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}.

Trace Inclusion 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 $TS \models P$.

Trace Inclusion 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 $TS \models P$.

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

Overview

1. Recap: Traces
2. Linear-Time Properties
3. Safety Properties

Invariants

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

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

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

$$\Phi = (a \vee b) \wedge \neg c$$

$$\begin{tikzcd} \overset{\{a\}}{s_0} \arrow[r] & \overset{\{b\}}{s_1} \arrow[r] \arrow[d] & \overset{\{a \vee b\}}{s_2} \cdots \\ & s_3\ \{c\} & \end{tikzcd}$$

Invariants

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

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

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

- Note that

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

Invariants

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

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

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

- Note that

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

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

Example Invariants

mutual exclusion (safety):

$$MUTEX = \begin{array}{c}\text{set of all infinite words } A_0\, A_1\, A_2 \dots \text{ s.t.}\\ \forall i \in \mathbb{N}.\ \mathsf{crit}_1 \notin A_i \ \text{ or } \ \mathsf{crit}_2 \notin A_i\end{array}$$

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

deadlock freedom for 5 dining philosophers:

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

invariant condition:

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

here: $AP = \{\mathsf{wait}_j : 0 \leq j \leq 4\} \cup \{\dots\}$

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:

- 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"

  ⇒ not an invariant, since it is not a state property vs. temporal 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

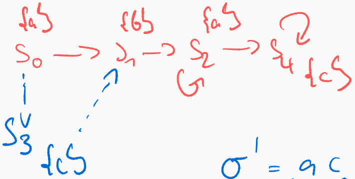
- But a safety property:
  - any infinite run violating the property has a finite prefix that is "bad"
  - i.e., in which money is withdrawn without issuing a PIN before

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.

$E_{safe} = a^* b a^* c$ "b must occur before c"



$abaac \models E_{safe}$

$\sigma' = \underbrace{ac}_{} baac$

$\hat{\sigma}$: bad prefix! (minimal)

Safety Properties

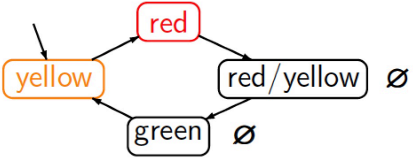
Definition: Safety Property
LT property E_{safe} over AP is a safety property if for all $\sigma \in \left(2^{AP}\right)^\omega \setminus E_{safe}$:

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

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

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

Examples



"every red phase is preceded by a yellow phase"

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

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

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

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

Safety Properties and Finite Traces

For transition system TS without terminal states

and safety property E_{safe}:

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

Closure

Definition: closure of a property
The closure of LT property P is defined as:

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

- $cl(P)$ 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 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} : $TS' \models E_{safe} \Rightarrow TS \models E_{safe}$.

$E_{safe} = a^* b a^* c$

TS'

$$\begin{tikzcd} \underset{s_0}{\{a\}} \arrow[r] & \underset{s_1}{\{b\}} \arrow[r] & \underset{s_2}{\{a\}} \arrow[r] \arrow[loop below] & \underset{s_3}{\{c\}} \end{tikzcd}$$

removing this loop gives TS with $Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$

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':

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

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

Finite versus Infinite Traces

For TS without terminal states and finite TS':

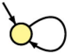
$$Traces(TS) \subseteq Traces(TS') \quad \text{iff} \quad Traces_{fin}(TS) \subseteq Traces_{fin}(TS')$$

this does not hold for infinite TS' (cf. next slide)
but also holds for image-finite TS'.[1]

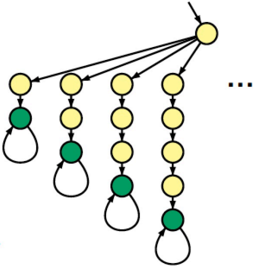
[1] Transition systems in which each state has finitely many direct successors.

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"

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

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