Lecture 8 – Trace Properties

AAA551: Programming Language Theory

Jihyeok Park

2026 Spring

Recall

- Trace Semantics
  - Transition Systems
  - Traces
  - Finite Trace Semantics
  - Infinite Trace Semantics
  - Compositionality
- Example
  - Non-Deterministic Imperative Language – NIMP

Verification

The goal of **verification** is to show that the **programs semantics** $[\![P]\!]$ satisfies a specific **desired property** $\mathbb{P}$:

$$[\![P]\!] \subseteq \mathbb{P}$$

Verification

The goal of **verification** is to show that the **programs semantics** $[\![P]\!]$ satisfies a specific **desired property** $\mathbb{P}$:

$$[\![P]\!] \subseteq \mathbb{P}$$

We will discuss classes of properties:

- **State Properties**: "Certain states are never observed"
- **Trace Properties**: "Certain traces are never observed"
  - **Safety Properties**: "Nothing bad happens"
  - **Liveness Properties**: "Something good eventually happens"

Contents

Contents

State Properties

Consider a state transition system $\mathcal{T} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$:

- $\mathbb{S}$ is a set of states
- $\rightarrow \subseteq \mathbb{S} \times \mathbb{S}$ is a transition relation
- $\mathbb{S}_{\mathcal{I}} \subseteq \mathbb{S}$ is a set of initial states

Definition (State Property)

A **state property** is a subset of states $\mathbb{P} \subseteq \mathbb{S}$.
$\mathbb{P}$ is satisfied if and only if all **reachable states** of P are belong to $\mathbb{P}$:

$$[\![P]\!]_{\mathcal{R}} \subseteq \mathbb{P}$$

where

$$[\![P]\!]_{\mathcal{R}} = \{\sigma_n \in \mathbb{S} \mid \exists \langle \sigma_0, \ldots, \sigma_n \rangle \in [\![P]\!]^*.\ \sigma_0 \in \mathbb{S}_{\mathcal{I}}\}$$

For example, *"the variable x is always positive"* is a state property.

Trace Properties

PLRG

Definition (Trace Property)

A **trace property** is a subset of traces $\mathbb{P} \subseteq \mathbb{S}^\infty$.
$\mathbb{P}$ is satisfied if and only if all traces of P are belong to $\mathbb{P}$:

$$[\![P]\!]^\infty \subseteq \mathbb{P}$$

Recall that

- $[\![P]\!]^*$ is the **finite trace semantics** of P.
- $[\![P]\!]^\omega$ is the **infinite trace semantics** of P.
- $[\![P]\!]^\infty$ is the **finite and infinite trace semantics** of P.

Trace Properties

PLRG

Definition (Trace Property)

A **trace property** is a subset of traces $\mathbb{P} \subseteq \mathbb{S}^\infty$.
$\mathbb{P}$ is satisfied if and only if all traces of P are belong to $\mathbb{P}$:

$$[\![P]\!]^\infty \subseteq \mathbb{P}$$

Recall that

- $[\![P]\!]^*$ is the **finite trace semantics** of P.
- $[\![P]\!]^\omega$ is the **infinite trace semantics** of P.
- $[\![P]\!]^\infty$ is the **finite and infinite trace semantics** of P.

Note that all state properties are trace properties.

Trace Properties

PLRG

Definition (Trace Property)

A **trace property** is a subset of traces $\mathbb{P} \subseteq \mathbb{S}^\infty$.
$\mathbb{P}$ is satisfied if and only if all traces of P are belong to $\mathbb{P}$:

$$[\![P]\!]^\infty \subseteq \mathbb{P}$$

Recall that

- $[\![P]\!]^*$ is the **finite trace semantics** of P.
- $[\![P]\!]^\omega$ is the **infinite trace semantics** of P.
- $[\![P]\!]^\infty$ is the **finite and infinite trace semantics** of P.

Note that all state properties are trace properties.

For example, *"the variable x is eventually becomes zero"* is a trace property but not a state property.

Verification

To **verify** a property $\mathbb{P}$ of a program P, we need to show that the program semantics $[\![P]\!]$ is a subset of the property $\mathbb{P}$:

Definition (Verification)

A program P **satisfies** a property $\mathbb{P}$ if and only if:

$$[\![P]\!] \subseteq \mathbb{P}$$

Verification

To **verify** a property $\mathbb{P}$ of a program P, we need to show that the program semantics $[\![P]\!]$ is a subset of the property $\mathbb{P}$:

Definition (Verification)

A program P **satisfies** a property $\mathbb{P}$ if and only if:

$$[\![P]\!] \subseteq \mathbb{P}$$

Definition (Stronger Property)

A $\mathbb{P}$ is **stronger** than $\mathbb{P}'$ if and only if $\mathbb{P}$ is a subset of $\mathbb{P}'$:

$$\mathbb{P} \subseteq \mathbb{P}'$$

If a program satisfies a stronger property, it also satisfies a weaker property:

$$\mathbb{P} \subseteq \mathbb{P}' \quad \text{and} \quad [\![P]\!] \subseteq \mathbb{P} \implies [\![P]\!] \subseteq \mathbb{P}'$$

Contents

Safety Properties

The informal definition of **safety properties** is:

"Nothing bad happens"

Safety Properties

The informal definition of **safety properties** is:

"Nothing bad happens"

For example,

- *"the variable x is always positive"* is a safety property.
  (**bad thing**: reaching a state where $x \leq 0$)
- *"the output of the sort function is always sorted"* is a safety property.
  (**bad thing**: producing an output that is not sorted)

Safety Properties

The informal definition of **safety properties** is:

"Nothing bad happens"

For example,

- *"the variable x is always positive"* is a safety property.
  (**bad thing**: reaching a state where $x \leq 0$)
- *"the output of the sort function is always sorted"* is a safety property.
  (**bad thing**: producing an output that is not sorted)

All state properties are safety properties, but there exist safety properties that are not state properties. For example,

- *"output of the sort function should have same elements as the input"* is a safety but not a state property.
  (**bad thing**: producing an output that does not have the same elements as the input)

Safety Properties – Irreducibility

Let's formalize the definition of safety properties.

Safety Properties – Irreducibility

Let's formalize the definition of safety properties.

Intuitively, a safety property $\mathbb{P}$ satisfies the following two conditions:

1. **Irreducibility**: *"Once a bad thing happens, it cannot be undone."*

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \notin \mathbb{P} \implies \tau' \notin \mathbb{P})$$

Safety Properties – Irreducibility

Let's formalize the definition of safety properties.

Intuitively, a safety property $\mathbb{P}$ satisfies the following two conditions:

1. **Irreducibility**: *"Once a bad thing happens, it cannot be undone."*

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \notin \mathbb{P} \implies \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \in \mathbb{P} \impliedby \tau' \in \mathbb{P})$$

Safety Properties – Irreducibility

Let's formalize the definition of safety properties.

Intuitively, a safety property $\mathbb{P}$ satisfies the following two conditions:

1. **Irreducibility**: *"Once a bad thing happens, it cannot be undone."*

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \notin \mathbb{P} \implies \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \in \mathbb{P} \impliedby \tau' \in \mathbb{P})$$

Thus, a safety property is **prefix closed**:

"If a trace $\tau' \in \mathbb{S}^{\infty}$ is in $\mathbb{P}$, then all prefixes of τ' are also in $\mathbb{P}$."

Safety Properties – Irreducibility

Let's formalize the definition of safety properties.

Intuitively, a safety property $\mathbb{P}$ satisfies the following two conditions:

1. **Irreducibility**: *"Once a bad thing happens, it cannot be undone."*

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \notin \mathbb{P} \implies \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau, \tau' \in \mathbb{S}^{\infty}. \qquad \tau \preceq \tau' \implies (\tau \in \mathbb{P} \impliedby \tau' \in \mathbb{P})$$

Thus, a safety property is **prefix closed**:

"If a trace $\tau' \in \mathbb{S}^{\infty}$ is in $\mathbb{P}$, then all prefixes of τ' are also in $\mathbb{P}$."

Let's define a **prefix closure operator** PCl.

Safety Properties – Finite Bad Prefix

2. **Finite Bad Prefix**: *"A bad thing can be observed in a finite time."*

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \notin \mathbb{P} \implies (\exists \tau' \in \mathbb{S}^{*}. \quad \tau' \preceq \tau \quad \wedge \quad \tau' \notin \mathbb{P})$$

Safety Properties – Finite Bad Prefix

2. **Finite Bad Prefix**: *"A bad thing can be observed in a finite time."*

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \notin \mathbb{P} \implies (\exists \tau' \in \mathbb{S}^{*}. \quad \tau' \preceq \tau \quad \wedge \quad \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \in \mathbb{P} \impliedby (\forall \tau' \in \mathbb{S}^{*}. \quad (\tau' \preceq \tau \implies \tau' \in \mathbb{P}))$$

Safety Properties – Finite Bad Prefix

2. **Finite Bad Prefix**: *"A bad thing can be observed in a finite time."*

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \notin \mathbb{P} \implies (\exists \tau' \in \mathbb{S}^{*}. \quad \tau' \preceq \tau \quad \wedge \quad \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \in \mathbb{P} \impliedby (\forall \tau' \in \mathbb{S}^{*}. \quad (\tau' \preceq \tau \implies \tau' \in \mathbb{P}))$$

In other words,

"If all finite prefixes of a trace $\tau \in \mathbb{S}^{\infty}$ are in $\mathbb{P}$, then τ is in $\mathbb{P}$."

Safety Properties – Finite Bad Prefix

2. **Finite Bad Prefix**: *"A bad thing can be observed in a finite time."*

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \notin \mathbb{P} \implies (\exists \tau' \in \mathbb{S}^{*}. \quad \tau' \preceq \tau \quad \wedge \quad \tau' \notin \mathbb{P})$$

which is equivalent to:

$$\forall \tau \in \mathbb{S}^{\infty}. \quad \tau \in \mathbb{P} \impliedby (\forall \tau' \in \mathbb{S}^{*}. \quad (\tau' \preceq \tau \implies \tau' \in \mathbb{P}))$$

In other words,

"If all finite prefixes of a trace $\tau \in \mathbb{S}^{\infty}$ are in $\mathbb{P}$, then τ is in $\mathbb{P}$."

We need to defined a **prefix limit operator** Lim.

Prefix Closure

PLRG

Definition (Prefix Closure)

The **prefix closure operator** $\mathsf{PCl} : \mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^*)$ is defined as:

$$\mathsf{PCl}(X) = \{\tau' \in \mathbb{S}^* \mid \exists \tau \in X.\, \tau' \preceq \tau\}$$

Note that it takes a set of **finite and infinite traces** X, but returns a set of **finite traces** that are prefixes of the traces in X.

Prefix Closure

PLRG

Definition (Prefix Closure)

The **prefix closure operator** $\mathsf{PCl} : \mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^*)$ is defined as:

$$\mathsf{PCl}(X) = \{\tau' \in \mathbb{S}^* \mid \exists \tau \in X.\, \tau' \preceq \tau\}$$

Note that it takes a set of **finite and infinite traces** X, but returns a set of **finite traces** that are prefixes of the traces in X.

The PCl operator is:

- **monotone**: $X \subseteq Y \implies \mathsf{PCl}(X) \subseteq \mathsf{PCl}(Y)$
- **idempotent**: $\mathsf{PCl}(\mathsf{PCl}(X)) = \mathsf{PCl}(X)$

Prefix Limit

Definition (Prefix Limit)

The **prefix limit operator** $\mathsf{Lim} : \mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ is defined as:

$$\mathsf{Lim}(X) = X \cup \{\tau \in \mathbb{S}^\infty \mid \forall \tau' \in \mathbb{S}^*.\ (\tau' \preceq \tau \implies \tau' \in X)\}$$

Prefix Limit

Definition (Prefix Limit)

The **prefix limit operator** $\mathsf{Lim} : \mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ is defined as:

$$\mathsf{Lim}(X) = X \cup \{\tau \in \mathbb{S}^\infty \mid \forall \tau' \in \mathbb{S}^*.\ (\tau' \preceq \tau \implies \tau' \in X)\}$$

The Lim operator is:

- **monotone**: $X \subseteq Y \implies \mathsf{Lim}(X) \subseteq \mathsf{Lim}(Y)$
- **extensive**: $X \subseteq \mathsf{Lim}(X)$
- **idempotent**: $\mathsf{Lim}(\mathsf{Lim}(X)) = \mathsf{Lim}(X)$

Formal Definition of Safety Properties

Let's say a trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a safety property if and only if its **limit of prefix closure** is itself:

Definition (Safety Property)

A trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a **safety property** if and only if:

$$\mathsf{Safe}(\mathbb{P}) = \mathbb{P}$$

where $\mathsf{Safe} = \mathsf{Lim} \circ \mathsf{PCl}$

Formal Definition of Safety Properties

Let's say a trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a safety property if and only if its **limit of prefix closure** is itself:

Definition (Safety Property)

A trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a **safety property** if and only if:

$$\text{Safe}(\mathbb{P}) = \mathbb{P}$$

where $\text{Safe} = \text{Lim} \circ \text{PCl}$

One interesting fact is that we can always construct a safety property from any trace property by applying the Safe operator:

Theorem

For a given trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$, $\text{Safe}(\mathbb{P})$ **is a safety property***:*

$$\text{Safe}(\text{Safe}(\mathbb{P})) = \text{Safe}(\mathbb{P})$$

Formal Definition of Safety Properties

Let's say a trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a safety property if and only if its **limit of prefix closure** is itself:

Definition (Safety Property)

A trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a **safety property** if and only if:

$$\text{Safe}(\mathbb{P}) = \mathbb{P}$$

where $\text{Safe} = \text{Lim} \circ \text{PCl}$

One interesting fact is that we can always construct a safety property from any trace property by applying the Safe operator:

Theorem

For a given trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$, $\text{Safe}(\mathbb{P})$ **is a safety property***:*

$$\text{Safe}(\text{Safe}(\mathbb{P})) = \text{Safe}(\mathbb{P})$$

We need to prove that Safe is a **idempotent**.

Safe is Upper Closure Operator

Let's prove that the Safe operator is an upper closure operator.

Theorem

The Safe : $\mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ *operator is an* **upper closure operator**, *which satisfies the following properties:*

- **monotone**: $X \subseteq X' \implies \text{Safe}(X) \subseteq \text{Safe}(X')$
- **extensive**: $X \subseteq \text{Safe}(X)$
- **idempotent**: $\text{Safe}(\text{Safe}(X)) = \text{Safe}(X)$

Formal Definition of Safety Properties

- Safe is **monotone**:

$$X \subseteq X' \implies \mathsf{Safe}(X) \subseteq \mathsf{Safe}(X')$$

Since PCI and Lim are monotone, their composition is also monotone.

Formal Definition of Safety Properties

- Safe is **monotone**:

$$X \subseteq X' \implies \mathsf{Safe}(X) \subseteq \mathsf{Safe}(X')$$

  Since PCI and Lim are monotone, their composition is also monotone.

- Safe is **extensive**:

$$X \subseteq \mathsf{Safe}(X)$$

  - If $\tau \in X \cap \mathbb{S}^*$, it is a prefix of itself, and Lim is extensive, so:

$$\tau \in \mathsf{PCI}(X) \subseteq \mathsf{Lim}(\mathsf{PCI}(X)) = \mathsf{Safe}(X)$$

  - If $\tau \in X \cap \mathbb{S}^\omega$, then all finite prefixes of τ are in $\mathsf{PCI}(X)$, so $\tau \in \mathsf{Lim}(\mathsf{PCI}(X)) = \mathsf{Safe}(X)$.

Formal Definition of Safety Properties

- Safe is **idempotent**:

$$\mathsf{Safe}(\mathsf{Safe}(X)) = \mathsf{Safe}(X)$$

Let's prove the two directions separately:

- $\boxed{\mathsf{Safe}(X) \subseteq \mathsf{Safe}(\mathsf{Safe}(X))}$:

  Since Safe is extensive, we are done.

- $\boxed{\mathsf{Safe}(\mathsf{Safe}(X)) \subseteq \mathsf{Safe}(X)}$:

$\tau \in \mathsf{Safe}(\mathsf{Safe}(\mathbb{P}))$

$\implies \forall \tau' \in \mathbb{S}^*.\ (\tau' \preceq \tau. \Rightarrow \tau' \in \mathsf{PCl} \circ \mathsf{Safe}(\mathbb{P}))$ (by def. of Lim)

$\implies \forall \tau' \in \mathbb{S}^*.\ (\tau' \preceq \tau. \Rightarrow \exists \tau'' \in \mathsf{Safe}(\mathbb{P}).\ \tau' \preceq \tau'')$ (by def. of PCl)

$\implies \forall \tau' \in \mathbb{S}^*.\ (\tau' \preceq \tau. \Rightarrow \tau' \in \mathsf{PCl}(\mathbb{P}))$ (by def. of Lim)

$\implies \tau \in \mathsf{Safe}(\mathbb{P})$ (by def. of Lim)

Verification by Invariance

To verify that a program P satisfies a safety property $\mathbb{P}$, we need to think about **invariance** (i.e., "what is always true?") of the program.

Verification by Invariance

To verify that a program P satisfies a safety property $\mathbb{P}$, we need to think about **invariance** (i.e., "what is always true?") of the program.

Definition (Invariance)

Given transition system $\mathcal{T} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$ for a program P,

$$[\![\mathcal{T}]\!]^* = \mathbf{lfp}(F_*) = \bigcup_{n \in \mathbb{Z}} F_*^n(\varnothing)$$

where $\mathcal{I} = \{\epsilon\} \cup \{\langle \sigma \rangle \mid \sigma \in \mathbb{S}\}$ and $F_* : \mathcal{P}(\mathbb{S}^*) \rightarrow \mathcal{P}(\mathbb{S}^*)$:

$$F_*(X) = \mathcal{I} \cup \{\langle \sigma_0, \ldots, \sigma_n, \sigma' \rangle \mid \langle \sigma_0, \ldots, \sigma_n \rangle \in X \wedge \sigma_n \rightarrow \sigma'\}$$

A set of finite traces $\mathbb{I} \subseteq \mathbb{S}^*$ is an **invariant** if and only if $\mathbb{I}$ is closed under F_*:

$$F_*(\mathbb{I}) \subseteq \mathbb{I}$$

Verification by Invariance – Example

ℓ_0 : $(\!|\texttt{true}|\!)$

```
s = 0;
```

ℓ_1 : $(\!|s = 0|\!)$

```
i = 0;
```

ℓ_2 : $(\!|i = 0 \wedge s = 0|\!)$

```
while (i < n) {
```

ℓ_3 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
    s = s + x[i];
```

ℓ_4 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i} x[k]|\!)$

```
    i = i + 1;
```

ℓ_5 : $(\!|1 \leq i \leq n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
}
```

ℓ_6 : $(\!|i = n \wedge s = \sum_{k=0}^{n-1} x[k]|\!)$

Consider a program that computes the sum of an array x of size n.

Verification by Invariance – Example

PLRG

ℓ_0 : $(\!|\texttt{true}|\!)$

```
s = 0;
```

ℓ_1 : $(\!|s = 0|\!)$

```
i = 0;
```

ℓ_2 : $(\!|i = 0 \wedge s = 0|\!)$

```
while (i < n) {
```

ℓ_3 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
    s = s + x[i];
```

ℓ_4 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i} x[k]|\!)$

```
    i = i + 1;
```

ℓ_5 : $(\!|1 \leq i \leq n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
}
```

ℓ_6 : $(\!|i = n \wedge s = \sum_{k=0}^{n-1} x[k]|\!)$

Consider a program that computes the sum of an array x of size n.

The **safety property** we want to verify is $s = \sum_{k=0}^{n-1} x[k]$ at the end of the program.

Verification by Invariance – Example

ℓ_0 : $(\!|\texttt{true}|\!)$

```
s = 0;
```

ℓ_1 : $(\!|s = 0|\!)$

```
i = 0;
```

ℓ_2 : $(\!|i = 0 \wedge s = 0|\!)$

```
while (i < n) {
```

ℓ_3 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
    s = s + x[i];
```

ℓ_4 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i} x[k]|\!)$

```
    i = i + 1;
```

ℓ_5 : $(\!|1 \leq i \leq n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
}
```

ℓ_6 : $(\!|i = n \wedge s = \sum_{k=0}^{n-1} x[k]|\!)$

Consider a program that computes the sum of an array x of size n.

The **safety property** we want to verify is $s = \sum_{k=0}^{n-1} x[k]$ at the end of the program.

We define the **local invariant** at each program point $\ell_i \in \mathbb{L}$ as a memory predicate m_i that holds at ℓ_i.

Verification by Invariance – Example

PLRG

ℓ_0 : $(\!|\texttt{true}|\!)$

```
s = 0;
```

ℓ_1 : $(\!|s = 0|\!)$

```
i = 0;
```

ℓ_2 : $(\!|i = 0 \wedge s = 0|\!)$

```
while (i < n) {
```

ℓ_3 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
    s = s + x[i];
```

ℓ_4 : $(\!|0 \leq i < n \wedge s = \sum_{k=0}^{i} x[k]|\!)$

```
    i = i + 1;
```

ℓ_5 : $(\!|1 \leq i \leq n \wedge s = \sum_{k=0}^{i-1} x[k]|\!)$

```
}
```

ℓ_6 : $(\!|i = n \wedge s = \sum_{k=0}^{n-1} x[k]|\!)$

Consider a program that computes the sum of an array x of size n.

The **safety property** we want to verify is $s = \sum_{k=0}^{n-1} x[k]$ at the end of the program.

We define the **local invariant** at each program point $\ell_i \in \mathbb{L}$ as a memory predicate m_i that holds at ℓ_i.

The **global invariant** is the union of all local invariants:

$$\mathbb{I} = \{\langle \ell_0, m_0 \rangle, \ldots, \langle \ell_6, m_6 \rangle\}$$

Verification by Invariance – Soundness

PLRG

Theorem (Soundness)

If $\mathbb{I}$ is an invariant of a program P and $\mathbb{I}$ is stronger than a safety property $\mathbb{P}$, then P satisfies $\mathbb{P}$.

Verification by Invariance – Soundness

PLRG

Theorem (Soundness)

If $\mathbb{I}$ is an invariant of a program P and $\mathbb{I}$ is stronger than a safety property $\mathbb{P}$, then P satisfies $\mathbb{P}$.

Since $F_*(\mathbb{I}) \subseteq \mathbb{I}$ and F_* is monotone, we have:

$$\varnothing \subseteq \mathbb{I} \implies F_*(\varnothing) \subseteq F_*(\mathbb{I}) \subseteq \mathbb{I} \implies \ldots F_*^n(\varnothing) \subseteq \mathbb{I}$$

Thus, $[\![P]\!]^* = \mathbf{lfp}(F_*) = \bigcup_{n\in\mathbb{Z}} F_*^n(\varnothing) \subseteq \mathbb{I}$.

Now, we have the following:

$$\begin{aligned}
[\![P]\!]^\infty &= \mathsf{Safe}([\![P]\!]^*) && \text{(by def. of Safe)} \\
&\subseteq \mathsf{Safe}(\mathbb{I}) && \text{(since above)} \\
&\subseteq \mathsf{Safe}(\mathbb{P}) && \text{(since Safe is monotone and } \mathbb{I} \subseteq \mathbb{P}) \\
&= \mathbb{P} && \text{(since } \mathbb{P} \text{ is a safety property)}
\end{aligned}$$

Verification by Invariance – Completeness

PLRG

Theorem (Completeness)

If a program P satisfies a safety property $\mathbb{P}$, then there exists an invariant $\mathbb{I}$ such that $\mathbb{I}$ is stronger than $\mathbb{P}$.

Verification by Invariance – Completeness

PLRG

Theorem (Completeness)

If a program P satisfies a safety property $\mathbb{P}$, then there exists an invariant $\mathbb{I}$ such that $\mathbb{I}$ is stronger than $\mathbb{P}$.

The program semantics itself is an invariant that is stronger than the safety property:

$$\mathbb{I} = [\![P]\!]^*$$

since it is a least fixed point of F_*, it is closed under F_*.

Then, since P satisfies $\mathbb{P}$, we have $\mathbb{I} = [\![P]\!]^* \subseteq \mathbb{P}$.

Verification by Invariance – Completeness

PLRG

Theorem (Completeness)

If a program P satisfies a safety property $\mathbb{P}$, then there exists an invariant $\mathbb{I}$ such that $\mathbb{I}$ is stronger than $\mathbb{P}$.

The program semantics itself is an invariant that is stronger than the safety property:

$$\mathbb{I} = [\![P]\!]^*$$

since it is a least fixed point of F_*, it is closed under F_*.

Then, since P satisfies $\mathbb{P}$, we have $\mathbb{I} = [\![P]\!]^* \subseteq \mathbb{P}$.

It does **NOT** mean that we can always **algorithmically construct** an **invariant** from a safety property, since the program semantics is typically **undecidable**.

Contents

Liveness Properties

The informal definition of **liveness properties** is:

"Something good eventually happens"

For example,

- *"the variable x eventually becomes zero"* is a liveness property.
  (**good thing**: reaching a state where $x = 0$)

- *"a program always terminates"* is a liveness property.
  (**good thing**: reaching a terminal state)

Eventuality

How to formalize the following **eventuality** of liveness properties?

"Something good eventually happens"

We can formalize it as follows:

$$\forall \tau \in \mathbb{S}^*. \qquad \exists \tau' \in \mathbb{P}. \qquad \tau \preceq \tau'$$

By definition of PCI, the above is equivalent to:

$$\forall \tau \in \mathbb{S}^*. \qquad \tau \in \mathsf{PCI}(\mathbb{P})$$

which means that **(1)** $\mathsf{PCI}(\mathbb{P}) = \mathbb{S}^*$.

If we apply Lim to both sides, we have **(2)** $\mathsf{Lim} \circ \mathsf{PCI}(\mathbb{P}) = \mathsf{Lim}(\mathbb{S}^*) = \mathbb{S}^\infty$.

Finally, **(3)** $\mathbb{P} = \mathbb{P} \cup \varnothing = \mathbb{P} \cup (\mathbb{S}^\infty \setminus \mathsf{Lim} \circ \mathsf{PCI}(\mathbb{P})) = \mathbb{P} \cup (\mathbb{S}^\infty \setminus \mathsf{Safe}(\mathbb{P}))$.

Formal Definition of Liveness Properties

In summary, we have the following three equivalent definitions of liveness properties:

Definition (Liveness Property)

A trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$ is a **liveness property** if and only if it satisfies one of the following equivalent conditions:

1. $\mathsf{PCl}(\mathbb{P}) = \mathbb{S}^*$
2. $\mathsf{Lim} \circ \mathsf{PCl}(\mathbb{P}) = \mathbb{S}^\infty$
3. $\mathsf{Live}(\mathbb{P}) = \mathbb{P}$

where $\mathsf{Live} : \mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ is defined as:

$$\mathsf{Live}(\mathbb{P}) = \mathbb{P} \cup (\mathbb{S}^\infty \setminus \mathsf{Safe}(\mathbb{P}))$$

Live is Idempotent and Extensive

PLRG

Theorem

The Live : $\mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ *operator satisfies the following properties:*

- **idempotent***:* $\text{Live}(\text{Live}(X)) = \text{Live}(X)$
- **extensive***:* $X \subseteq \text{Live}(X)$

Note that Live is **not monotone** while Safe is monotone.

Live is Idempotent and Extensive

PLRG

Theorem

The Live : $\mathcal{P}(\mathbb{S}^\infty) \to \mathcal{P}(\mathbb{S}^\infty)$ *operator satisfies the following properties:*

- **idempotent***:* $\text{Live}(\text{Live}(X)) = \text{Live}(X)$
- **extensive***:* $X \subseteq \text{Live}(X)$

Note that Live is **not monotone** while Safe is monotone.

Since Live is idempotent, the following is true:

Theorem

For a given trace property $\mathbb{P} \subseteq \mathbb{S}^\infty$, $\text{Live}(\mathbb{P})$ **is a liveness property***:*

$$\text{Live}(\text{Live}(\mathbb{P})) = \text{Live}(\mathbb{P})$$

Verification by Variance

To verify that a program P satisfies a liveness property $\mathbb{P}$, we need to consider a **ranking function** that maps a state to a well-founded set, which measures how far the program is from reaching a good state.

Verification by Variance

To verify that a program P satisfies a liveness property $\mathbb{P}$, we need to consider a **ranking function** that maps a state to a well-founded set, which measures how far the program is from reaching a good state.

Definition (Ranking Function)

For a given transition system $\mathcal{T} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$ for a program P, a **ranking function** is a function $\phi : \mathbb{S} \rightarrow W$ satisfying the following condition:

- $(W, \sqsubseteq)$ is **well-founded ordering**,
  which means that all non-empty subset of W has a minimal element:

$$\forall S \subseteq W. \qquad S \neq \varnothing \implies \exists m \in S. \qquad \forall x \in S. \qquad \neg(x \sqsubset m)$$

- All transitions decrease the rank:

$$\forall \sigma, \sigma' \in \mathbb{S}. \qquad \sigma \rightarrow \sigma' \implies \phi(\sigma) \sqsupset \phi(\sigma')$$

Verification by Variance – Example

Let's define a ranking function for the sum of an array program.

```
ℓ0 : s = 0;
ℓ1 : i = 0;
ℓ2 : while (i < n) {
ℓ3 :     s = s + x[i];
ℓ4 :     i = i + 1;
ℓ5 : }
ℓ6 :
```

Ranking function:

$$
\begin{array}{rlll}
\phi: & \mathbb{S} & \to & \mathbb{N} \\
 & \langle \ell_0, m \rangle & \mapsto & 3n+6 \\
 & \langle \ell_1, m \rangle & \mapsto & 3n+5 \\
 & \langle \ell_2, m \rangle & \mapsto & 3n+4 \\
 & \langle \ell_3, m \rangle & \mapsto & 3(n-m(i))+3 \\
 & \langle \ell_4, m \rangle & \mapsto & 3(n-m(i))+2 \\
 & \langle \ell_5, m \rangle & \mapsto & 3(n-m(i))+4 \\
 & \langle \ell_6, m \rangle & \mapsto & 0
\end{array}
$$

We skip the soundness and completeness of verification by variance.

Contents

Decomposition of Trace Properties

PLRG

Theorem

Any trace property $\mathbb{P}$ *can be decomposed into the intersection of a safety property* $\mathsf{Safe}(\mathbb{P})$ *and a liveness property* $\mathsf{Live}(\mathbb{P})$*:*

$$\mathbb{P} = \mathsf{Safe}(\mathbb{P}) \cap \mathsf{Live}(\mathbb{P})$$

Decomposition of Trace Properties

PLRG

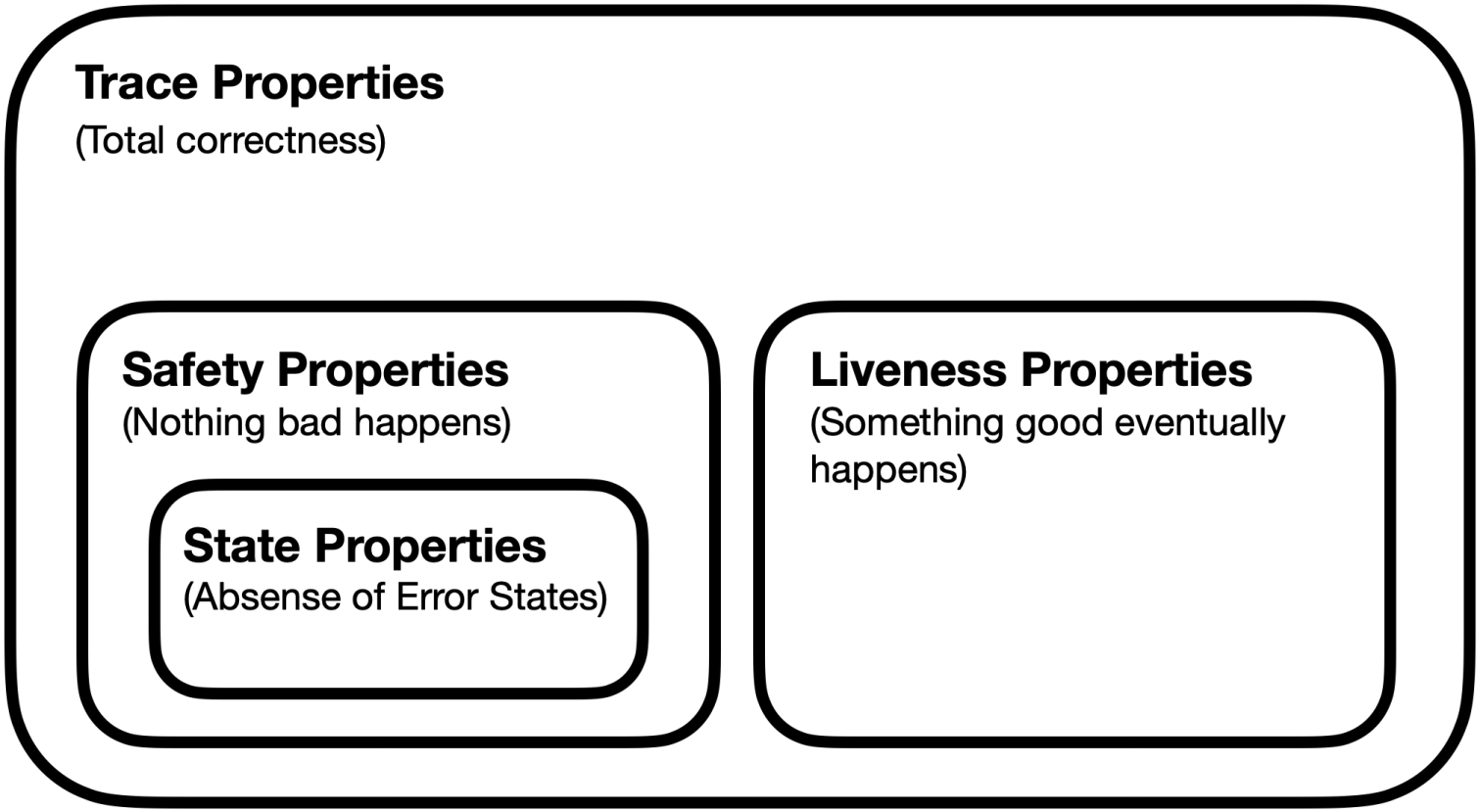
Theorem

Any trace property $\mathbb{P}$ *can be decomposed into the intersection of a safety property* $\mathsf{Safe}(\mathbb{P})$ *and a liveness property* $\mathsf{Live}(\mathbb{P})$*:*

$$\mathbb{P} = \mathsf{Safe}(\mathbb{P}) \cap \mathsf{Live}(\mathbb{P})$$

$$\begin{aligned}
\mathsf{Safe}(\mathbb{P}) \cap \mathsf{Live}(\mathbb{P}) &= \mathsf{Safe}(\mathbb{P}) \cap (\mathbb{P} \cup (\mathbb{S}^\infty \setminus \mathsf{Safe}(\mathbb{P}))) \\
&\quad \text{(by def. of Live)} \\
&= (\mathsf{Safe}(\mathbb{P}) \cap \mathbb{P}) \cup (\mathsf{Safe}(\mathbb{P}) \cap (\mathbb{S}^\infty \setminus \mathsf{Safe}(\mathbb{P}))) \\
&\quad \text{(by distributive law of } \cap \text{ over } \cup) \\
&= \mathbb{P} \cup \varnothing \\
&\quad \text{(by Safe is extensive: } \mathbb{P} \subseteq \mathsf{Safe}(\mathbb{P})) \\
&= \mathbb{P}
\end{aligned}$$

Decomposition of Trace Properties



The proof of any trace property can be decomposed into:

- proof of a safety property Safe($\mathbb{P}$) by invariance
- proof of a liveness property Live($\mathbb{P}$) by variance

Next Lecture

- Abstract Machines

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr