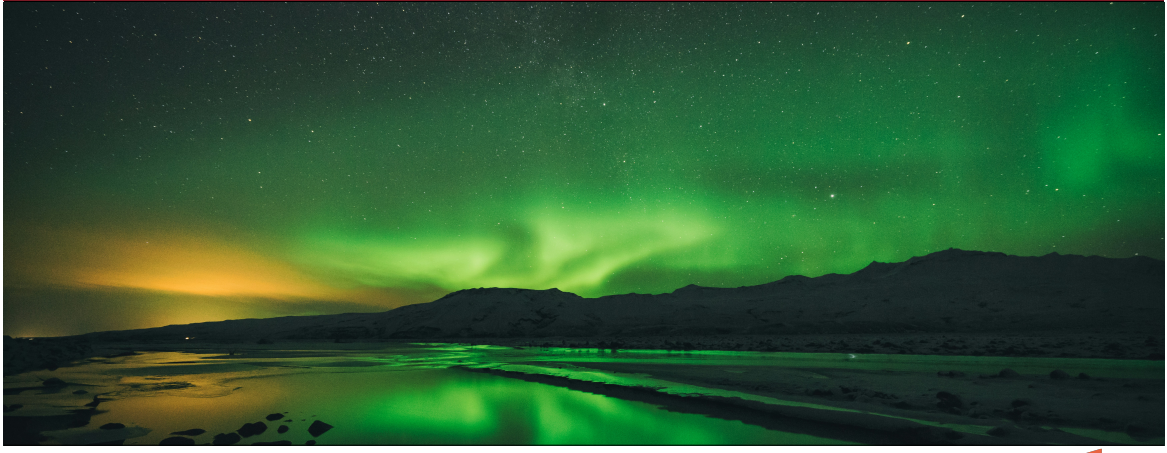
A Sequent Calculus For Trace Formula Implication

Niklas Heidler
Joint Work with Reiner Hähnle

TECHNISCHE UNIVERSITÄT DARMSTADT



Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Trace formulas** are a novel logic that
  - specifies **whole sets of program traces**
  - can even specify recursive traces using **fixpoint operations**
  - allows the specification of
    - precisely all traces of simple imperative programs
    - abstract trace properties

Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Trace formulas** are a novel logic that
  - specifies **whole sets of program traces**
  - can even specify recursive traces using **fixpoint operations**
  - allows the specification of
    - precisely all traces of simple imperative programs
    - abstract trace properties
- Verification if a program **satisfies** a trace formula requires trace formula **weakening**

Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Trace formulas** are a novel logic that
  - specifies **whole sets of program traces**
  - can even specify recursive traces using **fixpoint operations**
  - allows the specification of
    - precisely all traces of simple imperative programs
    - abstract trace properties
- Verification if a program **satisfies** a trace formula requires trace formula **weakening**
- **Main Contribution:** Calculus for Trace Formula Implication

Connection Between Programs and Formulas

by Gurov & Hähnle, FSCD, 2025

TECHNISCHE UNIVERSITÄT DARMSTADT

For every Rec program **P**, there exists a **strongest trace formula** that specifies **precisely** the traces generated by **P**.

Software Engineering Group

Rec Programs

TECHNISCHE UNIVERSITÄT DARMSTADT

A **Rec Program** is a tuple (S, T) using **global variables only**, where

- S is a statement generated by

$$S ::= \textbf{skip} \mid x := a \mid S; S \mid \textbf{if } b \textbf{ then } S \textbf{ else } S \mid m()$$

- T is a sequence of parameter-less methods $m\{S\}$ consisting of name m and body S

Example

$S_d := down()$ with $T_d := down \ \{ \ \textbf{if } x = 0 \textbf{ then } skip \textbf{ else } x := x - 1; down() \ \}$

Trace Formulas

Unary Predicates

TECHNISCHE UNIVERSITÄT DARMSTADT

Trace Formulas are a finite trace specification logic generated by

$$\Phi ::= p$$

- A **unary predicate** p semantically maps to all finite traces which satisfy p in its initial state:

$$[\![p]\!]_{\mathrm{V}} = \{s \cdot \sigma \mid s \models p \wedge \sigma \in \mathit{State}^*\}$$

Example

$\mathbf{x} = \mathbf{0}$ specifies all traces satisfying $\mathbf{x} = \mathbf{0}$ in its first state.

Trace Formulas

Binary Relations

TECHNISCHE UNIVERSITÄT DARMSTADT

Trace Formulas are a finite trace specification logic generated by

$$\Phi ::= p \mid R$$

- A **binary relation** R semantically maps to all binary traces included in R:

$$[\![R]\!]_{\mathbb{V}} = \{s \cdot s' \mid R(s, s')\}$$

Common Relations

Id (Identity) and $\mathbf{Sb}^{\mathbf{a}}_{\mathbf{x}}$ (Update) are common relations:

$$(s, s') \in \mathbf{Id} \iff s = s' \qquad (s, s') \in \mathbf{Sb}^{\mathbf{a}}_{\mathbf{x}} \iff s' = s[x \mapsto eval_s(a)]$$

Trace Formulas
Conjunctions and Disjunctions

TECHNISCHE UNIVERSITÄT DARMSTADT

Trace Formulas are a finite trace specification logic generated by

$$\Phi ::= p \mid R \mid \Phi \wedge \Phi \mid \Phi \vee \Phi$$

- **Conjunctions/Disjunctions** map to the intersection/union of the composite specified traces:

$$[\![\Phi_1 \wedge \Phi_2]\!]_{\mathbb{V}} = [\![\Phi_1]\!]_{\mathbb{V}} \cap [\![\Phi_2]\!]_{\mathbb{V}} \qquad [\![\Phi_1 \vee \Phi_2]\!]_{\mathbb{V}} = [\![\Phi_1]\!]_{\mathbb{V}} \cup [\![\Phi_2]\!]_{\mathbb{V}}$$

Example

$\mathbf{x} = \mathbf{0} \wedge \mathbf{Id}$ is a trace formula specifying all traces of length 2 with $\mathbf{x} = \mathbf{0}$ in both states.

Trace Formulas

Chops

TECHNISCHE UNIVERSITÄT DARMSTADT

Trace Formulas are a finite trace specification logic generated by

$$\Phi ::= p \mid R \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \frown \Phi$$

- The **chop** joins the traces of two trace formulas sequentially:

$$[\![\Phi_1 \frown \Phi_2]\!]_{\mathbb{V}} = \{\sigma_1 \cdot s \cdot \sigma_2 \in P(State^+) \mid \sigma_1 \cdot s \in [\![\Phi_1]\!]_{\mathbb{V}} \wedge s \cdot \sigma_2 \in [\![\Phi_2]\!]_{\mathbb{V}}\}$$

- Originates from Temporal Interval Logic

Example

Id $\frown$ **Id** is a trace formula specifying all unchanging traces of length 3.

Trace Formulas

Fixpoint Operator

TECHNISCHE UNIVERSITÄT DARMSTADT

Trace Formulas are a finite trace specification logic generated by

$$\Phi ::= p \mid R \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi^\frown\Phi \mid X \mid \mu X.\Phi$$

- The **free recursive variable** X maps according to valuation $\mathbb{V} : RVar \to P(State^+)$
- The **fixpoint operator** maps to the least fixpoint of Φ, i.e. its minimal recursive traces:

$$[\![\mu X.\Phi]\!]_\mathbb{V} = \bigcap \{\gamma \subseteq State^+ \mid [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma]} \subseteq \gamma\}$$

Example

$\mu\mathbf{X}.(\mathbf{R} \vee \mathbf{R}^\frown\mathbf{X})$ is a trace formula modeling the transitive closure of relation R.

Trace Formulas

Fixpoint Operator: Demonstration

TECHNISCHE UNIVERSITÄT DARMSTADT

$$[\![\mu X.\Phi]\!]_{\mathbb{V}} = \bigcap \{\gamma \subseteq \mathit{State}^+ \mid [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma]} \subseteq \gamma\}$$

$[\![\mu \mathbf{X}.(\mathbf{Sb}_{\mathbf{y}}^{\mathbf{y+1}} \vee \mathbf{Sb}_{\mathbf{y}}^{\mathbf{y+1}} \frown \mathbf{X})]\!]_{\mathbb{V}}$ specifies the following traces:

- $([y \mapsto 5], [y \mapsto 6])$
- $([y \mapsto 4], [y \mapsto 5], [y \mapsto 6])$
- $([y \mapsto 3], [y \mapsto 4], [y \mapsto 5], [y \mapsto 6])$
- $([y \mapsto 0], [y \mapsto 1], ..., [y \mapsto 2024], [y \mapsto 2025])$
- and many more...

! Note that $\mathbb{V}$ is irrelevant, as the formula is closed.

Unique Characteristics of Trace Formulas

TECHNISCHE UNIVERSITÄT DARMSTADT

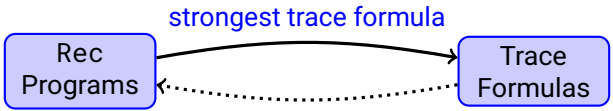
The most important characteristics of trace formulas are the support of

1. the **chop-operator** $\Phi_1 \frown \Phi_2$
   $\Rightarrow$ difference to μ-calculus
2. the **fixpoint-operator** $\mu X.\Phi$
   $\Rightarrow$ modeling of recursive traces using fixpoints

! Related logics typically go either

- without the chop, or
- without the fixpoint

Strongest Trace Formulas: Concept



Example

The **strongest trace formula** of S_{down} is $stf(S_{down})$:

$$S_{down} := \textbf{down}() \text{ with } down \{ \textbf{ if } \mathbf{x = 0} \textbf{ then skip else } \mathbf{x := x - 1}; \textbf{down}() \}$$

$$stf(S_{down}) := \mathbf{Id} \frown \mu \mathbf{X_{down}}.((\mathbf{x = 0} \wedge \mathbf{Id} \frown \mathbf{Id}) \vee (\mathbf{x \neq 0} \wedge \mathbf{Id} \frown \mathbf{Sb_x^{x-1}} \frown \mathbf{Id} \frown \mathbf{X_{down}}))$$

Strongest Trace Formulas: Main Theorem

by Gurov & Hähnle, FSCD, 2025

TECHNISCHE UNIVERSITÄT DARMSTADT

A program S **satisfies** a closed trace formula Φ iff $traces(S) \subseteq [\![\Phi]\!]$.

$traces(S) \subseteq [\![\Phi]\!]$	S **satisfies** (abstract) property Φ
iff	
$stf(S) \models \Phi$	$stf(S)$ **implies** (abstract) property Φ

! Reduction to **Trace Formula Implication Problem**.

Main Contribution: Calculus for Trace Formula Implication

Sequents

First Idea of a Sequent Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- Shape of a **sequent**:

$$\underbrace{\Gamma \vdash \Delta}$$

Γ and Δ sets of *arbitrary trace formulas*

- A sequent $\Gamma \vdash \Delta$ is called **valid** iff

$$\bigwedge \Gamma \models \bigvee \Delta$$

i.e. $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$ for any arbitrary valuation $\mathbb{V}$

! Sequents are interpreted relative to current state.

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Current state information** of antecedent Γ:

$$P_\Gamma := \{p \in \Gamma \mid p \in Pred\}$$

- The **predicate rule** infers additional information about the current state and adds it to the antecedent:

$$\text{PRED}\ \frac{\Gamma, \mathbf{q} \vdash \Delta}{\Gamma \vdash \Delta}\ \bigwedge P_\Gamma \rightarrow \mathbf{q}$$

Software Engineering Group

Calculus Rules

Predicate Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Current state information** of antecedent Γ:

$$P_\Gamma := \{p \in \Gamma \mid p \in \mathit{Pred}\}$$

- The **predicate rule** infers additional information about the current state and adds it to the antecedent:

$$\text{PRED}\ \frac{\Gamma, \mathbf{q} \vdash \Delta}{\Gamma \vdash \Delta}\ \bigwedge P_\Gamma \rightarrow \mathbf{q}$$

Example

$$\text{PRED}\ \frac{x > 0, x < 0, \mathbf{false} \vdash \Delta}{x > 0, x < 0 \vdash \Delta}\ \mathbf{x > 0 \wedge x < 0 \rightarrow false}$$

Calculus Rules

Chop Rules: Identity (Simplified Variant)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In the case of Id, the current state information P_Γ is **preserved**.

$$\texttt{CH-ID}\ \frac{P_\Gamma, Id \vdash \Psi \qquad P_\Gamma, \Phi \vdash \Psi'}{\Gamma, Id \frown \Phi \vdash \Psi \frown \Psi', \Delta}$$

- Information about Γ and Δ is lost in simplified variant.

Calculus Rules

Chop Rules: Identity (Simplified Variant)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In the case of Id, the current state information P_Γ is **preserved**.

$$\texttt{CH-ID}\ \frac{P_\Gamma, Id \vdash \Psi \qquad P_\Gamma, \Phi \vdash \Psi'}{\Gamma, Id^\frown \Phi \vdash \Psi^\frown \Psi', \Delta}$$

- Information about Γ and Δ is lost in simplified variant.

Example

$$\texttt{CH-ID}\ \frac{x = 0, Id \vdash Id \qquad \mathbf{x = 0}, Id \vdash Id}{x = 0, Id^\frown Id \vdash Id^\frown Id}$$

Software Engineering Group

Calculus Rules

Chop Rules: Update (Simplified Variant)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In case of Sb^a_x, the current state information P_Γ is modified to the **strongest postcondition**.

$$\text{CH-UPD}\ \frac{P_\Gamma, Sb^a_x \vdash \Psi \qquad spc_{x:=a}(P_\Gamma), \Phi \vdash \Psi'}{\Gamma, Sb^a_x \frown \Phi \vdash \Psi \frown \Psi', \Delta}$$

- Information about Γ and Δ is lost in simplified variant.

Calculus Rules

Chop Rules: Update (Simplified Variant)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In case of Sb^a_x, the current state information P_Γ is modified to the **strongest postcondition**.

$$\text{CH-UPD}\ \frac{P_\Gamma, Sb^a_x \vdash \Psi \qquad spc_{x:=a}(P_\Gamma), \Phi \vdash \Psi'}{\Gamma, Sb^a_x \frown \Phi \vdash \Psi \frown \Psi', \Delta}$$

- Information about Γ and Δ is lost in simplified variant.

Example

$$\text{CH-UPD}\ \frac{x = 0, Sb^1_x \vdash Sb^1_x \qquad \mathbf{x = 1}, Id \vdash Id}{x = 0, Sb^1_x \frown Id \vdash Sb^1_x \frown Id}$$

Calculus Rules

Unfolding Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

- We can **unfold** fixpoint operators $\mu X.\Phi$ into the semantically equivalent formula $\Phi[\mu X.\Phi/X]$.
- **Unfolding rules** can unfold fixpoint formulas in the antecedent and succedent:

$$\text{UNFL}\ \frac{\Gamma, \Phi[\mu X.\Phi/X] \vdash \Delta}{\Gamma, \mu X.\Phi \vdash \Delta} \qquad \text{UNFR}\ \frac{\Gamma \vdash \Psi[\mu X.\Psi/X], \Delta}{\Gamma \vdash \mu X.\Psi, \Delta}$$

Calculus Rules

Unfolding Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

- We can **unfold** fixpoint operators $\mu X.\Phi$ into the semantically equivalent formula $\Phi[\mu X.\Phi/X]$.
- **Unfolding rules** can unfold fixpoint formulas in the antecedent and succedent:

$$\text{UNFL}\ \frac{\Gamma, \Phi[\mu X.\Phi/X] \vdash \Delta}{\Gamma, \mu X.\Phi \vdash \Delta} \qquad \text{UNFR}\ \frac{\Gamma \vdash \Psi[\mu X.\Psi/X], \Delta}{\Gamma \vdash \mu X.\Psi, \Delta}$$

Example

$$\text{UNFR}\ \frac{\Gamma \vdash R \vee R^\frown \mu X.(R \vee R^\frown X)}{\Gamma \vdash \mu X.(R \vee R^\frown X)}$$

Software Engineering Group

The Problem with Unfoldings

TECHNISCHE UNIVERSITÄT DARMSTADT

Let us assume a sequent of the shape $\mu X_1.\,\Phi \vdash \mu X_2.\,\Psi$.

- Unfolding strategy of $\mu X_1.\,\Phi$ **not always applicable**, as the amount of needed unfolding steps can be very large or may even be unbounded.

Example (down cont.)

$$\mathbf{Sb_x^{10}} \frown Id \frown \mu X_{down}.((x = 0 \wedge Id \frown Id) \vee (x \neq 0 \wedge Id \frown Sb_x^{x-1} \frown Id \frown X_{down})) \vdash \Delta$$

$\Rightarrow$ Unfolding strategy *possible*

$$Id \frown \mu X_{down}.((x = 0 \wedge Id \frown Id) \vee (x \neq 0 \wedge Id \frown Sb_x^{x-1} \frown Id \frown X_{down})) \vdash \Delta$$

$\Rightarrow$ Unfolding strategy *impossible*

Sequents

Adapted Sequent Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Objective**: Show $\mu X_1.\Phi \vdash \mu X_2.\Psi$
  $\Longrightarrow$ **Solution**: Create a **connection between the recursive variables** of both fixpoints.

Sequents

Adapted Sequent Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Objective**: Show $\mu X_1.\Phi \vdash \mu X_2.\Psi$
  $\Longrightarrow$ **Solution**: Create a **connection between the recursive variables** of both fixpoints.
- A **recursive variable assumption set** ξ is a set of tuples of the shape $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$.
  It is called **valid** in $\mathbb{V}$ iff for all its tuples $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$, it holds that

$$[\![\mathbf{X_1} \wedge \mathbf{Inv}]\!]_{\mathbb{V}} \subseteq [\![\mathbf{X_2}]\!]_{\mathbb{V}}$$

Sequents

Adapted Sequent Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Objective**: Show $\mu X_1.\Phi \vdash \mu X_2.\Psi$
  $\Longrightarrow$ **Solution**: Create a **connection between the recursive variables** of both fixpoints.
- A **recursive variable assumption set** ξ is a set of tuples of the shape $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$.
  It is called **valid** in $\mathbb{V}$ iff for all its tuples $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$, it holds that

$$[\![\mathbf{X_1} \wedge \mathbf{Inv}]\!]_{\mathbb{V}} \subseteq [\![\mathbf{X_2}]\!]_{\mathbb{V}}$$

- Adapted shape of a **sequent**:

$$\xi \diamond \Gamma \vdash \Delta$$

Software Engineering Group

Sequents

Adapted Sequent Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Objective**: Show $\mu X_1.\Phi \vdash \mu X_2.\Psi$
  $\Longrightarrow$ **Solution**: Create a **connection between the recursive variables** of both fixpoints.
- A **recursive variable assumption set** ξ is a set of tuples of the shape $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$.
  It is called **valid** in $\mathbb{V}$ iff for all its tuples $(\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2})$, it holds that

$$[\![\mathbf{X_1} \wedge \mathbf{Inv}]\!]_{\mathbb{V}} \subseteq [\![\mathbf{X_2}]\!]_{\mathbb{V}}$$

- Adapted shape of a **sequent**:

$$\xi \diamond \Gamma \vdash \Delta$$

- A sequent $\xi \diamond \Gamma \vdash \Delta$ is called **valid** iff

$$[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}} \text{ for any arbitrary valuation } \mathbb{V} \text{ that is valid for } \xi$$

Calculus Rules

Fixpoint Induction Rule

TECHNISCHE UNIVERSITÄT DARMSTADT

- The **fixpoint induction rule** first establishes that the **invariant** holds in current state P_Γ and then shows that the **invariant** is preserved to all recursive calls:

$$\texttt{FPI}\ \frac{P_\Gamma \vdash \mathbf{Inv} \qquad \xi, (\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2}) \diamond \mathbf{Inv}, \Phi \vdash \Psi}{\xi \diamond \Gamma, \mu X_{1.}\, \Phi \vdash \mu X_{2.}\, \Psi, \Delta}$$

Software Engineering Group

Calculus Rules

Fixpoint Induction Rule

TECHNISCHE UNIVERSITÄT DARMSTADT

- The **fixpoint induction rule** first establishes that the **invariant** holds in current state P_Γ and then shows that the **invariant** is preserved to all recursive calls:

$$\texttt{FPI}\ \frac{P_\Gamma \vdash \mathbf{Inv} \qquad \xi, (\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2}) \diamond \mathbf{Inv}, \Phi \vdash \Psi}{\xi \diamond \Gamma, \mu X_1.\Phi \vdash \mu X_2.\Psi, \Delta}$$

Example

$$\texttt{FPI}\ \frac{\mathbf{x = 10} \vdash \mathbf{x > 0} \qquad (\mathbf{X_1}|_{\mathbf{x>0}}, \mathbf{X_2}) \diamond \mathbf{x > 0}, Sb_x^{x+1} \vee Sb_x^{x+1} \frown \mathbf{X_1} \vdash R_{pos} \vee R_{pos} \frown \mathbf{X_2}}{\mathbf{x = 10}, \mu X_1.(Sb_x^{x+1} \vee Sb_x^{x+1} \frown \mathbf{X_1}) \vdash \mu X_2.(R_{pos} \vee R_{pos} \frown \mathbf{X_2})}$$

Software Engineering Group

Calculus Rules

Recursive Variable Rule

- We then infer the trace inclusion between recursive variables based on ξ:

$$\text{RVAR}\ \frac{P_\Gamma \vdash \mathbf{Inv}}{\xi, (\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2}) \diamond \Gamma, \mathbf{X_1} \vdash \mathbf{X_2}, \Delta}$$

Calculus Rules

Recursive Variable Rule

TECHNISCHE UNIVERSITÄT DARMSTADT

- We then infer the trace inclusion between recursive variables based on ξ:

$$\text{RVAR}\ \frac{P_\Gamma \vdash \mathbf{Inv}}{\xi, (\mathbf{X_1}|_{\mathbf{Inv}}, \mathbf{X_2}) \diamond \Gamma, \mathbf{X_1} \vdash \mathbf{X_2}, \Delta}$$

Example

$$\text{CH-UPD}\ \frac{\begin{array}{c}\vdots\\ \mathbf{x} > \mathbf{0}, Sb_x^{x+1} \vdash R_{pos}\end{array} \qquad \text{RVAR}\ \frac{\mathbf{x} > \mathbf{1} \vdash \mathbf{x} > \mathbf{0}}{(\mathbf{X_1}|_{\mathbf{x>0}}, \mathbf{X_2}) \diamond \mathbf{x} > \mathbf{1}, \mathbf{X_1} \vdash \mathbf{X_2}}}{(\mathbf{X_1}|_{\mathbf{x>0}}, \mathbf{X_2}) \diamond \mathbf{x} > \mathbf{0}, Sb_x^{x+1} \frown \mathbf{X_1} \vdash R_{pos} \frown \mathbf{X_2}}$$

Software Engineering Group

Incompleteness of the Calculus

Method Contracts

So far, we can only handle **tail recursive** occurrences of recursive variables or fixpoint operations. We lack rules otherwise:

$$\mathbf{x = 10}, X_1 \frown \Phi \vdash X_2 \frown \Psi$$

Incompleteness of the Calculus

Method Contracts

TECHNISCHE UNIVERSITÄT DARMSTADT

So far, we can only handle **tail recursive** occurrences of recursive variables or fixpoint operations. We lack rules otherwise:

$$\mathbf{x = 10}, X_1 \frown \Phi \vdash X_2 \frown \Psi$$

Solution: Method Contracts

- Employment of a rule that **proves** and a rule that **applies** a method contract.
- A method contract (**pre**, **post**) is *valid* for Φ in $\mathbb{V}$ iff

$$[\![\bigwedge v^i_{old} = v^i \wedge \mathbf{pre} \wedge \Phi \frown \mathit{true}]\!]_{\mathbb{V}} \subseteq [\![\Phi \frown \mathbf{post}]\!]_{\mathbb{V}}$$

Incompleteness of the Calculus

Synchronization

TECHNISCHE UNIVERSITÄT DARMSTADT

There are no rules that can perform fixpoint induction on a **mismatched shape** of recursive variables inside fixpoint formulas:

$$\mu X_1.(R \vee R \frown X_1) \vdash \mu X_2.(R \vee X_2 \frown R)$$

Incompleteness of the Calculus

Synchronization

TECHNISCHE UNIVERSITÄT DARMSTADT

There are no rules that can perform fixpoint induction on a **mismatched shape** of recursive variables inside fixpoint formulas:

$$\mu X_1.(R \vee R^\frown X_1) \vdash \mu X_2.(R \vee X_2{}^\frown R)$$

Solution: Synchronization

- Employment of a rule for **synchronization** between recursive variables:

$$\mu X_2.(R \vee X_2{}^\frown R) \rightsquigarrow \mu X_2.(R \vee R^\frown X_2)$$

Chop Formulas

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Chop formulas** are a subclass of trace formulas for fixed relation R and fixed recursive variable X generated by the following grammar:

$$\Phi ::= \Psi \mid \Psi \vee \Phi \text{ with } \Psi ::= \mathbf{R} \mid \mathbf{X} \mid \Psi \frown \Psi$$

- Each chop formula can be mapped onto a **context-free grammar**.

Example

$\mathbf{Id} \frown \mathbf{Id} \vee \mathbf{Id} \frown \mathbf{X} \frown \mathbf{Id}$ is a chop formula with CFG production rule $\mathbf{X} \to \mathbf{Id}\ \mathbf{Id} \mid \mathbf{Id}\ \mathbf{X}\ \mathbf{Id}$

Extension of the Calculus with μ-Formula Synchronization

TECHNISCHE UNIVERSITÄT DARMSTADT

- The language specified by the context-free grammar must be **regular** (*Parikh's lemma*).
- Fixpoint (chop) formulas can be **reshaped** depending on their language:

$$\text{SYNC}\ \frac{\xi \diamond \Gamma \vdash \mu X.\Psi', \Delta}{\xi \diamond \Gamma \vdash \mu X.\Psi, \Delta}\ L(Gr(\Psi')) \subseteq L(Gr(\Psi))$$

Example

$$\text{SYNC}\ \frac{\mu X_1.(R \vee R^\frown X_1) \vdash \mu X_2.(R \vee R^\frown X_2)}{\mu X_1.(R \vee R^\frown X_1) \vdash \mu X_2.(R \vee X_2{}^\frown R)}$$

Software Engineering Group

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Trace formulas** can be used to
  - specify programs **precisely**
  - specify **abstract** trace properties
- The **sequent calculus** for trace formula implication proves that $stf(S) \models \Phi$ with
  - **fixpoint inductions** for symbolic derivations (formalized in **Isabelle**)
  - **method contracts** for state updates over calls
  - **synchronization** for synchronizing recursive variables for fixpoint inductions
- **Soundness** of the calculus
- Future Work: Resolve **Incompleteness** of the calculus

Appendix: Isabelle Model

TECHNISCHE UNIVERSITÄT DARMSTADT

- Formalization of the calculus achieved in theorem prover `Isabelle/HOL`.
- Properties can be inferred by **manual** rule application or **semi-automatic** rule application.
  $\Rightarrow$ Invariants for fixpoint formulas need to be **manually** provided.

```
lemma "[stf S_down] ⊢ [μ ''dec''. ((Rel R_x_dec) ⊔ Rel R_x_dec ⌢ RVar ''dec'')]"
  unfolding S_down_def
  apply simp
  apply tfi_auto
  apply lenr_prep_fpi
  apply (rule FPI; auto)
  apply tfi_auto .
```

Figure: Assume *R_x_dec* is a relation specifying that *x* **does not** increase.

Appendix: Calculus Rules

Relation Rules: Axiom

TECHNISCHE UNIVERSITÄT DARMSTADT

- Let relation R **restricted** on P_Γ be

$$\mathbf{R}|_{P_\Gamma} := \{s \cdot s' \mid \mathbf{R}(s, s') \wedge s \models \bigwedge P_\Gamma\}$$

- The **relation rule axiom** determines if the relation of the antecedent is included in the relation of the succedent:

$$\text{REL}\ \frac{}{\Gamma, \mathbf{R} \vdash \mathbf{R}', \Delta}\ \mathbf{R}|_{P_\Gamma} \subseteq \mathbf{R}'$$

Example

$$\text{REL}\ \frac{}{\mathbf{x} > \mathbf{0}, \mathbf{Sb}_{\mathbf{x}}^{\mathbf{x+1}} \vdash \mathbf{R_{pos}}}\ \mathbf{Sb}_{\mathbf{x}}^{\mathbf{x+1}}|_{\mathbf{x>0}} \subseteq \mathbf{R_{pos}}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

- Let us define **repetitions** of a formula containing recursive variables:

$$repeat_0(\Phi) := \Phi \text{ and } repeat_i(\Phi) := \Phi[repeat_{i-1}(\Phi)/X]) \text{ for } i \geq 1,$$

- We can use **lengthening rules** to *internally lengthen* the fixpoint formulas:

$$\text{LENL } \frac{\Gamma, \mu X.repeat_i(\Phi) \vdash \Delta}{\Gamma, \mu X.\Phi \vdash \Delta}\ i \geq 1 \qquad \text{LENR } \frac{\Gamma \vdash \mu X.repeat_i(\Psi), \Delta}{\Gamma \vdash \mu X.\Psi, \Delta}\ i \geq 1$$

Example

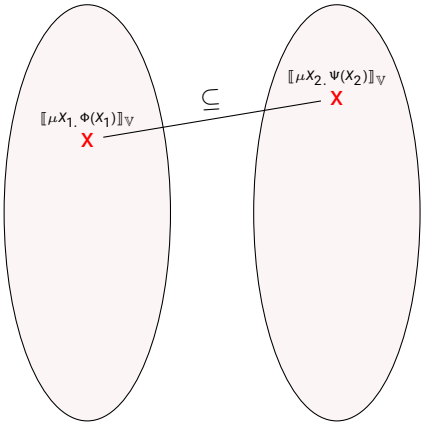
$$\text{LENR } \frac{\Gamma \vdash \mu X.(R \vee R^\frown(R \vee R^\frown X))}{\Gamma \vdash \mu X.(R \vee R^\frown X)}\ 1 \geq 1$$

Appendix: Fixpoint Induction

Proof Objective

- We aim to **inductively** deduce the $\subseteq$-inclusion of both least fixpoints.
- Based on **powerset lattice of finite traces** $P(State^+)$.

How to infer the proof objective?



Appendix: Fixpoint Induction

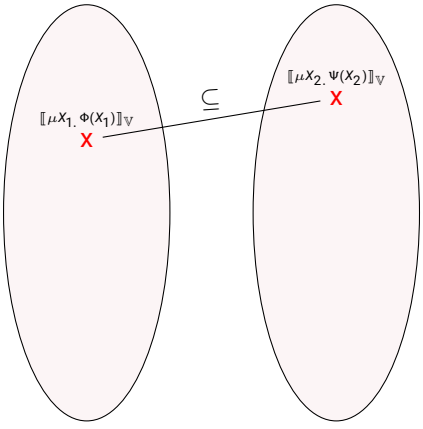
Induction Principle

TECHNISCHE UNIVERSITÄT DARMSTADT

Assumption:

For any set of finite traces $\gamma_1, \gamma_2 \in P(State^+)$ with $\gamma_1 \subseteq \gamma_2$, it holds that

$$[\![\Phi(X_1)]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi(X_2)]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$



Software Engineering Group

Appendix: Fixpoint Induction

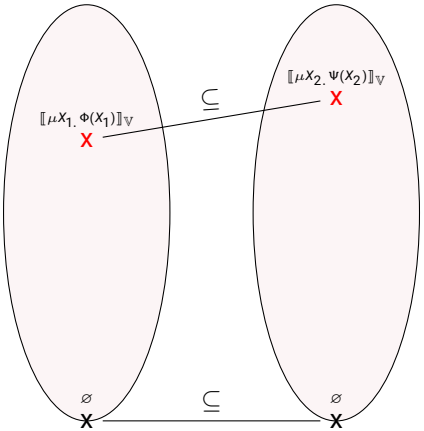
Induction Principle

TECHNISCHE UNIVERSITÄT DARMSTADT

Assumption:

For any set of finite traces $\gamma_1, \gamma_2 \in P(State^+)$ with $\gamma_1 \subseteq \gamma_2$, it holds that

$$[\![\Phi(X_1)]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi(X_2)]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$



Software Engineering Group

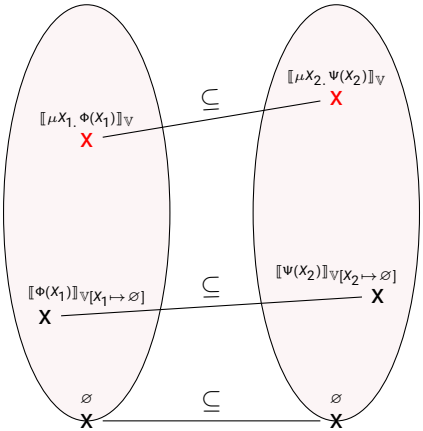
Appendix: Fixpoint Induction

Induction Principle

Assumption:

For any set of finite traces $\gamma_1, \gamma_2 \in P(State^+)$ with $\gamma_1 \subseteq \gamma_2$, it holds that

$$[\![\Phi(X_1)]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi(X_2)]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$



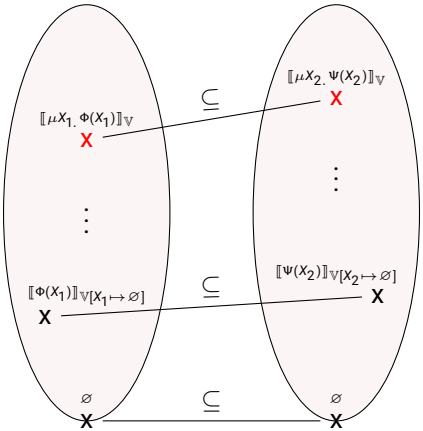
Appendix: Fixpoint Induction

Induction Principle

Assumption:

For any set of finite traces $\gamma_1, \gamma_2 \in P(State^+)$ with $\gamma_1 \subseteq \gamma_2$, it holds that

$$[\![\Phi(X_1)]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi(X_2)]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$



Appendix: Fixpoint Induction

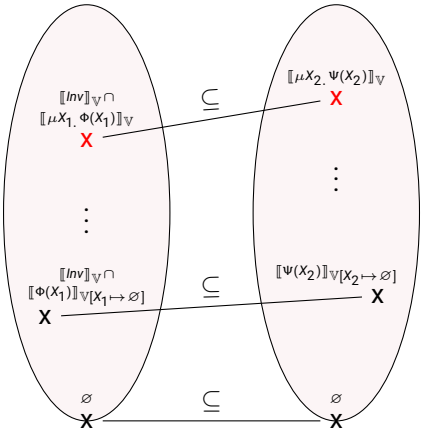
Induction Principle with Invariants

TECHNISCHE UNIVERSITÄT DARMSTADT

Assumption:

For any set of finite traces $\gamma_1, \gamma_2 \in P(State^+)$ with $[\![Inv]\!]_{\mathbb{V}} \cap \gamma_1 \subseteq \gamma_2$, it holds that

$$[\![Inv]\!]_{\mathbb{V}} \cap [\![\Phi(X_1)]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi(X_2)]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$



Software Engineering Group

Appendix: Calculus Rules

Method Contract Rules

$$\texttt{MC}\ \frac{v^i_{old} \in \mathit{fresh}(\mathit{Var}) \qquad \mathbb{C}' = \mathbb{C}[m \mapsto (\mathit{pre}, \mathit{post})] \qquad \xi \diamond \langle \mathit{pre}(\Phi) \rangle \vdash_{\mathbb{C}'} \langle \mathit{post}(\Phi) \rangle \qquad \xi \diamond \Gamma, \mu X_m.\Phi \vdash_{\mathbb{C}'} \Delta}{\xi \diamond \Gamma, \mu X_m.\Phi \vdash_{\mathbb{C}} \Delta}$$

$$\texttt{CH-RVAR}\ \frac{\mathbb{C}(m) = (\mathit{pre}, \mathit{post}) \qquad P_\Gamma \vdash_{\mathbb{C}} p \wedge \mathit{pre} \qquad \xi \diamond P_\Gamma[v^i_{old}/v^i], \mathit{post}, \Phi \vdash_{\mathbb{C}} \Psi}{\xi, (X_m|_p, X) \diamond \Gamma, X_m \frown \Phi \vdash_{\mathbb{C}} X \frown \Psi, \Delta}$$

$$\texttt{CH-FPI}\ \frac{\mathbb{C}(m) = (\mathit{pre}, \mathit{post}) \qquad P_\Gamma \vdash_{\mathbb{C}} I \wedge \mathit{pre} \qquad \xi, (X_m|_I, X) \diamond I, \Phi_1 \vdash_{\mathbb{C}} \Psi_1 \qquad \xi \diamond P_\Gamma[v^i_{old}/v^i], \mathit{post}, \Phi_2 \vdash_{\mathbb{C}} \Psi_2}{\xi \diamond \Gamma, (\mu X_m.\Phi_1) \frown \Phi_2 \vdash_{\mathbb{C}} (\mu X.\Psi_1) \frown \Psi_2, \Delta}$$