Gradual Verification

with Implicit Dynamic Frames

Master Thesis of

Johannes Bader
Karlsruhe Institute of Technology
Karlsruhe, Germany

Advised by

Assoc. Prof. Jonathan Aldrich
Carnegie Mellon University
Pittsburgh, USA

Assoc. Prof. Éric Tanter
University of Chile
Santiago, Chile

Gradual Verification

(with Implicit Dynamic Frames)

```
int getFour(int i)
    requires ?; // haven't figured that one out, yet
    ensures  result = 4;
{
    i := i + 1;
    return i;
}
```

Motivation

- Program verification (against some specification)
- Two flavors: static & dynamic

```
// spec: callable only if (this.balance >= amount)
void withdrawCoins(int amount)
{
   // business logic
   this.balance -= amount;
}
```

Dynamic Verification

- runtime checks
- testing techniques
- guarantee compliance **at runtime**

```
void withdrawCoins(int amount)
{
   assert this.balance >= amount;
   // business logic
   this.balance -= amount;
}
```

Dynamic Verification - Drawbacks

- runtime checks — runtime overhead
- testing techniques — additional efforts
- guarantee compliance **at runtime** — pot. late detection

```
void withdrawCoins(int amount)
    requires this.balance >= amount;
{
    // business logic
    this.balance -= amount;
}
```

Static Verification

- declarative
- formal logic
- guarantee compliance **in advance**

```
void withdrawCoins(int amount)
    requires this.balance >= amount;
{
    // business logic
    this.balance -= amount;
}
```

Static Verification - Drawbacks

- declarative
- formal logic
- guarantee compliance **in advance**

limited syntax
decidability

annotation pressure

```
void withdrawCoins(int amount)
    requires this.balance >= amount;
    ensures  this.balance = old(this.balance) – amount;
{
    // business logic
    this.balance -= amount;
}
```

Solution? Static + Dynamic

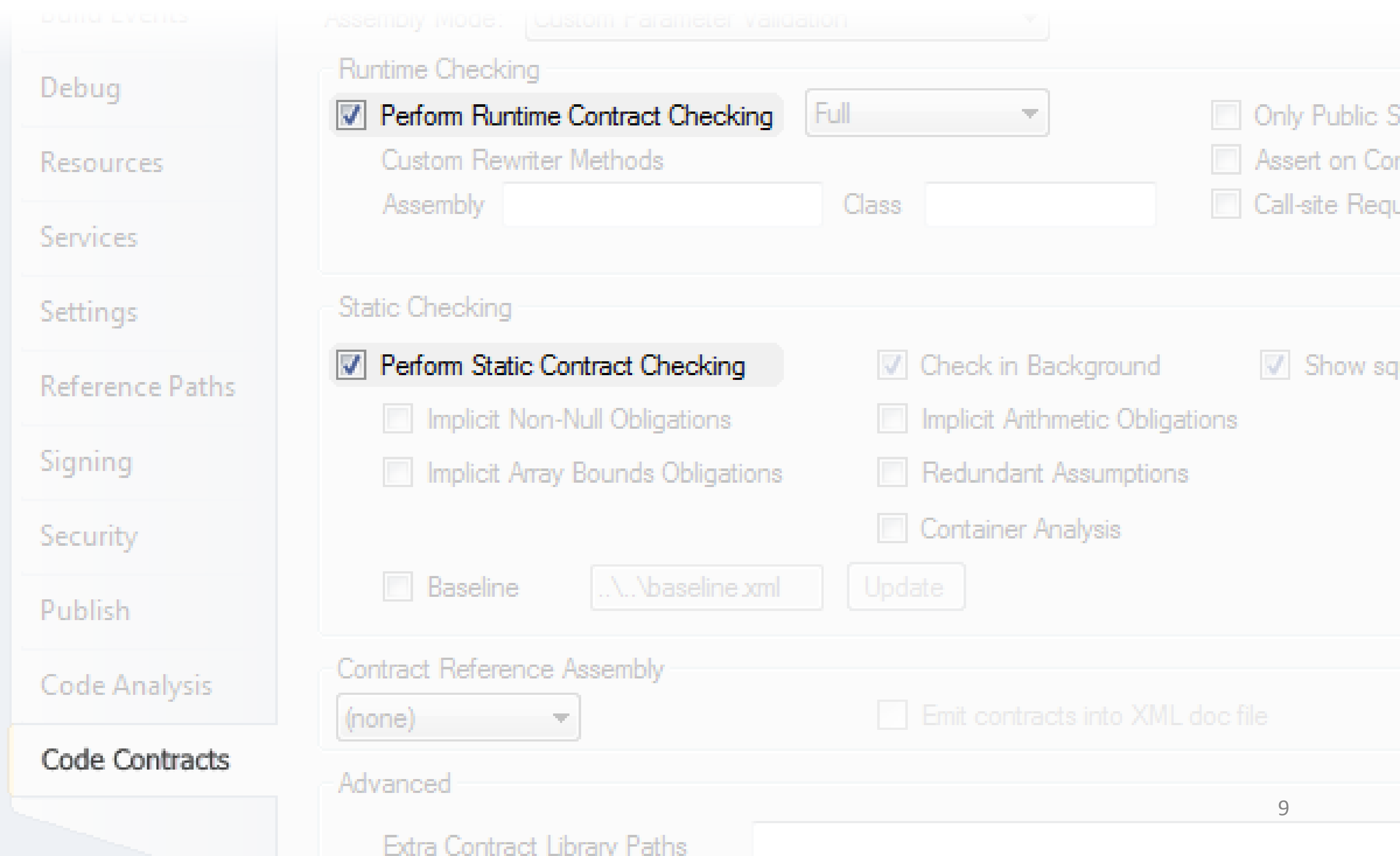
Means:

- “relaxed” static verification (warnings on failure)
- turn contracts into runtime assertions

Notable implementations:

- Java with JML annotations
  - “ESC/Java” for static verification
  - “JML4c” for dynamic verification
- Code Contracts for .NET (by RiSE, MSR)

Solution? Static + Dynamic



Solution! Static ⊕ Dynamic

"Static Typing Where Possible,
Dynamic Typing When Needed" (Erik Meijer)

```
void withdrawCoins(int amount)
    requires this.balance >= amount;
{
    …
}
…
acc.balance = 100;
acc.withdrawCoins(50); // can prove acc.balance >= 50
acc.withdrawCoins(30); // can't prove acc.balance >= 30
acc.withdrawCoins(30); // can't prove acc.balance >= 30
```

Approach

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \models \phi$

Soundness

Gradualization

Syntax

$\widetilde{s} \in \widetilde{\textsc{Stmt}}$

$\widetilde{\phi} \in \widetilde{\textsc{Formula}}$

Program State

$\widetilde{\pi} \in \widetilde{\textsc{ProgramState}}$

Semantics

Static $\widetilde{\vdash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}\}$

Dynamic $\widetilde{\pi} \mathrel{\widetilde{\longrightarrow}} \widetilde{\pi}$

Formula $\widetilde{\pi} \mathrel{\widetilde{\models}} \widetilde{\phi}$

Soundness

Statically Verified Language

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \vDash \phi$

Soundness

$$e \quad ::= \quad x \ \mid\ n \ \mid\ e_1 + e_2$$

$$s \quad ::= \quad x \ \texttt{:=}\ e \ \mid\ \texttt{assert}\ \phi \ \mid\ s_1\texttt{;}\ s_2$$

$$\phi \quad ::= \quad \texttt{true} \ \mid\ (e_1\ \texttt{=}\ e_2) \ \mid\ \phi_1 \wedge \phi_2$$

$$= (\textsc{Var} \rightharpoonup \mathbb{N}_0) \times \textsc{Stmt}$$

Statically Verified Language

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \vDash \phi$

Soundness

$$\frac{x\ \texttt{:=}\ e}{\vdash \{\phi[e/x]\}\ x\ \texttt{:=}\ e\ \{\phi\}}\ \textsc{HAssign}$$

$$\frac{\phi \Rightarrow \phi_a}{\vdash \{\phi\}\ \texttt{assert}\ \phi_a\ \{\phi\}}\ \textsc{HAssert}$$

$$\frac{\begin{array}{c}\phi_{q1} \Rightarrow \phi_{q2} \\ \vdash \{\phi_p\}\ s_1\ \{\phi_{q1}\} \qquad \vdash \{\phi_{q2}\}\ s_2\ \{\phi_r\}\end{array}}{\vdash \{\phi_p\}\ s_1\texttt{;}\ \ s_2\ \{\phi_r\}}\ \textsc{HSeq}$$

Statically Verified Language

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \models \phi$

Soundness

$$\frac{\mathcal{N}_\sigma(e) = n}{\langle \sigma, x\ \texttt{:=}\ e\texttt{;}\ \ s\rangle \longrightarrow \langle \sigma[x \mapsto n], s\rangle}\ \textsc{SsAssign}$$

$$\frac{\langle \sigma, \texttt{assert}\ \ \phi_a\texttt{;}\ \ s\rangle \models \phi_a}{\langle \sigma, \texttt{assert}\ \ \phi_a\texttt{;}\ \ s\rangle \longrightarrow \langle \sigma, s\rangle}\ \textsc{SsAssert}$$

Statically Verified Language

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \vDash \phi$

Soundness

$$\frac{\vdash \{\phi\}\ s\ \{\phi'\}}{\vDash \{\phi\}\ s\ \{\phi'\}}\ \textsc{Soundness}$$

$$\vDash \{\phi\}\ s\ \{\phi'\}$$

$$\overset{\text{def}}{\iff}$$

$$\forall \pi, \pi'.\ \pi \xrightarrow{s} \pi' \land \pi \vDash \phi \implies \pi' \vDash \phi'$$

Approach

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \vDash \phi$

Soundness

Gradualization

Syntax

$\widetilde{s} \in \widetilde{\textsc{Stmt}}$

$\widetilde{\phi} \in \widetilde{\textsc{Formula}}$

Program State

$\widetilde{\pi} \in \widetilde{\textsc{ProgramState}}$

Semantics

Static $\widetilde{\vdash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}\}$

Dynamic $\widetilde{\pi} \ \widetilde{\longrightarrow}\ \widetilde{\pi}$

Formula $\widetilde{\pi}\ \widetilde{\vDash}\ \widetilde{\phi}$

Soundness

Gradualization – Goal 1/3

Introduction of wildcard formula ?

- placeholder for arbitrary (satisfiable) formula
- enables Hoare deduction despite incomplete information
- enables gradual annotation of programs (? as default)

Formula Precision

$$\widetilde{\phi_1} \sqsubseteq \widetilde{\phi_2} \overset{\text{def}}{\iff} \gamma(\widetilde{\phi_1}) \subseteq \gamma(\widetilde{\phi_2})$$

Gradualization – Goal 2/3

Compatibility with static language

- don't reject source code that was accepted before — static
- observable behavior is not changed — dynamic

$$\textsc{Formula} \subseteq \widetilde{\textsc{Formula}} \qquad \textsc{Stmt} \subseteq \widetilde{\textsc{Stmt}}$$

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \widetilde{\vdash} \{\phi\}\ s\ \{\phi'\}$$

$$\pi \xrightarrow{s} \pi' \implies \exists \pi''.\ \pi \overset{s}{\widetilde{\longrightarrow}} \pi'' \wedge (\forall \phi.\ \pi' \vDash \phi \implies \pi'' \vDash \phi)$$

Gradualization – Goal 3/3

Gradual guarantee (Siek et al.), adapted

Reducing precision will not

- introduce verification failure — static
- change observable behavior — dynamic

$$\text{Given} \quad \widetilde{\phi_1} \sqsubseteq \widetilde{\phi_2} \wedge \widetilde{\phi_1'} \sqsubseteq \widetilde{\phi_2'} \wedge \widetilde{s_1} \sqsubseteq \widetilde{s_2} \wedge \widetilde{\pi_1} \sqsubseteq \widetilde{\pi_2}$$

$$\widetilde{\vdash} \{\widetilde{\phi_1}\} \; \widetilde{s_1} \; \{\widetilde{\phi_1'}\} \implies \widetilde{\vdash} \{\widetilde{\phi_2}\} \; \widetilde{s_2} \; \{\widetilde{\phi_2'}\}$$

$$\widetilde{\pi_1} \xrightarrow{\widetilde{s_1}} \widetilde{\pi_1'} \implies \exists \widetilde{\pi_2'}.\; \widetilde{\pi_2} \xrightarrow{\widetilde{s_2}} \widetilde{\pi_2'} \wedge \widetilde{\pi_1'} \sqsubseteq \widetilde{\pi_2'}$$

Gradual Predicate Lifting

Introduction

$$\forall \phi_1, \phi_2 \in \textsc{Formula}.\ P(\phi_1, \phi_2) \implies \widetilde{P}(\phi_1, \phi_2)$$

Monotonicity

$$\forall \widetilde{\phi_1}, \widetilde{\phi_2}, \widetilde{\phi_1'}, \widetilde{\phi_2'} \in \widetilde{\textsc{Formula}}.\ \widetilde{\phi_1} \sqsubseteq \widetilde{\phi_1'} \wedge \widetilde{\phi_2} \sqsubseteq \widetilde{\phi_2'} \wedge \widetilde{P}(\widetilde{\phi_1}, \widetilde{\phi_2}) \implies \widetilde{P}(\widetilde{\phi_1'}, \widetilde{\phi_2'})$$

$$P(\phi_1, \phi_a, \phi_2) \stackrel{\text{def}}{=} \phi_1 = \phi_2 \wedge \phi_1 \Rightarrow \phi_a$$

$$\widetilde{P}(\widetilde{\phi_1}, \widetilde{\phi_a}, \widetilde{\phi_2}) \stackrel{\text{def}}{=} \widetilde{\phi_1} \approx \widetilde{\phi_2} \wedge \widetilde{\phi_1} \mathrel{\widetilde{\Rightarrow}} \widetilde{\phi_a}$$

$$\widetilde{\phi_1} \approx \widetilde{\phi_2} \stackrel{\text{def}}{=} \widetilde{\phi_1} = \widetilde{\phi_2} \vee \widetilde{\phi_1} = \textsf{?} \vee \widetilde{\phi_2} = \textsf{?}$$

$$\widetilde{\phi_1} \mathrel{\widetilde{\Rightarrow}} \widetilde{\phi_2} \stackrel{\text{def}}{=} \widetilde{\phi_1} \Rightarrow \widetilde{\phi_2} \vee \widetilde{\phi_1} = \textsf{?} \vee \widetilde{\phi_2} = \textsf{?}$$

$$\frac{\phi \Rightarrow \phi_a}{\vdash \{\phi\}\ \texttt{assert}\ \phi_a\ \{\phi\}}\ \textsc{HAssert}$$

$$\frac{}{\widetilde{\vdash} \{\widetilde{\phi_1}\}\ \texttt{assert}\ \widetilde{\phi_a}\ \{\widetilde{\phi_2}\}}\ \widetilde{\textsc{H}}\textsc{Assert}$$

Gradual Predicate Lifting

Introduction

$$\forall \phi_1, \phi_2 \in \textsc{Formula}.\ P(\phi_1, \phi_2) \implies \widetilde{P}(\phi_1, \phi_2)$$

Monotonicity

$$\forall \widetilde{\phi_1}, \widetilde{\phi_2}, \widetilde{\phi_1'}, \widetilde{\phi_2'} \in \widetilde{\textsc{Formula}}.\ \widetilde{\phi_1} \sqsubseteq \widetilde{\phi_1'} \land \widetilde{\phi_2} \sqsubseteq \widetilde{\phi_2'} \land \widetilde{P}(\widetilde{\phi_1}, \widetilde{\phi_2}) \implies \widetilde{P}(\widetilde{\phi_1'}, \widetilde{\phi_2'})$$

(Optimality)

$\widetilde{P}$ is smallest predicate closed under above rules

⇔

$$\widetilde{P}(\widetilde{\phi_1}, \widetilde{\phi_2}) \iff \exists \phi_1 \in \gamma(\widetilde{\phi_1}), \phi_2 \in \gamma(\widetilde{\phi_2}).\ P(\phi_1, \phi_2)$$

Gradual Predicate Lifting

$$
\begin{aligned}
& \widetilde{\pi} \mathrel{\widetilde{\vDash}} \widetilde{\phi} \\
\iff & \exists \pi \in \gamma(\widetilde{\pi}), \phi \in \gamma(\widetilde{\phi}).\ \pi \vDash \phi \\
\iff & \exists \phi \in \gamma(\widetilde{\phi}).\ \widetilde{\pi} \vDash \phi \\
\iff & \widetilde{\pi} \vDash \widetilde{\phi} \lor \widetilde{\phi} = \texttt{?}
\end{aligned}
$$

$$
\frac{\widetilde{\pi} \vDash \phi}{\widetilde{\pi} \mathrel{\widetilde{\vDash}} \phi}\ \textsc{EvalPhiStatic} \qquad\qquad \frac{}{\widetilde{\pi} \mathrel{\widetilde{\vDash}} \texttt{?}}\ \textsc{EvalPhiStatic}
$$

$$
\widetilde{P}(\widetilde{\phi_1}, \widetilde{\phi_2}) \iff \exists \phi_1 \in \gamma(\widetilde{\phi_1}), \phi_2 \in \gamma(\widetilde{\phi_2}).\ P(\phi_1, \phi_2)
$$

Gradual Function Lifting

Introduction

$$\forall \phi \in \textsc{Formula}.\ f(\phi) \sqsubseteq \widetilde{f}(\phi)$$

Monotonicity

$$\forall \widetilde{\phi_1}, \widetilde{\phi_2} \in \widetilde{\textsc{Formula}}.\ \widetilde{\phi_1} \sqsubseteq \widetilde{\phi_2} \implies \widetilde{f}(\widetilde{\phi_1}) \sqsubseteq \widetilde{f}(\widetilde{\phi_2})$$

(Optimality)

$\widetilde{f}$ has most precise return values among all liftings

$\Leftrightarrow$ $\widetilde{f}(\widetilde{\phi}) = \alpha(\overline{f}(\gamma(\widetilde{\phi})))$ where $\langle \alpha, \gamma \rangle$ is $\{\ \overline{f}\ \}$-partial Galois connection

Gradual Partial Function Lifting

Introduction

$$\forall \phi \in \textsc{Formula} \cap \mathsf{dom}(f).\ f(\phi) \sqsubseteq \widetilde{f}(\phi)$$

Monotonicity

$$\forall \widetilde{\phi_1}, \widetilde{\phi_2} \in \widetilde{\textsc{Formula}}.\ \widetilde{\phi_1} \sqsubseteq \widetilde{\phi_2} \land \widetilde{\phi_1} \in \mathsf{dom}(\widetilde{f}) \implies \widetilde{f}(\widetilde{\phi_1}) \sqsubseteq \widetilde{f}(\widetilde{\phi_2})$$

(Optimality)

$\widetilde{f}$ has smallest domain and most precise return values among all liftings

$\Leftrightarrow$ $\quad \widetilde{f}(\widetilde{\phi}) = \alpha(\overline{f}(\gamma(\widetilde{\phi})))$ $\quad$ where $\langle \alpha, \gamma \rangle$ is $\{\ \overline{f}\ \}$-partial Galois connection

Gradual Partial Function Lifting

$$\frac{\langle \sigma, \texttt{assert}\ \ \phi_a\,;\ \ s\rangle \vDash \phi_a}{\langle \sigma, \texttt{assert}\ \ \phi_a\,;\ \ s\rangle \longrightarrow \langle \sigma, s\rangle}\ \textsc{SsAssert}$$

$$\frac{\langle \sigma, \texttt{assert}\ \ \widetilde{\phi_a}\,;\ \ s\rangle \mathrel{\widetilde{\vDash}} \widetilde{\phi_a}}{\langle \sigma, \texttt{assert}\ \ \widetilde{\phi_a}\,;\ \ s\rangle \mathrel{\widetilde{\longrightarrow}} \langle \sigma, s\rangle}\ \textsc{SsAssert}$$

⇔ $\widetilde{f}(\widetilde{\phi}) = \alpha(\overline{f}(\gamma(\widetilde{\phi})))$ where $\langle \alpha, \gamma \rangle$ is $\{\ \overline{f}\ \}$-partial Galois connection

Bonus: { F }-partial Galois connection

Definition 28 (Partial Galois connection). *Let* $(C, \sqsubseteq_C)$ *and* $(A, \sqsubseteq_A)$ *be two posets,* $\mathcal{F}$ *a set of operators on* C, $\alpha : C \rightharpoonup A$ *a partial function and* $\gamma : A \to C$ *a total function. The pair* $\langle \alpha, \gamma \rangle$ *is an* $\mathcal{F}$*-partial Galois connection if and only if:*

1. *If* $\alpha(c)$ *is defined, then* $c \sqsubseteq_C \gamma(\alpha(c))$, *and*
2. *If* $\alpha(c)$ *is defined, then* $c \sqsubseteq_C \gamma(a)$ *implies* $\alpha(c) \sqsubseteq_A a$, *and*
3. *For all* $F \in \mathcal{F}$ *and* $c \in C$, $\alpha(F(\gamma(c)))$ *is defined.*

This definition can be generalized for a set $\mathcal{F}$ *of arbitrary* n*-ary operators.*

Gradual Verification - Approach

Syntax

$s \in \textsc{Stmt}$

$\phi \in \textsc{Formula}$

Program State

$\pi \in \textsc{ProgramState}$

Semantics

Static $\vdash \{\phi\}\ s\ \{\phi\}$

Dynamic $\pi \longrightarrow \pi$

Formula $\pi \vDash \phi$

Soundness

Gradualization

Syntax

$\widetilde{s} \in \widetilde{\textsc{Stmt}}$

$\widetilde{\phi} \in \widetilde{\textsc{Formula}}$

Program State

$\widetilde{\pi} \in \widetilde{\textsc{ProgramState}}$

Semantics

Static $\widetilde{\vdash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}\}$

Dynamic $\widetilde{\pi} \mathrel{\widetilde{\longrightarrow}} \widetilde{\pi}$

Formula $\widetilde{\pi} \mathrel{\widetilde{\vDash}} \widetilde{\phi}$

Soundness

Gradual Verification - Approach

Syntax		**Syntax**
$s \in \textsc{Stmt}$	$\longrightarrow$	$\widetilde{s} \in \widetilde{\textsc{Stmt}}$
$\phi \in \textsc{Formula}$	syntax extension $\longrightarrow$ abstract interpretation	$\widetilde{\phi} \in \widetilde{\textsc{Formula}}$
Program State		**Program State**
$\pi \in \textsc{ProgramState}$	$\longrightarrow$	$\widetilde{\pi} \in \widetilde{\textsc{ProgramState}}$
Semantics		**Semantics**
Static $\vdash \{\phi\}\ s\ \{\phi\}$	predicate lifting $\longrightarrow$	Static $\widetilde{\vdash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}\}$
Dynamic $\pi \longrightarrow \pi$	partial function lifting $\longrightarrow$	Dynamic $\widetilde{\pi} \mathrel{\widetilde{\longrightarrow}} \widetilde{\pi}$
Formula $\pi \models \phi$	predicate lifting $\longrightarrow$	Formula $\widetilde{\pi} \mathrel{\widetilde{\models}} \widetilde{\phi}$
Soundness	$\longrightarrow$	**Soundness**

Gradual Soundness

$$\frac{\vdash \{\phi\}\ s\ \{\phi'\}}{\vDash \{\phi\}\ s\ \{\phi'\}}\ \textsc{Soundness}$$

$$\vDash \{\phi\}\ s\ \{\phi'\}$$

$$\overset{\text{def}}{\iff}$$

$$\forall \pi, \pi'.\ \pi \xrightarrow{s} \pi' \land \pi \vDash \phi \implies \pi' \vDash \phi'$$

$$\frac{\widetilde{\vdash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}'\}}{\widetilde{\vDash} \{\widetilde{\phi}\}\ \widetilde{s};\ \texttt{assert}\ \widetilde{\phi}'\ \{\widetilde{\phi}'\}}\ \widetilde{\textsc{S}}\textsc{oundness}$$

$$\widetilde{\vDash} \{\widetilde{\phi}\}\ \widetilde{s}\ \{\widetilde{\phi}'\}$$

$$\overset{\text{def}}{\iff}$$

$$\forall \widetilde{\pi}, \widetilde{\pi}'.\ \widetilde{\pi} \overset{\widetilde{s}}{\widetilde{\longrightarrow}} \widetilde{\pi}' \land \widetilde{\pi}\ \widetilde{\vDash}\ \widetilde{\phi} \implies \widetilde{\pi}'\ \widetilde{\vDash}\ \widetilde{\phi}'$$

$$\widetilde{\vdash} \{?\}\ \texttt{y := 4}\ \{\texttt{(x = 2)} \land \texttt{(y = 4)}\}$$

$$\widetilde{\vDash} \{?\}\ \texttt{y := 4; assert (x = 2)}\ \{\texttt{(x = 2)} \land \texttt{(y = 4)}\}$$

Gradual Verification – Put to the Test

$$\dfrac{\widetilde{\phi_1} \mathrel{\widetilde{\Rightarrow}} \widetilde{\phi_2} \qquad \mathrel{\widetilde{\vdash}} \{?\}\ \texttt{y := 2}\ \{\widetilde{\phi_1}\} \qquad \mathrel{\widetilde{\vdash}} \{\widetilde{\phi_2}\}\ \texttt{x := 3}\ \{(\texttt{x = 3}) \land (\texttt{y = 2})\}}{\mathrel{\widetilde{\vdash}} \{?\}\ \texttt{y := 2; x := 3}\ \{(\texttt{x = 3}) \land (\texttt{y = 2})\}}\ \widetilde{\mathrm{HS}}\textsc{eq}$$

Implicit Dynamic Frames

- Gradual Frame rule