A Unified Framework for Verification Techniques for Object Invariants

Peter Müller

Microsoft Research

Joint work with
Sophia Drossopoulou and Adrian Francalanza

Object Invariants

- Express consistency criteria of objects
- Support induction scheme
  - Established by constructors
  - Preserved by all methods
  - Potentially broken within method body
- Challenges
  - Call-backs
  - Multi-object invariants
  - Subclass invariants

Textbook Solution

```
class C {
  int a, b;
  invariant 0 ≤ a < b;

  C( ) {  a := 0; b := 3;  }

  void m( ) {
    int k := 10 / ( b – a );
    a := a + 3;
    n( );
    b := ( k + 4 ) * b;
  }

  n( ) {  m( );  }
}
```

Assume invariant

Prove invariant of C and its subclasses

```
class Client {
  C c;
  invariant ... ≤ 10;
}
```

(crossed out)

```
class Sub extends C {
  invariant a ≤ 10;
}
```

```
class Client {
  void set(C c) { c.a := -1; }
}
```

(crossed out)

Verification Techniques Differ in

- Invariant semantics
  - When are which invariants expected to hold?
- Admissible invariants
  - Which objects may an invariant depend on
- Proof obligations
  - What has to be proven when?
- Program restrictions
  - Which objects may receive method calls or field updates?
- Type systems

Approach

- Develop formal framework
  - Independent of type system and verification logic
  - Characterize techniques in terms of 7 framework parameters
- Define soundness
  - Identify 5 criteria on framework parameters that are sufficient for soundness
- Instantiate framework
  - Obtain six techniques from the literature

Areas and Regions

- Object areas
  - Describe sets of objects
  - Assume (do not define) interpretation

$$[[\ \mathbb{a}\ ]]h,o \subseteq \mathrm{dom}(\ h\ )$$

- Invariant regions
  - Describe sets of object-class pairs
  - Assume (do not define) interpretation

$$[[\ \mathbb{r}\ ]]h,o \subseteq \{\ (o', C) \mid \mathrm{class}(o') <: C\ \}$$

Areas and Regions: Example

- Areas

$$\mathbb{a} ::= \text{self} \mid \text{any}$$

$$[[\ \text{self}\ ]]_{h,o} = \{\ o\ \}$$
$$[[\ \text{any}\ ]]_{h,o} = \text{dom}(\ h\ )$$

- Regions

$$\mathbb{r} ::= \text{emp} \mid \text{self} \mid \text{any}$$

$$[[\ \text{emp}\ ]]_{h,o} = \{\ \ \}$$
$$[[[\ \text{self}\ ]]_{h,o} = \{\ (o, C) \mid \text{class}(o) <: C\ \}$$
$$[[\ \text{any}\ ]]_{h,o} = \{\ (o', C) \mid \text{class}(o') <: C\ \}$$

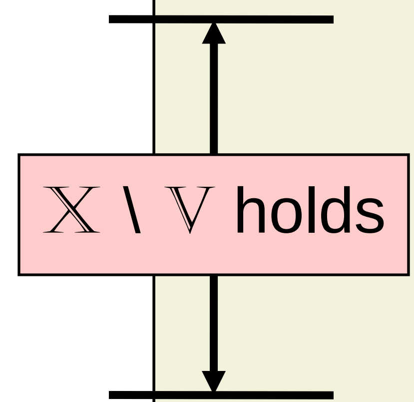
Framework Parameters

- Invariant semantics
  - $\mathbb{X}$(C) Invariants expected at visible states of C.m
  - $\mathbb{V}$(C) Invariants possibly violated by C.m
- Admissible invariants
  - $\mathbb{D}$(C) Invariants depending on C fields
- Proof obligations
  - $\mathbb{P}$(C,$\mathbb{a}$) Proof obligation for C.m before calls in $\mathbb{a}$
  - $\mathbb{E}$(C) Proof obligation C.m before end
- Program restrictions
  - $\mathbb{U}$(C,D) Objects whose D-fields may be updated by C.m
  - $\mathbb{C}$(C,D) Objects whose D-methods may be called from C.m

Meaning of Parameters: Example

```
invariant 0 ≤ this.a < this.b; // check (this,C) is in 𝔻

void m( ) {
  // assume 𝕏
  int k := 10 / ( b – a );
  this.a := this.a + 3;             // check this is in 𝕌
  // prove ℙ
  this.n( );                        // check this is in ℂ
  this.b := ( k + 4 ) * this.b;     // check this is in 𝕌
  // prove 𝔼
  // assume 𝕏
}
```



Parameters: Textbook Invariants

	Textbook
$\mathbb{X}$(C)	any
$\mathbb{V}$(C)	self
$\mathbb{D}$(C)	self
$\mathbb{P}$(C,$\mathbb{a}$)	emp
$\mathbb{E}$(C)	self
$\mathbb{U}$(C,D)	self
$\mathbb{C}$(C,D)	any

Programming Language

- Expressions and types

$$e ::= \textbf{this} \mid x \mid \ldots \mid e\&\textbf{prove}\ \mathbb{r}$$
$$t ::= C\ \mathbb{a}$$

- Runtime expressions

$$e ::= o \mid s \cdot e \mid \ldots \mid \text{FatalExc} \mid \text{VerifExc}$$

- Assume judgment: $h \models o,C$
- Assume (do not define) type system
  - Main judgment $\Gamma \vdash e : C\ \mathbb{a}$
  - Make requirements (soundness, etc.)

Invariant Semantics

- Method calls

$$\frac{h \models [\![\mathbb{X}]\!]_{h,s(\textbf{this})}}{s \cdot \textbf{call } e, h \rightarrow s \cdot \textbf{ret } e, h}$$

$$\frac{h \not\models [\![\mathbb{X}]\!]_{h,s(\textbf{this})}}{s \cdot \textbf{call } e, h \rightarrow \text{FatalExc}, h}$$

- Method termination

$$\frac{h \models [\![\mathbb{X}]\!]_{h,s(\textbf{this})}}{s \cdot \textbf{ret } v, h \rightarrow v, h}$$

$$\frac{h \not\models [\![\mathbb{X}]\!]_{h,s(\textbf{this})}}{s \cdot \textbf{ret } v, h \rightarrow \text{FatalExc}, h}$$

- Proof obligation

$$\frac{h \models [\![\mathbb{r}]\!]_{h,s(\textbf{this})}}{s \cdot v \,\&\, \textbf{prove } \mathbb{r}, h \rightarrow v, h}$$

$$\frac{h \not\models [\![\mathbb{r}]\!]_{h,s(\textbf{this})}}{s \cdot v \,\&\, \textbf{prove } \mathbb{r}, h \rightarrow \text{VerifExc}, h}$$

Verified Programs

- Field updates: check area

$$\frac{\Gamma \vdash e : C\ a \qquad a \subseteq \mathbb{U}(class(\Gamma), C') \qquad \text{C has field f defined in C'} \qquad \ldots}{\Gamma \vdash_{vf} e.f := e'}$$

- Method calls: check area, verify invariants

$$\frac{\Gamma \vdash e : C\ a \qquad a \subseteq \mathbb{C}(class(\Gamma), C') \qquad \text{C inherits m from C'} \qquad \ldots}{\Gamma \vdash_{vf} e.m(e'\ \&\mathbf{prove}\ \mathbb{P}(C', a))}$$

Soundness

- A verification technique is sound if the following properties hold for each well-typed, verified program P:
- Execution of P does not lead to FatalExc
- In each execution state σ of P, the following properties hold for each stack frame for a method C.m:
  - The invariants in $\mathbb{X}$(C) \ $\mathbb{V}$(C) hold
  - If σ is a visible state of m, the invariants in $\mathbb{X}$(C) hold

Soundness Criteria

S1: $\mathbb{a} \subseteq \mathbb{C}(C,D) \Rightarrow (\mathbb{a} \blacktriangleright \mathbb{X}(D)) \setminus (\mathbb{X}(C) \setminus \mathbb{V}(C)) \subseteq \mathbb{P}(C,\mathbb{a})$

S2: $\mathbb{V}(C) \cap \mathbb{X}(C) \subseteq \mathbb{E}(C)$

S3: $\mathbb{C}(C,D) \blacktriangleright (\ \mathbb{V}(D) \setminus \mathbb{X}(D)\ ) \subseteq \mathbb{V}(C)$

S4: $\mathbb{U}(C,D) \blacktriangleright \mathbb{D}(D) \subseteq \mathbb{V}(C)$

S5: $C <: D \Rightarrow \begin{cases} \mathbb{X}(C) \subseteq \mathbb{X}(D) \\ \mathbb{V}(C) \setminus \mathbb{X}(C) \subseteq \mathbb{V}(D) \setminus \mathbb{X}(D) \end{cases}$

Soundness Theorem

A verification technique that satisfies S1 – S5 is sound

Soundness: Textbook Invariants

$\mathbb{X}$(C)	any
$\mathbb{V}$(C)	self
$\mathbb{D}$(C)	self
$\mathbb{P}$(C,$\mathbb{a}$)	emp
$\mathbb{E}$(C)	self
$\mathbb{U}$(C,D)	self
$\mathbb{C}$(C,D)	any

any \ (any \ self) $\subseteq$ emp

self $\cap$ any $\subseteq$ self

any ▶ (self \ any) $\subseteq$ self

self ▶ self $\subseteq$ self

C <: D $\Rightarrow$ { any $\subseteq$ any ; self \ any $\subseteq$ self \ any }

Summary

- Parametric w.r.t. underlying type system and verification logic
- Abstract explanation how technique works
- Criteria for comparisons
- Guidelines for the development of further techniques
- Instantiation for six techniques from the literature, found one unsoundness