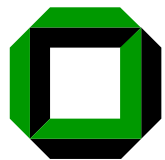

Computing Specifications

*From Implementation to
Specification by Construction.*

André Platzer



University of Karlsruhe

Overview

- What means computing specifications
- Applications
- Feasibility
- How to do it
- Examples
- Summary & Discussion

Verification in reverse

- Given a program, automatically find its specification.
- Not just any specification, better yet the *strongest* specification that the program satisfies.

Applications

- Reverse-engineer existing code into a formal model.
- Auto-complete partial specifications of methods.
- Explain what a complex piece of code really does. (Expectation? Special cases?)

Terminology I

- ψ is a *specification* of a program α if

$$\models \mathbf{x} \doteq x^{\text{pre}} \longrightarrow [\alpha]\psi$$

Terminology I: Example

- ψ is a specification if

$$\models \mathbf{x} \doteq x^{\text{pre}} \wedge \mathbf{y} \doteq y^{\text{pre}} \rightarrow$$
$$[\{\text{int } \mathbf{t} = \mathbf{x}; \mathbf{x} = \mathbf{y}; \mathbf{y} = \mathbf{t}; \}] \psi$$

Terminology I: Example

- ψ is a specification if

$$\models \mathbf{x} \doteq x^{\text{pre}} \wedge \mathbf{y} \doteq y^{\text{pre}} \rightarrow$$
$$[\{\text{int } t = \mathbf{x}; \mathbf{x} = \mathbf{y}; \mathbf{y} = t; \}] \psi$$

- $\psi_1 \quad := \quad \mathbf{x} \doteq y^{\text{pre}} \wedge \mathbf{y} \doteq x^{\text{pre}} \quad \checkmark$

- $\psi_2 \quad := \quad \mathbf{x} - \mathbf{y} \doteq y^{\text{pre}} - x^{\text{pre}} \quad \checkmark$

- $\psi_3 \quad := \quad \mathbf{x} \neq x^{\text{pre}} \wedge \mathbf{y} \neq y^{\text{pre}} \quad \times$

Terminology II

- ψ is *stronger* than ϕ :

$$\phi \preceq \psi \quad :\iff \quad \psi \vDash \phi$$

Terminology II: Example

- ψ is *stronger* than ϕ :

$$\phi \preceq \psi \quad :\iff \quad \psi \vDash \phi$$

- $\mathbf{x} - \mathbf{y} \doteq y^{\text{pre}} - x^{\text{pre}}$

$$\preceq \quad \mathbf{x} \doteq y^{\text{pre}} \wedge \mathbf{y} \doteq x^{\text{pre}}$$

Terminology II: Example

- ψ is *stronger* than ϕ :

$$\phi \preceq \psi \quad :\iff \quad \psi \vDash \phi$$

- $\mathbf{x} - \mathbf{y} \doteq y^{\text{pre}} - x^{\text{pre}}$

$$\preceq \quad \mathbf{x} \doteq y^{\text{pre}} \wedge \mathbf{y} \doteq x^{\text{pre}}$$

- Do strongest specifications always exist?
Are they unique?

Terminology II: Example

- ψ is *stronger* than ϕ :

$$\phi \preceq \psi \quad :\iff \quad \psi \vDash \phi$$

- $\mathbf{x} - \mathbf{y} \doteq y^{\text{pre}} - x^{\text{pre}}$

$$\preceq \quad \mathbf{x} \doteq y^{\text{pre}} \wedge \mathbf{y} \doteq x^{\text{pre}}$$

- Do strongest specifications always exist? **Yes!**

Are they unique? **Almost!**

- prefer “more readable” ψ .

Construction: Motivation

```
if (x > 9) {  
    x = x + 2;  
} else {  
    x = x - 1;  
}  
x = x + 4;
```

What specification? Call it \mathcal{C} ?

Construction: Motivation

```
if (x > 9) {  
    x = x + 2;  
} else {  
    x = x - 1;  
}  
x = x + 4;
```

What specification? Call it \mathcal{C} ?

$$\frac{\frac{\frac{x > 9 \vdash \langle x = x + 6 \rangle \mathcal{C}}{x > 9 \vdash \langle x = x + 2 + 4 \rangle \mathcal{C}}{x > 9 \vdash \langle x = x + 2 \rangle \langle x = x + 4 \rangle \mathcal{C}} \quad \frac{\frac{x \leq 9 \vdash \langle x = x + 3 \rangle \mathcal{C}}{x \leq 9 \vdash \langle x = x - 1 + 4 \rangle \mathcal{C}}{x \leq 9 \vdash \langle x = x - 1 \rangle \langle x = x + 4 \rangle \mathcal{C}}}{\vdash \langle \alpha \rangle \mathcal{C}}$$

Automatic construction

1. Call \mathcal{C} a constant symbol of type formula.
2. Prepare (special) proof obligation.

$$\vdash \mathbf{x} \doteq x^{\text{pre}} \rightarrow \langle \alpha \rangle \mathcal{C}$$

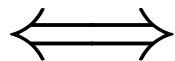
3. Perform sequent calculus.
4. Use (failed) proof attempts.
5. Extract specification.

Inference rules

$$\frac{D_1 \quad \dots \quad D_n}{C}$$

allowed if $\forall I \quad \forall s$

$$I, s \models C$$



$$I, s \models D_1 \wedge \dots \wedge D_n$$

Ex: mutual branch

```
if ( x < 0 ) {  
    x = 0;  
} else {  
    x = x + 1;  
}  
if ( 0 > x ) {  
    x = 17;  
} else {  
    x = x + 2;}
```


Ex: mutual branch

- Implementation result (accumulator mode)

$$\begin{aligned} \psi_{\mathcal{C}} \quad := \quad & (\mathbf{x} < 0 \rightarrow \langle \mathbf{x} = 2 \rangle \mathcal{C}) \\ & \wedge (\mathbf{x} \geq 0 \rightarrow \langle \mathbf{x} = \mathbf{x} + 3 \rangle \mathcal{C}) \end{aligned}$$

Ex: mutual branch

- Implementation result (accumulator mode)

$$\begin{aligned}\psi_{\mathcal{C}} &:= (\mathbf{x} < 0 \rightarrow \langle \mathbf{x} = 2 \rangle \mathcal{C}) \\ &\quad \wedge (\mathbf{x} \geq 0 \rightarrow \langle \mathbf{x} = \mathbf{x} + 3 \rangle \mathcal{C})\end{aligned}$$

- Implementation result (equation mode)

$$\begin{aligned}\psi' &:= (x^{\text{pre}} < 0 \rightarrow x^{\text{post}} \doteq 2) \\ &\quad \wedge (x^{\text{pre}} \geq 0 \rightarrow x^{\text{post}} \doteq x^{\text{pre}} + 3)\end{aligned}$$

Ex: algebraic loop

```
int i = 0;
while ( i < 4 ) {
    s = s + n*i;
    i++;
}
```

- Implementation result (equation mode)

$$\psi_C := s^{\text{post}} \doteq s^{\text{pre}} + \underbrace{6 * n}_{\text{simplified}} \wedge i^{\text{post}} \doteq 5$$

Ex: Diophantine equation

```
for ( int i = 1; i < 10; i ++ ) {
    for ( int j = 1; j < 10; j ++ ) {
        if ( i * i + j + 1 == i * j )
            // return "pair" (i,j)
            return ( 100 * i + j );
    }
}

return -1;
```

Implementation

- As an additional KeY module.
- It works.
- Still some inconveniences:
unnecessary internal variables.

Cons

- Specification collapses sometimes.
- Branching cases not (yet) reconciled.
- Unbounded loops are a major problem.

Pros

- + Apart from unbounded loops we have strongest specs.
- + No ad-hoc method but well-understood logic.
- + Close to theory.
- + Theorem provers are applicable.

Summary 1

- Computing specifications is useful.
- The strongest specification exists and is (rather) unique.
- Automatic specification construction is effective.
- Readability is the clue.

Summary 2

- Theoretical construction does not achieve readability.
- Theorem prover approach is better.
- Sequent calculus is applicable (in a natural way).
- Construction of logical representation \neq inference of all program properties.

Repository

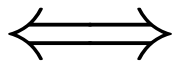
- The end of the presentation

Connexion

- ψ is a *specification* of program α if

$$\models \mathbf{x} \doteq x^{\text{pre}} \longrightarrow$$

$$[\alpha]\psi(x)$$



$$\models \mathbf{x} \doteq x^{\text{pre}} \longrightarrow$$

$$\left(\langle \alpha \rangle \mathbf{x} \doteq x^{\text{post}} \longrightarrow \psi \right)$$

Overall statement

$$\forall \mathcal{C} \quad \forall I \quad \forall s$$
$$I, s \models \mathbf{x} \doteq x^{\text{pre}} \rightarrow$$
$$(\langle \alpha \rangle \mathcal{C} \leftrightarrow \psi)$$

Especially

$$\mathcal{C} \quad := \quad \mathbf{x} \doteq x^{\text{post}}$$

Comparison with sp

- $sp(P, \ulcorner s \urcorner) = Q \quad \Rightarrow \quad \models P \rightarrow \langle \mathbf{s} \rangle Q$

Comparison with sp

- $sp(P, \ulcorner s \urcorner) = Q \quad \Rightarrow \quad \models P \rightarrow \langle \mathbf{s} \rangle Q$

-

$$\frac{\Gamma, b \vdash \langle \mathbf{s} \rangle \phi, \Delta \quad \Gamma, \neg b \vdash \langle \mathbf{t} \rangle \phi, \Delta}{\Gamma \vdash \langle \mathbf{if}(b) \ s \ \mathbf{else} \ t \rangle \phi, \Delta}$$

-

$$sp(P, \ulcorner \mathbf{if} \ (b) \sim s \sim \mathbf{else} \sim t \urcorner) = \\ sp(P \wedge b, \ulcorner s \urcorner) \vee sp(P \wedge \neg b, \ulcorner t \urcorner)$$

Comparison with sp

- $sp(P, \lceil s \rceil) = Q \quad \Rightarrow \quad \models P \rightarrow \langle s \rangle Q$

-

$$\frac{\Gamma[\mathbf{x} \mapsto y], \mathbf{x} \doteq t[\mathbf{x} \mapsto y] \vdash \phi, \Delta[\mathbf{x} \mapsto y]}{\Gamma \vdash \langle \mathbf{x} := \mathbf{t} \rangle \phi, \Delta}$$

where y is a new logical variable.

-

$$sp(P, \lceil \mathbf{x} := \mathbf{t} \rceil) =$$

$$\exists x_0 (P[x \mapsto x_0] \wedge x \doteq t[x \mapsto x_0])$$

Strongest specification ψ_α

- $\forall \alpha \exists \psi_\alpha$

$$\models \quad \forall x^{\text{pre}} \quad \forall x^{\text{post}} \\ \left(\mathbf{x} \doteq x^{\text{pre}} \rightarrow \left(\langle \alpha \rangle \mathbf{x} \doteq x^{\text{post}} \leftrightarrow \psi_\alpha \right) \right)$$

where $FV(\psi_\alpha) \subseteq \{x^{\text{pre}}, x^{\text{post}}\}$.

List of inference rules

“+”

$$\frac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}$$

and-right

List of inference rules

“ ”

$$\frac{\Gamma \vdash \Delta}{\Gamma, \phi \vdash \Delta}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \phi}$$

weakening

List of inference rules

“ ”

$$\frac{\Gamma, \phi(X_1, \dots, X_n, s(X_1, \dots, X_n)) \vdash \Delta}{\Gamma, \exists y \phi(X_1, \dots, X_n, y) \vdash \Delta}$$

Skolemise where $\{X_1, \dots, X_n\}$ are the free variables occurring in $\exists y \phi(X_1, \dots, X_n, y)$ and s is a new Skolem-function constant.

List of inference rules

“+”

$$\frac{\Gamma, \phi[y \mapsto \epsilon y \phi] \vdash \Delta}{\Gamma, \exists y \phi \vdash \Delta}$$

ϵ -rule, sometimes called “critical axiom”.

List of inference rules

“+”

$$\frac{\Gamma, \phi[x \mapsto t], \forall x \phi \vdash \Delta}{\Gamma, \forall x \phi \vdash \Delta}$$

universal quantifier

List of inference rules

“+”

$$\frac{\Gamma, b \vdash \langle \mathbf{s} \rangle \phi, \Delta \quad \Gamma, \neg b \vdash \langle \mathbf{t} \rangle \phi, \Delta}{\Gamma \vdash \langle \mathbf{if}(b) \ \mathbf{s} \ \mathbf{else} \ \mathbf{t} \rangle \phi, \Delta}$$

branch

List of inference rules

“ ”

$$\frac{\Gamma \vdash b, \Delta \quad \Gamma, b \vdash \langle \mathbf{s} \rangle \phi, \Delta}{\Gamma \vdash \langle \mathbf{if}(b) \ \mathbf{s} \ \mathbf{else} \ \mathbf{t} \rangle \phi, \Delta}$$

“weakening” single-side branch

List of inference rules

“ ”

$$\frac{\Gamma \vdash \phi \quad \Gamma, \phi \vdash \Delta}{\Gamma \vdash \Delta}$$

cut with weakening

List of inference rules

“+”

$$\frac{\Gamma \vdash \phi, \Delta \quad \Gamma, \phi \vdash \Delta}{\Gamma \vdash \Delta}$$

cut.

- But good idea?

List of inference rules

“+”

$$\frac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \neg\phi, \Delta}{\Gamma \vdash \Delta}$$

cut-derived

List of inference rules

“+”

$$\frac{\Gamma \vdash \langle \text{if}(b)\{a; \text{while}(b)a\} \rangle \phi, \Delta}{\Gamma \vdash \langle \text{while}(b)a \rangle \phi, \Delta}$$

unwind loop once

List of inference rules

“ ”

$$\frac{\Gamma \vdash I \quad I, b \vdash [a]I \quad I, \neg b \vdash \phi, \Delta}{\Gamma \vdash [\text{while}(b)a]\phi, \Delta}$$

loop induction with invariant I .

List of inference rules

“ ”

$$\frac{\Gamma[\mathbf{x} \mapsto y], \mathbf{x} \doteq t[\mathbf{x} \mapsto y] \vdash \phi, \Delta[\mathbf{x} \mapsto y]}{\Gamma \vdash \langle \mathbf{x} := \mathbf{t} \rangle \phi, \Delta}$$

assignment rule where y is a new logical variable (implicit universally quantified).

List of inference rules

“+”

$$\frac{\Gamma \vdash \phi[\mathbf{x} \mapsto t], \Delta}{\Gamma \vdash \langle \mathbf{x} := \mathbf{t} \rangle \phi, \Delta}$$

Example: unifyRobinson

==>

```
(  t2 = self.IS_VARIABLE
 | t1 = self.IS_VARIABLE
 | {result:=1}
   ^true)
& (  t2 + self.OCCURS_IN < t1
    & t2 = self.IS_VARIABLE
  -> t1 = self.IS_VARIABLE
    | {result:=0}
     ^true)
& ...
```

Example: unifyRobinson

```
...
& ( t2 = self.IS_VARIABLE
    -> t2 + self.OCCURS_IN < t1
    | t1 = self.IS_VARIABLE
    | {result:=t2 + -t1}
      ^true)
& ( t1 + self.OCCURS_IN < t2
    & t1 = self.IS_VARIABLE
    -> {result:=0}
      ^true)
& ...
```


Example: unifyRobinson

...

```
& ( t1 = self.IS_VARIABLE  
  -> t1 + self.OCCURS_IN < t2  
    | {result:=t1 + -t2}  
      ^true)
```

Example: mathPolyPuzzle

==>

```
{ i := 2,  
  j := 5,  
  result := 205 }  
^true
```

(Example: symbolic loop)

```
int loopy(int c, int a) {  
    for (int i=0; i<3; i++) {  
        c = c + a;  
    }  
    return c;  
}
```

- Implementation result (equation mode)

$$\psi' := (\text{result} \doteq c + \underbrace{a + a + a}_{3*a})$$