Program Verification

Benjamin Schmid

2019-02-12

Swiss Olympiad in Informatics

Program Verification

What is Program Verification?

SMT Solvers

Weakest Liberal Precondition

Verification Code Generation

Based on Lecture "Program Verification" by Alexander Summers.

http://www.pm.inf.ethz.ch/education/courses/program-verification.html

What is Program Verification?

What is Program Verification?

Program Verification
Given a program, prove that it always behaves correctly.

Specification
Write down mathematically, what the program is supposed to do.

Contracts
Specify what has to hold before a method (precondition) and what will hold after a method (postcondition).

Contracts

```
method negate_positive(i: Int)
    returns (res: Int)


{
    res := -i
}
```

Contracts

```
method negate_positive(i: Int)
    returns (res: Int)
    requires i > 0
    ensures res < 0
{
    res := -i
}
```

Contracts

```
method sum(a: Int, b: Int)
    returns (res: Int)

{
    res := a + b
}
```

Contracts

```
method sum(a: Int, b: Int)
    returns (res: Int)
    ensures res == a + b
{
    res := a + b
}
```

SMT Solvers

SMT Solvers

SMT Problem
Given a first-order logic formula, with interpreted/uninterpreted symbols from (possibly several) theories, does it have a model?

$$\forall i \in \mathbb{N}.\, f(i) > 0 \land f(i) < f(i+1)$$

$$\exists i \in \mathbb{N}.\, \forall j \in \mathbb{N}.\, j \geq i \implies j = i$$

Weakest Liberal Precondition

Small Imperative Language

- Variables x, y, z, ...
- Expressions (e.g. $x + 4$)
- `skip` (does nothing)
- `x := e`
- `s1;s2`
- `if(b) { s1 } else { s2 }`
- `while(b) invariant A { s }`

Weakest Precondition

For some postcondition A and a statement (or program) s we define weakest precondition $wlp(s, A)$.

We want $wlp(s, A)$ to be:

- Sound: $\forall s, A.\ \models \{wlp(s, A)\}s\{A\}$
  if $wlp(s, A)$ holds before s, then A has to hold after it
- Minimal: $\forall s, A_1, A_2$. if $\models \{A_1\}s\{A_2\}$ then $A_1 \models wlp(s, A_2)$
  $wlp(s, A)$ can not be further "simplified"
- Efficiently Computable

Weakest Precondition

- $wlp(skip, A) = A$
- $wlp(x := e, A) = A[e/x]$
- $wlp(s_1; s_2, A) = wlp(s_1, wlp(s_2, A))$
- $wlp(\text{if}(b)\{s_1\}\text{else}\{s_2\}, A) =$
  $(b \implies wlp(s_1, A)) \wedge (\neg b \implies wlp(s_2, A))$
- $wlp(\text{while}(b) \text{ invariant } A_I\{s\}, A) =$
  $A_I \wedge \forall \vec{y}.\ ((A_I \wedge b \implies wlp(s, A_I)) \wedge (A_I \wedge \neg b \implies A))[\vec{y}/\vec{x}]$

Verification Code Generation

Given an annotated program $\{pre\}s\{post\}$

Ask the SMT solver

If $pre \wedge \neg wlp(s, post)$ is satisfiable, program is not correct

If it is not, the program is proven to be correct