08—Program Verification II

CS 5209: Foundation in Logic and AI

Martin Henz and Aquinas Hobor

March 11, 2010

Generated on Thursday 11th March, 2010, 16:11

Expressions in Core Language

Expressions come as arithmetic expressions E:

$$E ::= n \mid x \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

and boolean expressions B:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \& B) \mid (B \| B) \mid (E < E)$$

Where are the other comparisons, for example ==?

Commands in Core Language

Commands cover some common programming idioms.
Expressions are components of commands.

$$C ::= x = E \mid C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}$$

Example

Consider the factorial function:

$$\begin{aligned} 0! &\stackrel{\text{def}}{=} 1 \\ (n+1)! &\stackrel{\text{def}}{=} (n+1) \cdot n! \end{aligned}$$

We shall show that after the execution of the following Core program, we have $y = x!$.

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

Example

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

Example

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

- We need to be able to say that at the end, y is x!, provided that at the beginning, we have $x \geq 0$.

Assertions on Programs

Shape of assertions

$$(\!|\phi|\!)\ P\ (\!|\psi|\!)$$

Informal meaning

If the program *P* is run in a state that satisfies ϕ, then the state resulting from *P*'s execution will satisfy ψ.

Partial Correctness

Definition

We say that the triple $(\!|\phi|\!)$ *P* $(\!|\psi|\!)$ is *satisfied under partial correctness* if, for all states which satisfy ϕ, the state resulting from *P*'s execution satisfies ψ, provided that *P* terminates.

Notation

We write $\models_{\text{par}} (\!|\phi|\!)$ *P* $(\!|\psi|\!)$.

Total Correctness

Definition

We say that the triple $(\!|\phi|\!)$ *P* $(\!|\psi|\!)$ is *satisfied under total correctness* if, for all states which satisfy ϕ, *P* is guaranteed to terminate and the resulting state satisfies ψ.

Notation

We write $\models_{\mathrm{tot}} (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

Back to Factorial

Consider Fac1:

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

- $\models_{\text{tot}} (\!| x \geq 0 |\!)$ Fac1 $(\!| y = x! |\!)$
- $\not\models_{\text{tot}} (\!| \top |\!)$ Fac1 $(\!| y = x! |\!)$

Back to Factorial

Consider `Fac1`:

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

- $\models_{\text{tot}} (\!| x \geq 0 |\!)$ `Fac1` $(\!| y = x! |\!)$
- $\models_{\text{par}} (\!| \top |\!)$ `Fac1` $(\!| y = x! |\!)$

Rules for Partial Correctness

$$\frac{(\!|\phi|\!)\ C_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ C_1; C_2\ (\!|\psi|\!)}\ \text{[Composition]}$$

Rules for Partial Correctness (continued)

$$\frac{}{(\!|[x \to E]\psi|\!)\ x = E\ (\!|\psi|\!)}\ [\text{Assignment}]$$

Rules for Partial Correctness (continued)

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{\ C_1\ \}\ \texttt{else}\ \{\ C_2\ \}\ (\!|\psi|\!)}\ \text{[If-statement]}$$

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{\ C\ \}\ (\!|\psi \wedge \neg B|\!)}\ \text{[Partial-while]}$$

Rules for Partial Correctness (continued)

$$\frac{\vdash_{AR} \phi' \rightarrow \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \rightarrow \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ [\text{Implied}]$$

Proof Tableaux

Proofs have tree shape

All rules have the structure

$$\frac{\text{something}}{\text{something else}}$$

As a result, all proofs can be written as a tree.

Practical concern

These trees tend to be very wide when written out on paper. Thus we are using a linear format, called *proof tableaux*.

Interleave Formulas with Code

$$\frac{(\!|\phi|\!)\ C_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ C_1; C_2\ (\!|\psi|\!)}\ [\text{Composition}]$$

Shape of rule suggests format for proof of $C_1; C_2; \ldots; C_n$:

$(\!|\phi_0|\!)$
$C_1;$
$(\!|\phi_1|\!)$ justification
$C_2;$
$\vdots$
$(\!|\phi_{n-1}|\!)$ justification
$C_n;$
$(\!|\phi_n|\!)$ justification

Working Backwards

Overall goal

Find a proof that at the end of executing a program *P*, some condition ψ holds.

Common situation

If *P* has the shape $C_1; \ldots; C_n$, we need to find the weakest formula ψ' such that

$$(\!|\psi'|\!)\ C_n\ (\!|\psi|\!)$$

Terminology

The weakest formula ψ' is called *weakest precondition*.

Example

$(\!|y < 3|\!)$
$(\!|y + 1 < 4|\!)$ Implied
`y = y + 1;`
$(\!|y < 4|\!)$ Assignment

Another Example

Can we claim $u = x + y$ after `z = x; z = z + y; u = z;` ?

$(\!|\top|\!)$
$(\!|x + y = x + y|\!)$ Implied
`z = x;`
$(\!|z + y = x + y|\!)$ Assignment
`z = z + y;`
$(\!|z = x + y|\!)$ Assignment
`u = z;`
$(\!|u = x + y|\!)$ Assignment

An Alternative Rule for If

We have:

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{\ C_1\ \}\ \texttt{else}\ \{\ C_2\ \}\ (\!|\psi|\!)}\ \text{[If-statement]}$$

Sometimes, the following *derived rule* is more suitable:

$$\frac{(\!|\phi_1|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi_2|\!)\ C_2\ (\!|\psi|\!)}{(\!|(B \rightarrow \phi_1) \wedge (\neg B \rightarrow \phi_2)|\!)\ \texttt{if}\ B\ \{\ C_1\ \}\ \texttt{else}\ \{\ C_2\ \}\ (\!|\psi|\!)}\ \text{[If-stmt 2]}$$

Example

Consider this implementation of Succ:

```
a = x + 1;
if (a - 1 == 0) {
    y = 1;
} else {
    y = a;
}
```

Can we prove $(\!|\top|\!)$ Succ $(\!|y = x + 1|\!)$?

Another Example

$\vdots$

if (a $-$ 1 == 0) $\{$

$(\!|1 = x + 1|\!)$ If-Statement 2

y = 1;

$(\!|y = x + 1|\!)$ Assignment

$\}$ else $\{$

$(\!|a = x + 1|\!)$ If-Statement 2

y = a;

$(\!|y = x + 1|\!)$ Assignment

$\}$

$(\!|y = x + 1|\!)$ If-Statement 2

Another Example

$(\!|\top|\!)$

$(\!|(x + 1 - 1 = 0 \rightarrow 1 = x + 1) \wedge$

$(\neg(x + 1 - 1 = 0) \rightarrow x + 1 = x + 1)|\!)$ Implied

```
a = x + 1;
```

$(\!|(a - 1 = 0 \rightarrow 1 = x + 1) \wedge$

$(\neg(a - 1 = 0) \rightarrow a = x + 1)|\!)$ Assignment

if (a $-$ 1 == 0) $\{$

$(\!|1 = x + 1|\!)$ If-Statement 2

y = 1;

$(\!|y = x + 1|\!)$ Assignment

$\}$ else $\{$

$(\!|a = x + 1|\!)$ If-Statement 2

y = a;

$(\!|y = x + 1|\!)$ Assignment

Recall: Partial-while Rule

$$\frac{(\!|\psi \land B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{\ C\ \}\ (\!|\psi \land \neg B|\!)}\ \text{[Partial-while]}$$

Factorial Example

We shall show that the following Core program `Fac1` meets this specification:

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

Thus, to show:

$$(\!|\top|\!)\ \texttt{Fac1}\ (\!|y = x!|\!)$$

Partial Correctness of `Fac1`

$\vdots$

$(\!|\, y = z! \,|\!)$	
`while ( z != x ) {`	
$\quad(\!|\, y = z! \land z \neq x \,|\!)$	Invariant
$\quad(\!|\, y \cdot (z+1) = (z+1)! \,|\!)$	Implied
`    z = z + 1;`	
$\quad(\!|\, y \cdot z = z! \,|\!)$	Assignment
`    y = y * z;`	
$\quad(\!|\, y = z! \,|\!)$	Assignment
`}`	
$(\!|\, y = z! \land \neg(z \neq x) \,|\!)$	Partial-while
$(\!|\, y = x! \,|\!)$	Implied

Partial Correctness of Fac1

$(\| \top \|)$	
$(\| (1 = 0!) \|)$	Implied
y = 1;	
$(\| y = 0! \|)$	Assignment
z = 0;	
$(\| y = z! \|)$	Assignment
while (z != x) {	
$\vdots$	
}	
$(\| y = z! \wedge \neg(z \neq x) \|)$	Partial-while
$(\| y = x! \|)$	Implied

Ideas for Total Correctness

- The only source of non-termination is the `while` command.
- If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination.
  Why? Well-foundedness of natural numbers
- We shall include this argument in a new version of the `while` rule.

Rules for Partial Correctness (continued)

$$\frac{(\!|\psi \land B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{\ C\ \}\ (\!|\psi \land \neg B|\!)}\ \text{[Partial-while]}$$

$$\frac{(\!|\psi \land B \land 0 \leq E = E_0|\!)\ C\ (\!|\psi \land 0 \leq E < E_0|\!)}{(\!|\psi \land 0 \leq E|\!)\ \texttt{while}\ B\ \{\ C\ \}\ (\!|\psi \land \neg B|\!)}\ \text{[Total-while]}$$

Factorial Example (Again!)

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

What could be a good variant E?

Factorial Example (Again!)

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

What could be a good variant E?

E must strictly decrease in the loop, but not become negative.

Factorial Example (Again!)

```
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
```

What could be a good variant E?

E must strictly decrease in the loop, but not become negative.

Answer:

$$x - z$$

Total Correctness of `Fac1`

$\vdots$

$(\!| y = z! \land 0 \leq x - z |\!)$

`while ( z != x ) {`

$(\!| y = z! \land z \neq x \land 0 \leq x - z = E_0 |\!)$	Invariant
$(\!| y \cdot (z+1) = (z+1)! \land 0 \leq x - (z+1) < E_0 |\!)$	Implied
`z = z + 1;`	
$(\!| y \cdot z = z! \land 0 \leq x - z < E_0 |\!)$	Assignment
`y = y * z;`	
$(\!| y = z! \land 0 \leq x - z < E_0 |\!)$	Assignment
`}`	
$(\!| y = z! \land \neg(z \neq x) |\!)$	Total-while
$(\!| y = x! |\!)$	Implied

Total Correctness of `Fac1`

$(\!| x \leq 0 |\!)$
$(\!| (1 = 0! \land 0 \leq x - 0 |\!)$ Implied
`y = 1;`
$(\!| y = 0! \land 0 \leq x - 0 |\!)$ Assignment
`z = 0;`
$(\!| y = z! \land 0 \leq x - z |\!)$ Assignment
`while ( z != x ) {`
$\vdots$
`}`
$(\!| y = z! \land \neg(z \neq x) |\!)$ Total-while
$(\!| y = x! |\!)$ Implied