Formal Methods in software development

a.y.2017/2018

Prof.Anna Labella

Looking for a program: example

Given (|x> 0|) S (| y.y < x |)

many possible solutions for S:

```
y = 0;
```

```
y = 0;
while (y * y < x) {
    y = y + 1;
    }
y = y - 1;
```

Example (assignment and concatenation)

Write down a program P such that

(a) $(\!|\top|\!)\ P\ (\!|y = x + 2|\!)$

(b) $(\!|\top|\!)\ P\ (\!|z > x + y + 4|\!)$

Example (assignment and concatenation)

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

$(\!| x+1 + 1 = x + 2 |\!)$

t = x + 1;

$(\!| t + 1 = x + 2 |\!)$

z = t + 1;

$(\!| z = x + 2 |\!)$

y = z;

$(\!| y = x + 2 |\!)$

Example (if then else)

For each of the specifications below, write code for P and prove the partial correctness of the specified input/output behaviour:

(a) $(\!|\top|\!)\ P\ (\!|z = \max(w, x, y)|\!)$, where $\max(w, x, y)$ denotes the largest of w, x and y.

(b) $(\!|\top|\!)\ P\ (\!|((x = 5) \to (y = 3)) \land ((x = 3) \to (y = -1))|\!)$.

Using the method bottom-up

Syntesizing the program computing n^3

We look for assignments and build loop

First postcondition

`c=N`^{3}

We transform it in order to obtain the postcondition of a loop $I \wedge \neg C$

$c=x^3 \wedge x=N$

Metaprogram

```
1. establish I
2. while x≠N do
3. Preserve I while making x closer to N done;
```

invariant I $c=x^3$ variant $N-x$

Syntesizing the program computing n^3

```
3. Preserve I while making x closer to N done;
```
invariant I `c=x`^{3} variant `N-x`

becomes

```
3. x,c := x+1, E done;
```
invariant I `c=x`^{3} variant `N-x`

Where , to preserve invariant

`I ∧ x≠N ⇒ [x,c := x+1, E] I`

invariant I `c=x`^{3} variant `N-x`

Syntesizing the program computing n^3

$[x,c := x+1,\ E]\ I =$
$[x,c := x+1,\ E]\ c=x^3 =$
$E = (x+1)^3 =$
$E = x^3+3x^2+3x+1 =$
$E = c+3x^2+3x+1$

$E = c+d$
$d=3x^2+3x+1$
invariant $I \wedge I_2$ $c=x^3 \wedge d=3x^2+3x+1$ variant $N-x$

Where , to preserve invariant
$c=x^3 \wedge d=3x^2+3x+1 \Rightarrow [x,c := x+1,\ c+d]\ (c=x^3)$
But is I_2 invariant?
What is d?

Syntesizing the program computing n^3

```
[x,c,d := x+1, c+d, E'] I2 =
E' =3(x2+2x+1) + 3(x+1) +1 =
E' =d+6x+6

E' =d+e
e= 6x+6
```

invariant $I \wedge I_2 \wedge I_3$ $c=x^3 \wedge d=3x^2+3x+1 \wedge e=6x+6$ variant N−x

```
[x,c,d,e := x+1, c+d, d+e, E"] I3 =
E" =6(x+1)+6 =
E" =e+6
```

inizialization

```
[x,c,d,e := 0, C, D, E] I
```

Using the method bottom-up

Syntesizing the program computing n^3

Program

```
1. x,c,d,e := 0,0,1,6;
2. while x≠N do
3. x,c,d,e := x+1,c+d,d+e,e+6 done;
```

Bibliography

Lessons 1-4 Propositional Natural deduction (Huth Ryan chapter 1)
Lesson 5 OBDD (Huth Ryan chapter 6)
Lessons 6-7 Predicate Natural deduction (Huth Ryan chapter 2)
Lessons 8-10 Temporal logic (Huth Ryan chapter 3)
Lessons 11-13 Denotational semantics (dens.pdf)
Lesson 14 Operational semantics and languages
Lesson 15 Reactive systems
Lesson 16 Bisimulations (Milner: Communicating and mobile systems, intro2ccs.pdf)
Lesson 17 Hennessy Milner Logic CCS
Lesson 18 Hoare logic (Huth Ryan chapter 4)
Lesson 19 Synthesysing programs (Monin)