CS 3EA3: Lecture Notes - 10 February 2017

Scribed By: James Zhu

February 10, 2017

1 Announcements

- **Personalised Assignments:** If there is a topic which interests you *and/or* you are currently working on, **please ask Musa if you can do it for marks in this course!**
- **Quiz 3: Will be open-book,** but preferably not open laptop/phone.

  Know for Quiz:

  1. **Quiz questions borrowed from Sheet 5 and 6.**
  2. Lattices (builds from Sheet 4), Quantification, Textual Substitution (around 50% of quiz, please see Notes for 6 Feb and 8 Feb lectures)
  3. Also know the *ternary condition operators* from today's lecture.

  Tips:

  1. Print latest copy of **Theorem List** from course website.
  2. *Don't forget to read!*— when in doubt, look for a similar formula from Theorem List that you can use.
  3. Notable algorithm repeated in Notes and Sheets will show up.
  4. Don't worry too much about Galois Connections.
  5. Remember, Musa (and Curtis!) wants you to succeed, but be careful not to fall into the "open-book" fallacy: **open-book $\neq$ no need to study!**

2 Warm-up: Interesting Puzzle

Consider the following piece of code:

```
if a < 0 then P1
    else if b != 0 then P2
        else if a = b then P3
            else if a = 0 then P4
```

Question: When is P4 executed? Is it when:

1. a = 0
   OR
2. b != 0
   OR
3. b = 0
   OR
4. $1 \wedge 2 \wedge 3$, *ie.* false, *ie.* never
   OR
5. always

Answer:

Recall:

```
If b then t else f
```

- $b = true \implies t$
- $b = false \implies f$

Then, at P4, we should be able to *assert* (before b can happen):

$(b = 0) \wedge (0 \leq a) \wedge (a = 0) \wedge (a \neq b)$

This is $false$.

We can represent this in ***guarded notation:***

if
$\square a < 0 \rightarrow P1$
$\square (0 \leq a) \wedge (b \neq 0) \rightarrow P2$
$\square (b = 0) \wedge (a = 0) \rightarrow P3$
fi

Definition: So now we can specify a **general case** (to show why we are using guards in the first place).

if B then E else F fi
$\equiv$
if
$\square B \to E$
$\square \neg B \to F$
fi

Now, can we do this in C?

First try:

```
int x = if 0 <= y then 3 else x - 2;
```

We assume that x is defined.

WRONG!

Second try:

```
int x = if (0 <= y) {3;} else {x - 2;}
```

WRONG!

third try:

```
x = (0 < y) ? 3 : x - 2
```

This notation is called the **ternary conditional**.

General Case for Ternary Conditional

```
_ ? _ : _
```

3 Weakest Precondition

Lets define

wp S R

to be the condition **Q** such that ALL states which satisfies it will have **S** terminate afterwards with state satisfying **R**.

{Q} S {R}
$\equiv$

Q $\implies$ **wp S R**, where $\implies$ can be replaced with $\leq$

Look at Knaster-Tarski Lemma from Sheet 6 for fixed points.

Why: it can be used to define a **stable state** and derive a **rule for *loops*!**

4 Folding / Rolling Rule

```
while B do S
```

$\equiv$

```
if B then
    S; while B do S
else SKIP
```

```
f(x) = if B then S; X else SKIP fi
```

```
x = while B do S
```

$\equiv$

```
x = f(x)
```

Note: In C, ";": SKIP [*Empty statement*]

Let's be adventurous!

Take a look at this **DO** block:

do
$\Box B_1 \rightarrow S_1$
$\Box B_2 \rightarrow S_2$
$\vdots$
$\Box B_n \rightarrow S_n$
od

From this, we get:
BB $\equiv \exists i : 1...n \bullet B_i$

Therefore,

$H_0(R) = \neg BB \wedge R$
$H_k(R) = H_0(R) \vee$ wp "IF" $H_{k-1}(R)$

where

IF = if
$\Box B_1 \rightarrow S_1$
$\vdots$
$\Box B_n \rightarrow S_n$
fi

So now, we can define the *rule for loops*:

$$\text{wp Do R} = \exists k \bullet H_k(R)$$

- There is a bound k such that the loop will finish in **at most k steps** with condition **R**.

Note: This can be thought of solving: $\lim_{k \to \infty}$.