Abstract Interpretation: Exercises for day 3

[Note: you are welcome to program these exercises in groups of 2]

February 4, 2015

1. Implement the Parity analysis of the 3 counter machine in OCaml, including pretty printing of the analysis result, e.g., as an annotated program:[1]

```
1: inc y              {x:top, y:top, z:even}
2: zero y 1 else 1    {x:top, y:top, z:even}
3: stop               {x:bot, y:bot, z:bot}
```

2. Run the analysis on your program from handin 2 and discuss the result.

3. Which invariant does your implementation discover for line 3 of the following program:

```
1: inc z
2: zero z 3 else 4
3: inc y
4: dec z
5: stop
```

   How does that influence the predicted result at line 5? Can you do better (using the tools of abstract interpretation)?

4. Argue/prove that the calculated transfer function is actually monotone.

Bonus question

Why is

```
( <bot,bot,bot>. [pc' -> [x=0]#(S#(pc)) ] )
    U. ( <bot,bot,bot>. [pc'' -> [x<>0]#(S#(pc)) ] )
```

not (necessarily) the same as:

```
( <bot,bot,bot>. [pc' -> [x=0]#(S#(pc)) ][pc'' -> [x<>0]#(S#(pc)) ] )
```

in the `zero x pc' else pc''` case?

Can you write a program which reveals the difference?

[1] `Printf.printf` accepts optional width parameters which helps align things under each other. For example `'Printf.printf "%3i: %-20s" i s'` will reserve 3 characters for the integer `i` and 20 characters for the string `s`. Furthermore `i` will be right aligned, whereas `s` will be left aligned.