

Assignment #3: Reasoning About Programs — Progress

DUE: in class, Friday, April 20th.

1. **Progress** (6 points)

Consider the following program (the actions are labelled for your convenience):

```
assign
  a1 :    x := x + 1
  || a2 :    y := y + 1
  || a3 :    x ≠ y → z := z + 1
```

Are the following properties of this program? Justify your answer with a proof or counter-example, as appropriate.

- (a) **transient**.($x + y = k$)
- (b) **transient**.($x \leq k$)
- (c) $x = k \rightsquigarrow x > k$
- (d) **transient**.($z = k$)
- (e) $z = k \rightsquigarrow z \neq k$

2. **Termination** (6 points)

Given an array $A[0..N - 1]$ of distinct natural numbers, consider the following program:

```
assign
  ( || x : 0 ≤ x < N : 0 ≤ A[x] < N → A[x], A[A[x]] := A[A[x]], A[x] )
```

Does this program terminate? If so, prove your claim. If not, provide a counter-example.

3. **The Coffee Bean Problem** (11 points)

Consider a can of coffee beans. Each bean is either white or black. We are told the can is initially nonempty. Now consider the program that consists of a single action:

Choose two beans from the can; if they are the same color, toss them out and put in a white bean (there is a sufficient supply); if they are different colors, toss them out and put in a black bean (there is a sufficient supply).

This action is repeated.

- (a) (2 points) Let w be the number of white beans in the can and let b be the number of black beans. Write the program corresponding to the previous informal description. (Your program should have only two variables, w and b).

- (b) (2 points) Prove that $\{b \geq 0\}$ is an invariant of this program.
- (c) (3 points) Give a variant function for this program and use it to prove the program terminates.
- (d) (4 points) If the program terminates with $b = 1$, what must have been true of the beans in the can at the beginning of the computation? (Hint: find an invariant on w and/or b .)

4. **Metrics** (12 points)

In reviewing a proof of progress, you come across the following argument for why a particular (non-increasing) metric is guaranteed to decrease:

In every (non terminal) state an enabled action ($g \longrightarrow a$) exists whose effect is to strictly decrease the metric. Furthermore, if any other action falsifies this guard (g), it does so while decreasing the metric. Hence, the metric eventually decreases.

This argument is not correct. Why not? You may find it easiest to explain the problem by writing (or describing) a program that satisfies this rule and yet whose metric never decreases.