

# Concurrency Theory, Assignment Lecture 4

Krasimir Georgiev

September 17, 2015

We construct a program **Prog** in PGLEc with MPP for the function  $y = \text{Prog}(x1, x2) = x1 * (x2 + 1)$ . We first construct a program **MultiAccum** for the function  $y = \text{MultiAccum}(x1, x2, x3) = x1 * x2 + x3$ . The program **MultiAccum** reflects the following recursive definition of **MultiAccum**:

$$\text{MultiAccum}(x1, x2, x3) = \begin{cases} x3 & \text{if } x1 = 0 \\ \text{MultiAccum}(x1 - 1, x2, x2 + x3) & \text{otherwise.} \end{cases}$$

```
MultiAccum with comments:
L1; +x1==Z{;
  // If x1 = 0 return the accumulated value of x3.
  y=x3;
}{;
  // At this point x1 > 0, so x1 has a field p and
  // can be safely decremented.
  x1=x1.p;
  // Save x1 and x2 into temporaries.
  t1 = x1;
  t2 = x2;
  // Add t2 and x3 and put the result in x3.
  x1 = t2;
  x2 = x3;
  Add;
  x3 = y;
  // Restore the old values of x1 and x2 and continue.
  x1 = t1;
  x2 = t2;
  ##L1;
}
```

Note that  $\text{Prog}(x1, x2) = \text{MultAccum}(x1, x2 + 1, 0)$ .

```

Prog with comments:
// Increment x2.
x = x2;
S;
x2 = y;
// Initialize x3 = 0.
x3 = Z;
// Continue with MultAccum.
MultAccum;

```

The comments on the annotated versions of the programs constitute a short argument on the general correctness of the programs. Let's prove that the recursive definition of `MultAccum` is correct. We use induction on  $x1$ . Suppose  $x1 = 0$ . Then  $x1 * x2 + x3 = x3$ . Now suppose  $x1 > 0$  and  $\text{MultAccum}(x1 - 1, x2, x3) = (x1 - 1) * x2 + x3$  for all natural numbers  $x2$  and  $x3$ . Now  $\text{MultAccum}(x1, x2, x3) = \text{MultAccum}(x1 - 1, x2, x3 + x2) = (x1 - 1) * x2 + (x3 + x2) = x1 * x2 + x3$ .

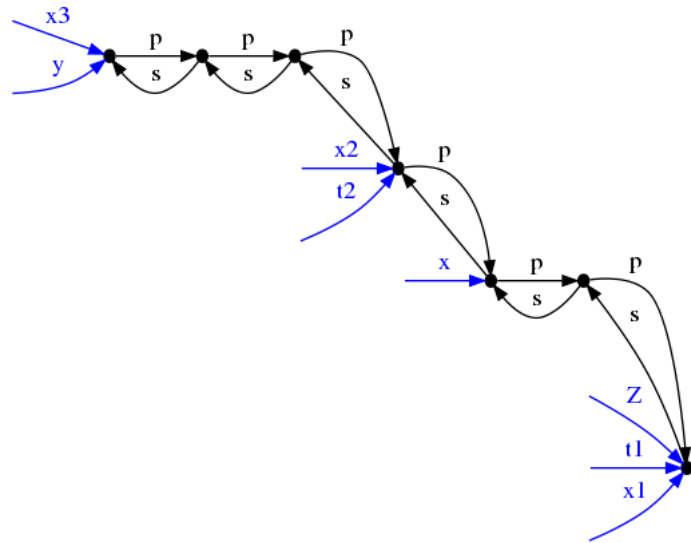
Following is a full listing of an evolution of `Prog` on  $x1 = x2 = 2$ .

```

Evolution for x1 = x2 = 2:
Z=new; y=Z;
x=y;
-x/s{; x.+s; y=new; y.+p; y.p=x; x.s=y; }{; y=x.s; };
x=y;
-x/s{; x.+s; y=new; y.+p; y.p=x; x.s=y; }{; y=x.s; };
x1=y; x2=y;

x=x2;
-x/s{; x.+s; y=new; y.+p; y.p=x; x.s=y; }{; y=x.s; };
x2=y;
x3=Z;
L1; +x1==Z{;
  y=x3;
}{;
  x1=x1.p;
  t1 = x1;
  t2 = x2;
  x1 = t2;
  x2 = x3;
  L0; +x1/p{;
    x1=x1.p;
    +x2/s{; x2=x2.s; }{;
      x2.+s; y=new; y.+p; y.p=x2; x2.s=y; x2=x2.s; };
    ##L0;
  }{; y=x2; };
  x3 = y;
  x1 = t1;
  x2 = t2;
  ##L1;
}

```



Here is the resulting fluid.

The source code of this assignment along with the programs can be found at <https://github.com/comco/concurrency-theory-assignments>.