

# SBf12, Assignment 3 Solutions

Kevin Oishi

October 27, 2012

## 1 Problem 1

### 1.1 Part (a)

Below is the gro code I used for part (a).

```
include gro

outfile := fopen("/tmp/1a_data.csv","w");

ROCK := signal(1,1);
PAPER := signal(1,1);
SCISSORS := signal(1,1);

thresh := 1;
program rpc(s) := {
  r := [state:=s];
  rfp :=0; gfp:=0; yfp:=0;

  r.state=ROCK : {rfp:=100*volume}
  r.state=PAPER : {gfp:=100*volume}
  r.state=SCISSORS : {yfp:=100*volume}

  true : {emit_signal(r.state,10*volume)}

  r.state = ROCK & get_signal(PAPER) > thresh : { die() }
  r.state = PAPER & get_signal(SCISSORS) > thresh : { die() }
  r.state = SCISSORS & get_signal(ROCK) > thresh : { die() }
};

program main() := {
  t := 0;
  s := 0;
  s2 := 0;
  n := 0;
```

```

rocks := 0;
papers := 0;
scissors := 0;
true : {s := s + dt, t := t + dt}
s > 1 : {
  rocks := sumlist maptocells if r.state=ROCK then 1 else 0 end end,
  papers := sumlist maptocells if r.state=PAPER then 1 else 0 end end,
  scissors := sumlist maptocells if r.state=SCISSORS then 1 else 0 end end,
  fprintf(outfile,t," ",rocks," ",papers," ",scissors,"\n")
}
s2 > 10 : {
  snapshot("/tmp/a4/1a/"<>tostring(n)<>".tif"),
  s2 := 0,
  n := n+1
}
};

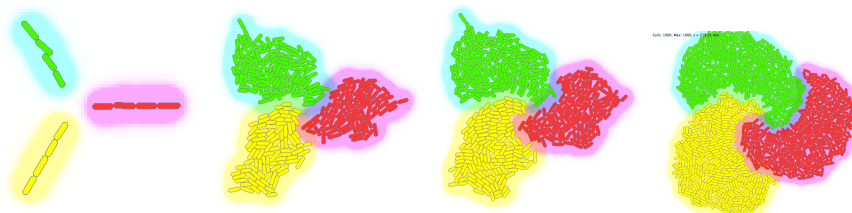
```

```

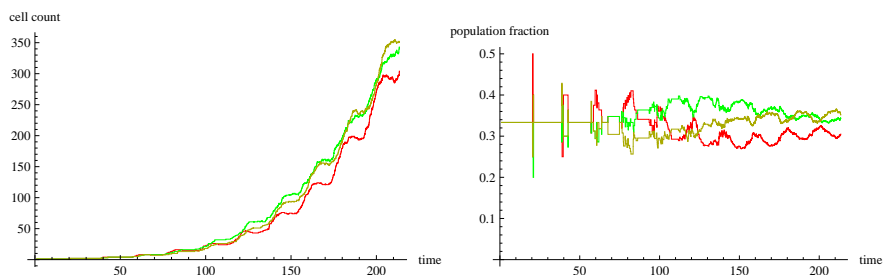
ecoli([x:=100*cos(0),y:=100*sin(0),theta:=0], program rpc(ROCK));
ecoli([x:=100*cos(2*pi/3),y:=-100*sin(2*pi/3),theta:=pi/3], program rpc(PAPER));
ecoli([x:=100*cos(4*pi/3),y:=-100*sin(4*pi/3),theta:=-pi/3], program rpc(SCISSORS));

```

Here are a few snapshots from a typical run of this program:



In these snapshots “rock” cells are red, “paper” cells are green, and “scissor” cells are yellow. The rock colony can invade the scissor colony, the scissor colony can invade the paper colony, and the paper colony can invade the rock colony. The resulting morphology is a spiral pattern with chirality related to the initial ordering of cell types. Plots of population and population fraction vs. time are shown below.



From these plots, you can see that the population of each colony type oscillates around an exponential increase as the various cell types annihilate each other.

## 1.2 Part (b)

Here's the code I used in part (b):

```
include gro
chemostat(true);

iter:=3;
outfile := fopen("/tmp/1b_data_10_<>toString(iter)<>".csv,"w");

ROCK := signal(1,1);
PAPER := signal(1,1);
SCISSORS := signal(1,1);

thresh := 1;
k := 0.001;
program rpc(s) := {
  r := [state:=s];
  rfp :=0; gfp:=0; yfp:=0;

  r.state=ROCK : {rfp:=100*volume, gfp:=0, yfp:=0}
  r.state=PAPER : {gfp:=100*volume, rfp:=0, yfp:=0}
  r.state=SCISSORS : {yfp:=100*volume, rfp:=0, gfp:=0}

  true : {emit_signal(r.state,10*volume)}

  rate(k) & r.state=ROCK : { r.state := PAPER }
  rate(k) & r.state=ROCK : { r.state := SCISSORS }
  rate(k) & r.state=PAPER : { r.state := ROCK }
  rate(k) & r.state=PAPER : { r.state := SCISSORS }
  rate(k) & r.state=SCISSORS : { r.state := ROCK }
  rate(k) & r.state=SCISSORS : { r.state := PAPER }

  r.state = ROCK & get_signal(PAPER) > thresh : { die() }
  r.state = PAPER & get_signal(SCISSORS) > thresh : { die() }
  r.state = SCISSORS & get_signal(ROCK) > thresh : { die() }
};

program main() := {
  t := 0;
  s := 0;
  s2 := 0;
  n := 0;
```

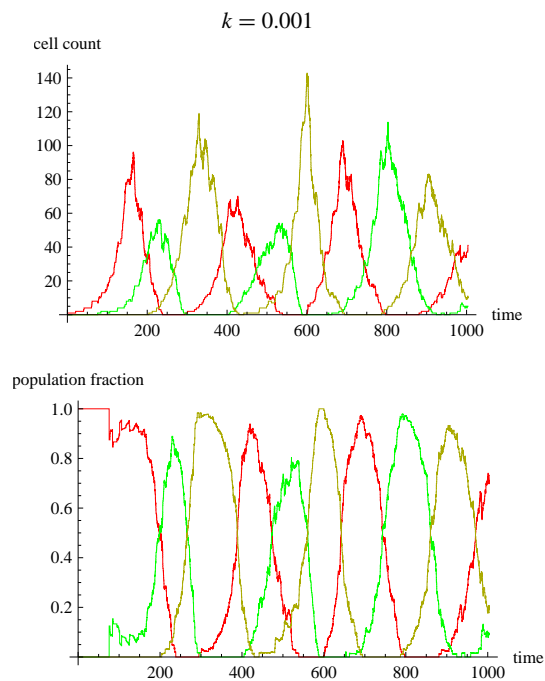
```

rocks := 0;
papers := 0;
scissors := 0;
true : {s := s + dt, s2 := s2+dt, t := t + dt}
s > 1 : {
  rocks := sumlist maptocells if r.state=ROCK then 1 else 0 end end,
  papers := sumlist maptocells if r.state=PAPER then 1 else 0 end end,
  scissors := sumlist maptocells if r.state=SCISSORS then 1 else 0 end end,
  fprintf(outfile,t," ",rocks," ",papers," ",scissors,"\n")
}
s2 > 1 : {
  snapshot("/tmp/1b_ "<>toString(iter)<>"_10_ "<>toString(n)<>".tif"),
  n := n+1,
  s2 := 0
}
};

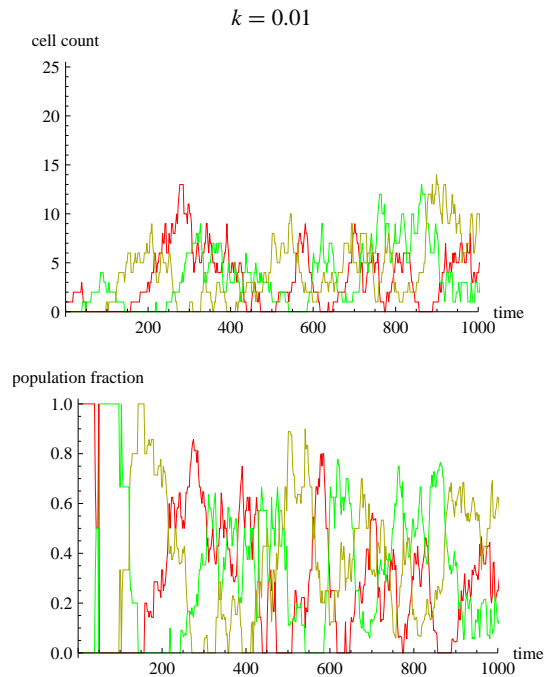
ecoli([], program rpc(ROCK));

```

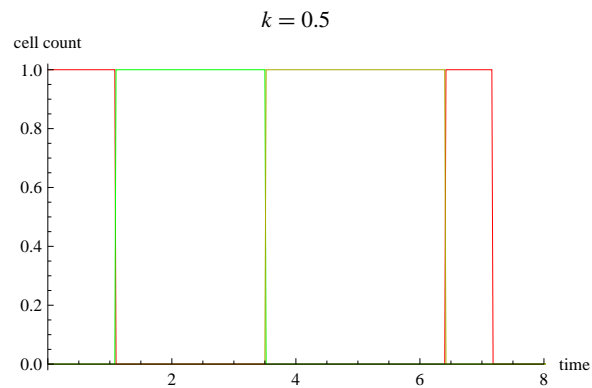
Different behaviors were obtained by varying the switching rate  $k$ . With a small value  $k = 0.001$  each cell type had an opportunity to sweep the entire population before a switch occurred. The result is a 3-phase oscillation moving from rock to paper to scissor cell types. This behavior is shown below with a plot of the cell count and population fraction vs. time for  $k = 0.001$ .



With a larger value of  $k = 0.01$ , switching occurred too frequently on average for a single cell type to sweep the colony. Small neighborhoods of cells within the chemostat switch from rock to paper to scissors, but there is no clear chemostat-scale synchronized behavior.



Finally, with a large value  $k = 0.5$ , switching occurs faster on average than cell division. Because I set the threshold to signal detection to be very low, cells can only survive by switching from rock to paper to scissors and back to rock. Admissible switching becomes increasingly unlikely as time increases. This is shown in the plot below where a single cell starts as rock, and successfully makes 3 switches before dying.



## 2 Problem 2

The gro code I used to implement this problem is shown below.

```
include gro
set("ecoli_growth_rate",0);

outfile := fopen("/tmp/2_00.csv","w");

ALPHA := signal(1,1);
DOX := signal(1,1);
GLU := signal(1,1);

fun hill x k . x/(k+x);

program cell3() := {
  rfp := 0;
  true : {
    rfp := 100*volume,
    emit_signal(ALPHA,10*volume*(1-hill (get_signal(DOX)) 0.05))
  }
};

program cell5() := {
  gfp := 0;
  yfp := 0;
  true : {
    yfp := 100*volume,
    emit_signal(ALPHA,10*volume*(1-hill (get_signal(GLU)) 0.05))
  }
};

program cell6() := {
  r := [s:=0, t:=0];
  gfp := 0;
  true : {r.s := r.s + dt, r.t := r.t + dt}
  r.s > 0.2 : {
    r.s := 0,
    fprintf(outfile,r.t,",",gfp/volume,"\n");
  }
  r.t > 50 : { stop() }

  rate(100*get_signal(ALPHA)) : { gfp := gfp + 1 }
  rate(gfp) : { gfp := gfp - 1 }
};
```

```

program main() := {
  true : {
    //set_signal(DOX,-20,20,100), set_signal(DOX,20,20,100),
    //set_signal(GLU,-20,-20,100), set_signal(GLU,20,-20,100),
    skip()
  }
};


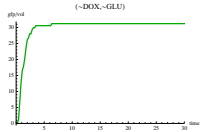

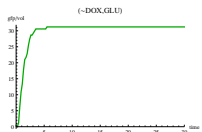

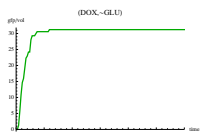


```

```

ecoli([x:=-20,theta:=pi/2], program cell3());
ecoli([x:=0,theta:=pi/2], program cell6());
ecoli([x:=20,theta:=pi/2], program cell5());

```

Below are figures showing snapshots from `gro`, output protein `gfp` vs. time, and the resulting truth table generated by exposing the system to different combinations of DOX and GLU. In these figures DOX responsive Cell 3 is on the left, GLU responsive Cell 5 is on the right, and the ALPHA responsive reporter Cell 6 is in the middle.

Inputs	Snapshot	<code>gfp</code> vs. time	truth table entry
-DOX, -GLU			$\frac{\text{DOX}}{F} \quad \frac{\text{GLU}}{F} \quad \left  \quad \frac{\text{gfp}}{T} \right.$
-DOX, GLU			$\frac{\text{DOX}}{F} \quad \frac{\text{GLU}}{T} \quad \left  \quad \frac{\text{gfp}}{T} \right.$
DOX, -GLU			$\frac{\text{DOX}}{T} \quad \frac{\text{GLU}}{F} \quad \left  \quad \frac{\text{gfp}}{T} \right.$
DOX, GLU			$\frac{\text{DOX}}{T} \quad \frac{\text{GLU}}{T} \quad \left  \quad \frac{\text{gfp}}{F} \right.$

### 3 Problem 3

This is a very cool problem! First off, it should be noted that Fischer, Lynch, and Paterson (1985) showed that in a natural model of asynchronous comput-

ing it is impossible to solve the consensus problem if even one process can fail. However extensions to this model, notably allowing randomization, makes consensus possible again. In writing up this solution I aimed for simplicity of code and logic, so this solution will not always solve the majority problem correctly but it does seem to work robustly for initial distributions that are far enough from 50/50.

```
include gro

set("ecoli_growth_rate",0);
outfile := fopen("/tmp/3_40_2.csv","w");

ZERO := signal(1,1);
ONE := signal(1,1);

g := [t:=0, zeros:=0, ones:=0];
fun max a b . if a>b then a else b end;

program rcp(s,h,k) := {
  r := [state:=s];
  rfp := 0;
  yfp := 0;

  r.state = ZERO : {
    rfp := 100*volume,
    yfp := 0,
    emit_signal(ZERO,50)
  }
  r.state = ONE : {
    rfp := 0,
    yfp := 100*volume ,
    emit_signal(ONE,50)
  }

  rate(k) & r.state=ZERO
    & get_signal(ONE) > max (get_signal(ZERO)-h) 0 : { r.state:=ONE }
  rate(k) & r.state=ONE
    & get_signal(ZERO) > max (get_signal(ONE)-h) 0 : { r.state:=ZERO }
  r.state=ZERO
    & rate(k*get_signal(ONE)/(get_signal(ONE)+get_signal(ZERO))) : {r.state := ONE}
  r.state=ONE
    & rate(k*get_signal(ZERO)/(get_signal(ONE)+get_signal(ZERO))) : {r.state := ZERO}
};

program main() := {
  s := 0;
```



```

x := {0,0};
true : {
  g.t := g.t+dt, s := s+dt ,
  x := { sumlist maptocells if r.state=ZERO then 1 else 0 end end,
        sumlist maptocells if r.state=ONE then 1 else 0 end end}
}
x != {g.zeros,g.ones} : {
  g.zeros := x[0];
  g.ones := x[1];
  fprintf(outfile,g.t,",",g.zeros,",",g.ones,"\n");
}
g.t <= dt : { print(g.t,",",g.zeros,",",g.ones,"\n") }
(x[0]=0 | x[1]=0) & x[0]+x[1]>1 : {stop()}
};

fun dist x . if rand(1000)/1000.0<=x then 0 else 1 end;
foreach q in cross (range 10) (range 10) do
  ecoli([x:=q[0]*25-125, y:=q[1]*25-125], program rcp(dist 0.4, 336.708, 0.1))
end;

```

The idea behind this algorithm is that every cell emits a signal (ONE or ZERO). At steady state, a cell can evaluate the ONE and ZERO signal concentration at its location, normalizing for its own signal emissions by subtracting a constant  $h$ . An individual cell uses these normalized values to make a decision about the majority in its local neighborhood, and changes state accordingly. In order to make it more likely that the signal concentrations are near steady state before such a decision is made, these evaluations occur at a small tunable rate  $k$ . To avoid situations where a cell grid gets “stuck” before coming to consensus, at some very small rate cells that sense a contradictory signal may change state spontaneously. Below I plotted a few typical trajectories generated with this algorithm, starting with approximately 60%, 70%, 80%, 90%, and 100% of cells in state ZERO.

