Probabilistic Model Checking - Practical 4

1

Listing 1: 4 Philosophers

```
// randomized dining philosophers [LR81]
// four philosophers

mdp

// formulae
// left fork free (left neighbour is philosopher 2)
formula lfree = (p2>=0 & p2<=4) | p2=6 | p2=10;
// right fork free (right neighbour is philosopher 3)
formula rfree = (p3>=0 & p3<=3) | p3=5 | p3=7 | p3=11;

module phil1

   p1: [0..11];

   [] p1=0 -> (p1'=1); // trying
   [] p1=1 -> 0.5 : (p1'=2) + 0.5 : (p1'=3); // draw randomly
   [] p1=2 & lfree -> (p1'=4); // pick up left
   [] p1=3 & rfree -> (p1'=5); // pick up right
   [] p1=4 & rfree -> (p1'=8); // pick up right (got left)
   [] p1=4 & !rfree -> (p1'=6); // right not free (got left)
   [] p1=5 & lfree -> (p1'=8); // pick up left (got right)
   [] p1=5 & !lfree -> (p1'=7); // left not free (got right)
   [] p1=6 -> (p1'=1); // put down left
   [] p1=7 -> (p1'=1); // put down right
   [] p1=8 -> (p1'=9); // move to eating (got forks)
```

```
   [] p1=9 -> (p1'=10); // finished eating and put down left
   [] p1=9 -> (p1'=11); // finished eating and put down right
   [] p1=10 -> (p1'=0); // put down right and return to think
   [] p1=11 -> (p1'=0); // put down left and return to think

endmodule

// construct further modules through renaming
module phil2 = phil1 [ p1=p2, p2=p4, p3=p1 ] endmodule
module phil3 = phil1 [ p1=p3, p2=p1, p3=p4 ] endmodule
module phil4 = phil1 [ p1=p4, p2=p3, p3=p2 ] endmodule

// labels

// a philosopher is hungry
label "hungry" =
    ((p1>0)&(p1<8))|((p2>0)&(p2<8))|((p3>0)&(p3<8))|((p4>0)&(p4<8));
 // a philosopher is eating
label "eat" =
    ((p1>=8)&(p1<=9))|((p2>=8)&(p2<=9))|((p3>=8)&(p3<=9))|((p4>=8)&(p4<=9));
// philosopher 1 is hungry
label "hungry1" = (p1>0)&(p1<8);
// philosopher 2 is hungry
label "hungry2" = (p2>0)&(p2<8);
// philosopher 3 is hungry
label "hungry3" = (p3>0)&(p3<8);
// philosopher 4 is hungry
label "hungry4" = (p4>0)&(p4<8);
 // philosopher 1 is eating
label "eat1" = (p1>=8)&(p1<=9);
 // philosopher 2 is eating
label "eat2" = (p2>=8)&(p2<=9);
 // philosopher 3 is eating
label "eat3" = (p3>=8)&(p3<=9);
 // philosopher 4 is eating
label "eat4" = (p4>=8)&(p4<=9);
```

Listing 2: 5 Philosophers

```
// randomized dining philosophers [LR81]
// five philosophers

mdp

// formulae
// left fork free (left neighbour is philosopher 2)
formula lfree = (p2>=0 & p2<=4) | p2=6 | p2=10;
// right fork free (right neighbour is philosopher 3)
formula rfree = (p3>=0 & p3<=3) | p3=5 | p3=7 | p3=11;

module phil1

   p1: [0..11];

   [] p1=0 -> (p1'=1); // trying
   [] p1=1 -> 0.5 : (p1'=2) + 0.5 : (p1'=3); // draw randomly
   [] p1=2 & lfree -> (p1'=4); // pick up left
   [] p1=3 & rfree -> (p1'=5); // pick up right
   [] p1=4 & rfree -> (p1'=8); // pick up right (got left)
   [] p1=4 & !rfree -> (p1'=6); // right not free (got left)
   [] p1=5 & lfree -> (p1'=8); // pick up left (got right)
   [] p1=5 & !lfree -> (p1'=7); // left not free (got right)
   [] p1=6 -> (p1'=1); // put down left
   [] p1=7 -> (p1'=1); // put down right
   [] p1=8 -> (p1'=9); // move to eating (got forks)
   [] p1=9 -> (p1'=10); // finished eating and put down left
   [] p1=9 -> (p1'=11); // finished eating and put down right
   [] p1=10 -> (p1'=0); // put down right and return to think
   [] p1=11 -> (p1'=0); // put down left and return to think

endmodule

// construct further modules through renaming
module phil2 = phil1 [ p1=p2, p2=p4, p3=p1 ] endmodule
module phil3 = phil1 [ p1=p3, p2=p1, p3=p5 ] endmodule
module phil4 = phil1 [ p1=p4, p2=p5, p3=p2 ] endmodule
module phil5 = phil1 [ p1=p5, p2=p3, p3=p4 ] endmodule
```

```
// labels

// a philosopher is hungry
label "hungry" =
    ((p1>0)&(p1<8))|((p2>0)&(p2<8))|((p3>0)&(p3<8))|((p4>0)&(p4<8))|((p5>0)&(p5<8));
 // a philosopher is eating
label "eat" =
    ((p1>=8)&(p1<=9))|((p2>=8)&(p2<=9))|((p3>=8)&(p3<=9))|((p4>=8)&(p4<=9))|((p5>=8)&(p5
// philosopher 1 is hungry
label "hungry1" = (p1>0)&(p1<8);
// philosopher 2 is hungry
label "hungry2" = (p2>0)&(p2<8);
// philosopher 3 is hungry
label "hungry3" = (p3>0)&(p3<8);
// philosopher 4 is hungry
label "hungry4" = (p4>0)&(p4<8);
 // philosopher 5 is hungry
label "hungry5" = (p5>0)&(p5<8);
 // philosopher 1 is eating
label "eat1" = (p1>=8)&(p1<=9);
 // philosopher 2 is eating
label "eat2" = (p2>=8)&(p2<=9);
 // philosopher 3 is eating
label "eat3" = (p3>=8)&(p3<=9);
 // philosopher 4 is eating
label "eat4" = (p4>=8)&(p4<=9);
 // philosopher5 is eating
label "eat5" = (p5>=8)&(p5<=9);
```

2

Figure 1: E[Lost requests for T$\leq$ 20]