Experiments

Marco Tinacci

October 23, 2015

Robot arena

With this example we want to give an idea of how to translate a standard fully observable model in PRISM [1] to a partially observable one, introducing information about the observation function.

Robot module

The module define in Listing 1 describe a deterministic robot that can move in the four directions or stand still. `init` and `actions` sections describe respectively states and actions of the robot. We introduce an `observations` section that describe how sensors react to the environment.

```
module robot
        // init
        x0 : [0..D-1] init ix0;
        y0 : [0..D-1] init iy0;

        // observations
        [obs_n] (!see_h & see_n & ...) -> true;
        ...
        [obs_ne] (!see_h & see_n & see_e & ...) -> true;
        ...

        // actions
        [act_n] true
                -> (x0'=max(x0-1,0));
        ...
endmodule
```

Listing 1: Robot module

Random robot module

The module defined in Listing 2 define an environmental element that is a robot that follows a random walk behavior. This module does not require any modification with respect to the fully observable model. The module define in To replicate the random robot we use the Listing 3, in this case we consider just two random robots but the replica module can be used any number of times.

```
module RR1
        x1 : [0..D-1] init ix1;
        y1 : [0..D-1] init iy1;

        [s1]    true ->
                1/5 : (x1'=max(x1-1,0)) +
                1/5 : (y1'=min(y1+1,D-1)) +
                1/5 : (y1'=max(y1-1,0)) +
                1/5 : (x1'=min(x1+1,D-1)) +
                1/5 : true;
endmodule
```

Listing 2: Random robot module

```
module RR2 = RR1
        [ x1=x2, y1=y2, s1=s2, ix1=ix2, iy1=iy2 ]
endmodule
```

Listing 3: Random robot replica

Synchronization module

To implement the desired synchronization behavior we define a module dedicated to communication and timing between the main agent and the environment (Listing 4), in this case the robot and two other random robots.

From the code we can see that this module force an iterating sequence of actions composed in the following way:

1. robot observation
2. robot action
3. first random robot action
4. second random robot action

The scheme can be easily generalized to an arbitrary number of environmental elements.

```
module Synchronizer

        // turns: robot, RR1, RR2
        turn : [0..2] init 0;
        obs: bool init true;
        // [<observation set>]
        [obs_n] (turn=0 & obs=true)
                -> (obs'=false);
        ...
        [obs_ne]          (turn=0 & obs=true)
                -> (obs'=false);
        ...

        // [<action set>]
        [act_n] (turn=0 & obs=false)
                -> (turn'=1) & (obs'=true);
        ...

        // [<coordination set>]
        [s1]      turn=1 -> (turn'=2);
        [s2]      turn=2 -> (turn'=0);

endmodule
```

Listing 4: Synchronization module

Results

We considered the two LTL formulae:

- $\varphi^k_{avoid} = G^{\leq k}\neg\text{collision}$
- $\varphi^k_{track} = F^{\leq k}\text{vision}$

and we analyzed the minimum probability starting from an arbitrary initial configuration. Labels *collision* and *vision* represent respectively states in which the main robot's position is the same as the position of another robot, and states in which the main robot can perceive the presence of another robot close to it. We also analyze a third formula that consider the reachability within k steps of *vision* and not *collision*, but the results are identical to φ_{track}.

Results are shown in Figures 1 and 2 where we plot the probabilities step by step (fine grain in blue) and the probabilities filtered by the meaningful times, that is, the initial step of every synchronization sequence (coarse grain in red). This filtering has to be done to obtain the result we really want from the beginning, that is the minimum probability sequence considering the synchronization as an atomic operation.

We tried to stress more the dimension of the generated model, defining the robot arena of different sizes ($D = 3, 4, 5$) and different number of robots ($N = 3, 4, 5$). In Figures 3 and 4 we show the minimum probabilities of collision-avoidance and tracking inside an arena containing other 4 robots.

This means that also the formulae need to consider the synchronization mechanism employed and described by Listing 4 and thus the number of environmental modules considered in the PRISM model.

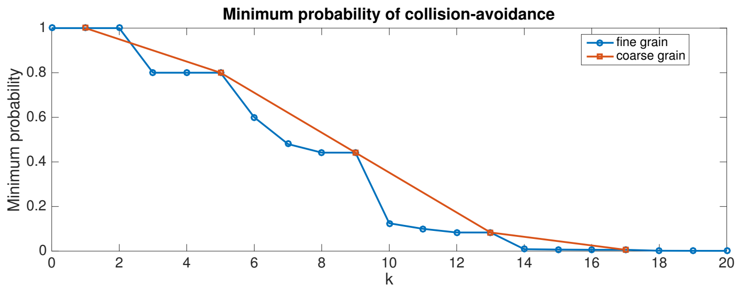

Figure 1: $\mathcal{P}_{min}\varphi^k_{avoid}$ with 3 robots in a 3×3 arena

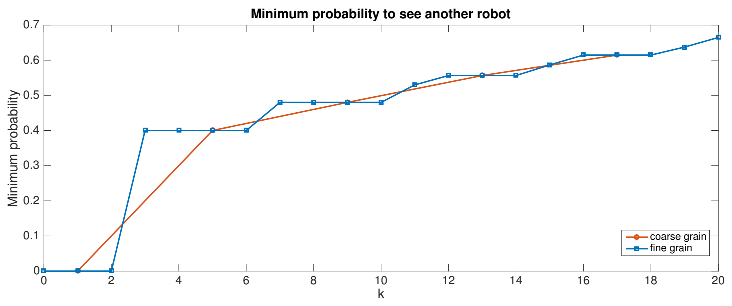

Figure 2: $\mathcal{P}_{min}\varphi^k_{track}$ with 3 robots in a 3×3 arena

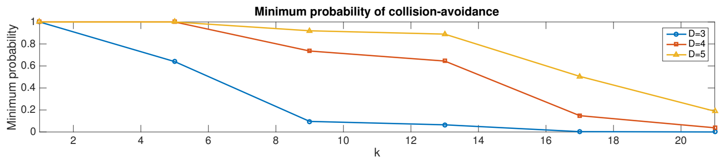

Figure 3: $\mathcal{P}_{min}\varphi^k_{avoid}$ with 5 robots in arenas of different sizes

Considerations

Extending the fully observable PRISM model into a partial observable one let us exploit its optimized engine, spareing a lot of computations with respect to the generation of the composed model.

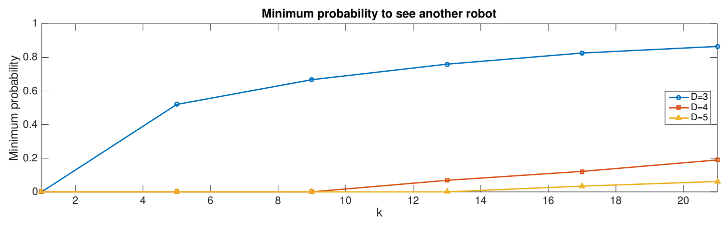

Figure 4: $\mathcal{P}_{min}\varphi^{k}_{track}$ with 5 robots in arenas of different sizes

References

[1] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA. Proceedings*, pages 585–591, 2011.