Question 2) Path to download code/binaries:

<https://drive.google.com/drive/folders/0B1IBCwXEnb-YUkd2OGhFeThDSkk?usp=sharing>

Turn/Ball still not seen

Turn,Dash/Ball still far

StateReadyToStrike

(S3)

StateLookingForOpponentGoal

(S2)

Turn/Ball too far

StateBallTooFar

(S1)

StateLookingForBall

(S0)

Turn,Dash/Ball disappeared

Turn/Ball disappeared

Turn,Dash/ball close

Turn/Ball seen close

Kick/Ball disappeared

Kick/Ball too far

Turn/can’t see goal

Kick/Ball still here but can’t see goal

Turn/see goal

Kick/Ball still here

The state-based agent was designed according to the state machine above. It has 4 states: “StateLookingForBall”: when the agent doesn’t know where the ball is, “StateBallTooFar”: when the agent knows where the ball is but is too far from it to kick it, “StateLookingForOpponentGoal” when the agent is close enough to the ball to kick it but doesn’t know where the opponent’s goal is and “StateReadyToStrike”: when the agent is close enough to the ball to kick it and knows where the opponent’s goal is.

The code was implemented by modifying mainly the “Run” method of the “Brain” class of the “Krislet“ program. The state machine is hardcoded and implemented using “State pattern”. The classes representing the states have the same names as the states themselves, the context class is named “StateContext” and the state interface implemented by the state classes is named “SoccerPlayerState” and all these classes are in the file “StateContext.java”. The way the code works is as follows: the initial state (“StateLookingForBall”) is set at the beginning of the “Run” method in the “Brain” class. Then while the game is on, the agent receives the environment updates at given intervals and uses them to update its current state and depending on the updated state pick the appropriate action to perform. The appropriate action to perform may vary not only based on a given agent state but also on the environment state as well e.g. let’s say the agent is in the “StateBallTooFar” state, depending on whether he has the correct direction or not he may either turn to have the correct direction or dash towards the ball. Also, due the fact that the environment is dynamic and non-deterministic the outcome of actions isn’t determined e.g. if an agent dashes towards the ball, on the next update the ball may be inaccessible to him if it was for example kicked away by another player in the meantime.

The given code should produce a behavior very similar to the original “Krislet” program. To run the code, launch the server and the monitor, then go to the binaries subfolder in the path given above and execute the TeamStart.bat file just as in the provided “robocupDemo”.

Examples of possible and impossible run(s) from the state machine:

* S0 -> turn -> S1 -> dash -> S2 -> turn -> S3 -> kick …
* S0 -> turn -> S2 -> turn -> S3 -> kick -> S1 -> dash -> S2 …
* S0 -> turn -> S3 … : this run is impossible as there is no transition from S0 to S3 in the state machine above.