**T-79.5102** **Autumn 2006**
**Special Course in Computational Logic**
**Tutorial 1**

**1.** The four tables given below summarize control commands and sensor information available in the RoboCup soccer server protocol. Consider implementing the following functionalities using the protocol.

(a) A ball tracking mechanism (try to find and approach the ball).

(b) Kicking the ball towards the other team's goal.

For more information about the student project, please consult `http://www.tcs.hut.fi/Studies/T-79.5102/2006AUT/project.shtml`

| **Body Commands** |
|---|
| **(turn *Moment*)**<br>The ***Moment*** is in degrees from −180 to 180. This command will turn the player's body direction ***Moment*** degrees relative to the current direction. |
| **(dash *Power*)**<br>This command accelerates the player in the direction of its body (not direction of the current speed). The ***Power*** is between ***minpower*** (−100) and ***maxpower*** (100). |
| **(kick *Power Direction*)**<br>Accelerates the ball with the given ***Power*** in the given ***Direction***. The direction is relative to the ***Direction*** of the body of the player and the power is again between ***minpower*** and ***maxparam***. |
| **(catch *Direction*)**<br>This is a goalie specific command. It tries to catch the ball in the given ***Direction*** relative to its body direction. If the catch is successful the ball will be in the goalie's hand until kicked away. |
| **(move *X Y*)**<br>This command can be executed only before kick off and after a goal. It moves the player to the exact position of ***X*** (between −54 and 54) and ***Y*** (between −32 and 32) in one simulation cycle. This is useful for before kick off arrangements. |
| **(turn_neck *Angle*)**<br>Rotate the player's neck with the given ***Angle*** relative to previous one. Note that the resulting neck angle will be between ***minneckang*** (−90) and ***maxneckang*** (90) relative to the player's body direction. |

| **Communication Commands** |
|---|
| **(say *Message*)**<br>This command broadcasts the ***Message*** through the field, and any player near enough (specified with ***audio_cut_dist*** which defaults to 50 meters), with enough hearing capacity will hear the ***Message***. The message is a string of valid characters. |

| **Miscellaneous Commands** |
|---|
| **(sense_body)**<br>Requests the server to send body information sensed by the player. Note the server sends body information every cycle for server versions 6.00 or later. |
| **(score)**<br>Request the server to send score information. The server's reply will be in the form **(score *Time OurScore OpponentScore*)**. |
| **(change_view *Width Quality*)**<br>Changes the view parameters of the player. ***Width*** is one of **narrow**, **normal** or **wide** and ***Quality*** is one of **high** or **low**. The amount and detail of the information returned by the visual sensor depends on the width of the view and the quality. But note that the frequency of sending information also depends on these parameters (e.g. if you change the quality from high to low, the frequency doubles, and the time between two see sensors will be cut to half). |

| **Sensor Information** |
|---|
| **(see *Time ObjInfo* ...)**<br>where ***ObjInfo*** is of the form<br>**(*ObjName Distance Direction* [*DistChange DirChange* [*BodyFacingDir HeadFacingDir*]])** or **(*ObjName Direction*)**<br>Note that the amount of information returned for each object depends on its distance. The more distant the object is the less information you get. The parameter ***ObjName*** is in one of the following forms: **(p [*TeamName* [*Unum*]])**, **(b)**, **(f *FlagInfo*)**, or **(g *Side*)**. Here **p** stands for player, **b** stands for ball, **f** stands for flag, and **g** stands for goal. ***Side*** is either **l** (for left) or **r** (for right). |
| **(hear *Time Sender Message*)**<br>The ***sender*** is one of the following: **self** (yourself), **referee** (the referee of the game), **online_coach_l**, or **online_coach_r**. When the sender is some other player, the relative direction of the sender is returned instead. |
| **(sense_body *Time* (view_mode { high \| low } { narrow \| normal \| wide }) (stamina *Stamina Effort*) (speed *Speed Angle*) (head_angle *Angle*) (kick *Count*) (dash *Count*) (turn *Count*) (say *Count*) (turn_neck *Count*) (catch *Count*) (move *Count*) (change_view *Count*))**<br>Body sensor returns the values of all state variables of the player such as remaining stamina, view mode and the speed of the player at the beginning of each cycle. The last eight parameters are counters of the commands received by the server. Use these counters to keep track of lost or delayed messages. |