

# Practical Session GOAL

## Agent for Tower Environment

In this practical assignment, you will program a GOAL agent that can build towers in the tower environment as first proposed in N. J. Nilsson. Teleo-reactive programs and the triple-tower architecture. Electronic Transactions on Artificial Intelligence, 5:99–110, 2001 (see also [http://www.robotics.stanford.edu/users/nilsson/trweb/TRTower/ TRTower links.html](http://www.robotics.stanford.edu/users/nilsson/trweb/TRTower/TRTower%20links.html)). In this environment the agent controls a gripper that can pick up and put down blocks. In addition, the environment comes with a GUI that allows to move the blocks while the agent is trying to build a tower. This introduces additional dynamics that the agent has to be able to handle.

In this assignment you will be guided step-by-step through the development of a GOAL agent program to control the gripper.

### *Install the Goal IDE and load the mas file*

Before we can start, you have to install the GOAL agent programming language and Integrated Development Environment on the machine. You need to perform the following steps:


1. Open a browser.
2. Go to <http://mmi.tudelft.nl/trac/goal/wiki/Releases>.
3. Download the latest GOAL IDE installer.
4. Open the installer by double-clicking on the downloaded jar file.
5. Follow the instructions for installing GOAL (click *next*, *next*, select 'I accept ...', *next*, *next*, *ok*, *next*, *next*, select 'Create additional shortcuts ...', *next*, *done*).

You have now installed the GOAL IDE and should be able to locate the Goal icon on your desktop. Double click this icon to start the GOAL IDE.

6. Go to file > open.
7. Go to the location where you stored the files 'DALTowerworld2.mas2g' and 'DALTowerbuilder2.goal'.
8. Select and open the file 'DALTowerworld2.mas2g'.


## Part 1: Percept Processing

We will first run the agent program to investigate what it does.

- Press run  in the IDE. You will see that an agent has been created named *towerBuilder*. The agent starts in paused mode and does not perform any actions yet.
- Double click on the label *towerBuilder* in the Process Overview window. A new window will open that allows you to inspect the mind of the *towerBuilder* agent, which is called an *introspector*. Initially, you will see the current beliefs of *towerBuilder*. Inspect the various mental state components by clicking on the tabs.
- Set up the tower environment by clicking three times on the *New Block* button. Blocks a, b and c appear on the table.

Now it is time to explore what *towerBuilder* does.


- Select the MAS node labeled *DALTowerworld2.mas2g* in the Process Overview window and press

the step button  two times (if you select the *towerBuilder* node, stepping does not work).

- Inspect the *Percepts* tab. Confirm that the percepts reflect the current state of the tower environment.
- Inspect the *Beliefs* tab. Press the *step* button another *three* times. Observe how the beliefs change.
- Return to the GOAL agent program by clicking the *Edit* button in the top right corner of the IDE. Double click the file *DALTowerbuilder2.goal*, as shown in the *Files* window on the left of the IDE. The GOAL program opens.
- Scroll down to the **event module**. In the **program** section of the **event module** you find two action rules for processing *block* percepts. Study these rules and explain the changes in the belief base as observed previously. Note that these rules are of the form **forall** <condition> **then** <action>, which means that the rule is executed for all instantiations of the condition.

### Assignment:

1. Write two action rules, analogously to the rules for processing block percepts, to process percepts of the form *percept(on(X,Y))*. These percepts represent that the agent perceives that a block X is on top of a block Y. Make sure that atoms of the form *on(X,Y)* are inserted into and deleted from the belief base when processing these percepts. Save your code and check the *Parse Info* tab at the bottom of the IDE to detect possible syntax errors in your code.

If the code does not contain parse errors (anymore), test it by doing the following. Return to the running program by pressing the Debug button at the top right of the IDE. Kill the currently running MAS by clicking on the node *DALTowerworld2.mas2g* and pressing the  button in the main tool bar just below the menu bar. Save your program, press run, and step the agent system several times until you see the beliefs changing as desired.

2. Write two action rules, again analogously to the existing ones, for processing percepts of the form *percept(holding(X))*. These percepts represent that the agent is holding a block X. Make sure that atoms of the form *holding(X)* are inserted into and deleted from the belief base when processing these percepts.

## Part 2: Action Specification

Now that we have specified how the agent should process percepts, it is time to specify the actions that the agent can execute.

- Scroll to the **actionspec** section as specified in the **init module**. There you find the action specification for the *pickup(X)* action, which is used to tell the gripper to pick up block X. Study the action specification's pre- and postconditions. Confirm that the precondition expresses when the agent is (physically) able to pick up a block.

Observe that the postcondition is *true*, because *pickup* is a *durative* action. The action takes time, and therefore the result of the action, namely that the agent is holding a block, should not be inserted into the belief base by means of the action's postcondition, but rather through processing of percepts as specified previously.

### Assignment:


Write an action specification for the action *putdown(X,Y)*, which is used to tell the gripper to put down a block X onto a block Y. The action specification's precondition should express that the agent can only put a block X down onto Y if it is holding X and there is no block on top of Y.

## Part 3: Making a Constructive Move

- Kill and restart the MAS. Double click on the *towerBuilder* agent to inspect its mental state, and press the step button several times until the relevant *block* and *on* atoms have been inserted into the belief base. Click on the *Goals* tab and press the step button several times until the goal *holding(b)* appears.
- Return to the GOAL agent program by clicking the *Edit* button. Study the action rule "if not(goal(holding(X))), not(bel(holding(Y))) then adoptgoal." occurring in the **program** section of the **event module**. This action rule calls the user-defined **module adoptgoal**, if the agent does not have the goal to hold a block and is not currently holding a block.
- Scroll down to the **module adoptgoal**. In the **program** section of that module there is a rule specifying when the agent should adopt the goal of holding a block. The condition of the rule is specified by means of a **macro**. Macros are used to increase readability of the agent program. The definition of the macro "constructiveMove(X, Y)" can be found just above the **main module**. Study the definition of this macro to understand when the agent adopts the goal of holding a block. Confirm that the condition specified by the macro was true for X=b (and which substitution for Y?), before the agent adopted the goal *holding(b)*.
- Return to the running MAS and press the step button several times more until you see the gripper move towards block b.
- Return to the GOAL agent program by clicking the *Edit* button. Study the action rule in the **program** section of the **main module**. Can you explain why the gripper moved to pick up block b?
- Return to the running MAS and click on the *Goals* tab. Press the step button several times until the goal *holding(b)* disappears. Can you explain why this happens? Which atom in the belief base is the cause of the goal being dropped?
- Press the step button several times more. As can be observed from the *towerBuilder* tab at the bottom of the IDE, no actions are enabled at this point in time.

### Assignment:

The agent should now put block b (which it is currently holding) down on top of block c. To let the agent do this, specify an action rule in the **main module** that selects the *putDown(X, Y)* action if the agent is holding block X and can make a constructive move to put X on top of Y.

Kill and restart the MAS to test your agent program. You can now press run  instead of stepping the agent. After putting block b on top of block c, you should see the agent picking up block a and putting it on b.

- Pause the MAS and inspect the *Goals* tab. Can you explain why the goal base is empty? Which atoms in the belief base are the cause of the goal *on(a,b)*, *on(b,c)*, *on(c,table)* . being dropped?

## Part 4: Moving Obstructing Blocks

- Kill and restart the MAS. After pressing the *New Block* button in the Tower Environment three times, move block a on top of block b through drag and drop. Step the agent while observing the *towerBuilder* tab at the bottom of the IDE. As you can see, the agent keeps entering and exiting the **adoptgoal module** without doing something. The reason for exiting the module without doing something is that the action rule in the **adoptgoal module** cannot be applied. Can you explain why, by looking at the definition of the macro "constructiveMove(X, Y)"?

### Assignment:

1. Define a macro "obstructingBlock(X)" (analogously to the macro constructiveMove), which should specify that block X is obstructing if: it prevents moving a block Y onto another block Z to achieve the goal that on(Y,Z), either because block X is above the block to which Y should be moved, or because block X is above Y which prevents moving it. Disjunction can be specified using semicolon. For example, "bel(p(X);q(Y))" specifies that the agent believes p(X) or q(Y). Insert the macro at the top of the **program** section of the **module adoptgoal**. Use the "above(X,Y)" predicate as defined in the **knowledge base** of the **module init**, to specify that block X is above block Y.
2. Add an action rule to the **program** section of the **module adoptgoal** that specifies that the agent should adopt the goal of holding block X if X is an obstructing block. Insert it below the rule for making constructive moves, to make sure the agent first makes constructive moves if it can (the default rule evaluation order of user-defined modules is *linear*, i.e., the rule that can be applied fires).

Test your program by killing and restarting the MAS, creating three blocks and stacking a on top of b. Now step the agent and inspect the *Goals* tab. The agent should adopt the goal to hold block a (as this block is obstructing) and move the gripper to pick up the block. Once the block is picked up, the goal to hold block a is dropped again.

- Continue stepping the agent while inspecting the *towerBuilder* tab at the bottom of the IDE. Observe that the agent keeps entering the **main module** and exiting again.

### Assignment:

1. Add an action rule to the **program** section of the **main module** that specifies that the agent should move a block that it is holding to the table, if it cannot perform a constructive move for the block that it is holding.

Test your program by killing and restarting the MAS, creating three blocks and stacking a on top of b. Press the run button. The agent should now put a onto the table, b onto c, and a onto b.

## Part 5: Dealing with Dynamics

- Kill and restart the MAS. Create three blocks (placed on the table). Press run. The agent will move the gripper with the aim of picking up block b. While the gripper is moving, place block a on top of block b. The speed with which the gripper is moving can be adjusted using the slider at the bottom right of the Tower Environment GUI.
- Inspect the *Goals* tab. Observe that the agent has the goal of holding block b. In the current state of the environment, the agent cannot achieve this goal directly. Rather, it should adopt the goal of holding block a, so as to clear block b.

### Assignment:

Add an action rule to the **program** section of the **event module** to specify that if the agent has the goal of holding a block while this block is not clear, it should drop the goal of holding the block. Insert the rule immediately below the rules for processing percepts.

Test your program by killing and restarting the MAS, and recreating the situation as described above. Once block a is put on top of block b, the agent should now drop the goal of holding block b. Which rule is then applied to adopt the goal of holding block a?

- Kill and restart the MAS. Create three blocks (placed on the table). Press run. The agent will move

the gripper with the goal of holding block b. While the gripper is moving, place block b on top of block c. The agent should now pick up block a, since block b is already in the desired position. Observe what happens instead.

**Assignment:**

1. Add another action rule to the **program** section of the **event module** to specify that if the agent has the goal of holding a block while this block is already in the desired position, it should drop the goal of holding the block. Hint: use the operator **goal-a** to express that a (part of a) goal has already been achieved. Test your program.
2. Can you think of other situations, resulting from the user moving blocks while the agent is building towers, in which the goal of holding a block should be dropped? Hint: blocks can also be moved in and out of the gripper while the gripper is moving.