Setting Up & Using the Goal Structure

Table of Contents

Setting Up the Goal Structure

In this section we will discuss how to set up and use the Goal Structure. Broadly speaking, the goal structure can be thought of as being a "container" within an agent that holds the agent's goals (in the form of `GoalChunk` world objects). Technically speaking, it is located within the top level of the MS. However, all interaction with the goal structure (from the simulating environment) is performed directly via the `Agent` class. For instance, we can view the contents of the goal structure by calling the `GetInternals` method. The code below demonstrates how we might accomplish this for our agent, `John`:

```
//Gets all of the items in the goal structure
IEnumerable<GoalChunk> gsContents =
    (IEnumerable<GoalChunk>)John.GetInternals
    (Agent.InternalWorldObjectContainers.GOAL_STRUCTURE);

//Gets the current goal
GoalChunk currentGoal = John.CurrentGoal;
```

Like actions, goals are represented as chunks (i.e., using the `GoalChunk` class) and are initialized through the `World` singleton object:

```
GoalChunk g = World.NewGoalChunk();
```

There are also two parameters that you can set in order to "tune" the behavior of the goal structure. They are located in the parameters class of the `MotivationalSubsystem` and can be used to specify:

1. The behavior of the goal structure (i.e., does it behave like a list or a stack)
2. How to set the activation for the current goal (i.e., use the actual activation specified when the goal was set, or use the full activation for the goal)

Below is an example of how to set these parameters (locally) for our agent, `John`:

```
John.MS.Parameters.CURRENT_GOAL_ACTIVATION_OPTION =
```

```
    MotivationalSubsystem.CurrentGoalActivationOptions.FULL;

John.MS.Parameters.GOAL_STRUCTURE_BEHAVIOR_OPTION =
    MotivationalSubsystem.GoalStructureBehaviorOptions.STACK;
```

The following lines of code demonstrate how a goal is initialized and used as part of the input for a component (in this example, a `SimplifiedQBPNetwork` used in the bottom level of the ACS):

```
... //Elided code initializing other world objects

GoalChunk g = World.NewGoalChunk();
Agent John = World.NewAgent("John");

SimplifiedQBPNetwork net =
    AgentInitializer.InitializeImplicitDecisionNetwork(John,
    SimplifiedQBPNetwork.Factory);

net.Input.Add(g);

... //Elided code performing additional initialization for the network
```

Note that all of the goals in the world are always specified as part of the "internal sensory information" and will automatically be "activated" in the `SensoryInformation` the next time one is perceived.

Now that we have shown you how to setup an agent to use goals, you need to know how to "activate" them. There are two methods for accomplishing this. The first is to set goals in the goal structure manually. The second is to set goals by using the "goal structure update action chunk." We begin by looking at how chunks are set manually.

Manually Setting a Goal

The simplest way to "activate" (or add) a goal in the goal structure is to manually "set" it. We do this by calling the `SetGoal` method for the agent where the goal is to be set. The code below demonstrates how we can do this for our agent, `John`:

```
John.SetGoal(g, 1);
```

We specify two items when calling this method: the goal that is to be set and its "activation level". This will "set" (or add) the goal in the goal structure. To "deactivate" (or remove) the goal from the goal structure we will call the `ResetGoal` method. In the CLARION theory, the term "reset" is equivalent to "remove" as it relates to the goal structure (as well as Working Memory). The following code demonstrates how we can manually reset (i.e., deactivate or remove) the goal in the goal structure:

```
John.ResetGoal(g);
```

These two simple methods provide you with all of the power you need to be able to use goals within the CLARION Library. However, manually setting the goals is only one of two ways to work with goals, and will often not be enough for more advanced simulations. The CLARION theory provides many more details regarding various additional methods for setting goals. For example, we can use "goal actions" in the ACS or in the MCS to perform operations on the goal structure. In the following section, we will look at how goals can be set using "goal actions" in the ACS.

Using Action Chunks

To begin, while the CLARION theory refers to actions that affect the goal structure as being "goal actions", the implementation uses a clearer term for describing these sorts of actions. In other words, in the CLARION Library, actions that perform updates on the goal structure are defined using the `GoalStructureUpdateActionChunk` class. The contents of these action chunks contain information about the sorts of updates that are to be performed. For example, suppose we want an action that "sets" the goal **g** in the goal structure. The following code sets up such an action:

```
GoalStructureUpdateActionChunk gAct = World.NewGoalStructureUpdateActionChunk();

gAct.Add(GoalStructure.RecognizedActions.SET, g);
```

Note that we specify, as the first parameter in our `Add` method, an enumerator called `RecognizedActions`. Several classes within the CLARION Library (namely those mechanism that can be manipulated using actions, e.g., the `NonActionCeneteredSubsystem`, the `GoalStructure`, `WorkingMemory`, etc.) define a `RecognizedActions` enumerator. This enumerator provides the list of commands that an action can perform on an instance of that class. The `GoalStructure` recognizes four types of actions:

- `SET`. "Adds" the goal to the goal structure
- `RESET`. "Removes" the goal from the goal structure
- `RESET_ALL`. "Removes" **ALL** of the goals from the goal structure
- `SET_RESET`. Combines the `RESET_ALL` and `SET` actions

If we want a component in the ACS to use this action, all we have to do is specify it in the output layer of the component. Below is an example of how we would set up this action in a network on the bottom level of the ACS.

```
... //Elided code performing additional initialization for the network

net.Output.Add(gAct);
```

Now, whenever the ACS selects this `GoalStructureUpdateActionChunk`, the system will perform the commands specified by that action. We will discuss another variation of this method (i.e., using a meta-cognitive module) in a later tutorial.

At this point you should have a basic foundation for building simulations in the CLARION Library using the goal structure. When you are ready to move on to the more complicated aspects of the MS (including integrating the MS with the MCS), the next tutorial, "*Intermediate MS & MCS Setup*", can be found in the "*Intermediate Tutorials*" section of the "*Tutorials*" folder.

Remember, as always, if you run into any problems, have additional questions, or want to report a bug, you can contact us at clarion.support@gmail.com.