Project presentation

The goal of this project was to implement a dialogue system for controlling a robot living in a virtual block world. The robot can move around objects of different forms, colors and sizes, and it can answer questions about the world and ask for clarification whenever it finds request ambiguous.

This has been performed using GrammaticalFramework (GF) for the grammar and Java/Prolog for the implementation. The world is described using a JSON file.

The world

Shrdlite uses a 2-dimensional blocks world.

The world is represented by a floor on which several objects (of different forms, colors and sizes) can lay. The objects can stand in/on each other (if it is permitted by the world’s laws).

The goal of this project is to move around the objects, according to the request of the user, through a robot arm which can pick up and put down objects.

The floor is divided in N spaces, meaning that there is room on the floor for no more than N objects at the same time. Each of this space is represented has a column, so that the world can be describe as a list of N columns of objects stake on each other.

The we implemented can contains objects of the following forms, colors and sizes:

Forms: Bricks, planks, balls, pyramids, boxes and tables.

Colors: Red, black, blue, green, yellow, white.

Sizes: Large, medium, small.

This world is described using a JSON file as follow:

[…]

Physical laws and the spatial relations

The world is ruled by physical laws that constrain the placement and movement of the objects:

* The floor can support any number of objects.
* All objects must be supported by something.
* The arm can only hold one object at the time.
* The arm can only pick up free objects.
* Objects are “in” boxes, but “on” other objects.
* Balls must be in boxes or on the floor, otherwise they roll away.
* Balls cannot support anything.
* Small objects cannot support medium or large objects.
* Boxes cannot contain pyramids or planks of the same size.
* Boxes can only be supported by tables or planks of the same size, but large boxes can also be supported by large bricks.

Note: We added the size medium so that we will have something to add to get more complicated worlds

Translation of a parse tree into a goal

It should work as well but we splitted us into two "teams" : one working on the translation and one on the solver+laws. And we defined the goal in a way that is not understandable by the solver

The solver

We need to add the facts of the type form(e,form,size).

We also need to add a rule that will allow the solver to take an object which is under a stack of objects

This will be done by something like this

take(o,k,[o'|[]],world,] :- put(o,k',[o'|[]],world,world'). take(o,k,[],world',). (here k is the number of the list of the list of lists in which o should be added)

We will keep working on the project during holidays in order to have at least a working code before the end of Easter.

Handling quantifiers

The "team" who worked on the translation of a tree into a goal already has ideas but did not implement them yet.

Ambiguous utterances

As we are working with prolog this should be fairly easily done.

Handling large worlds and complicated goals

We will read about probabilistic planning when we will have something working in order to handle larger worlds.

New linguistic structures

One of us has been working on it but as our solver does not work yet he did not add it to our code.

Description of the actions of the planner

As for ambiguous utterances it should be "easy" do define the description in the more concise way. In our solver we only have two ways of interacting with the world : either taking of putting an object from/on a list. So a sequence of actions of these types written in the more concise way possible will be our first goal when dealing with this task.

Speech recognition/synthesis

If we have some extra time by the end of the project we will try to add speech recognition and speech synthesis