CS405 Introduction to Empirical Modelling
Lab 5

1 Dependency Modelling Tool

The Depdency Modelling Tool (DMT) is a new Empirical Modelling tool. It represents an EM model by using directed acyclic graphs, and provides functionalities for the user to build the model by direct manipulation.
The DMT is written in Java and is therefore cross-platform compatible. You can find the JAR file in the EM archive (`/dcs/emp/empublic/projects/dmtWong2003`), and the latest version is 0.75 (`dmt075.jar`). There are other slightly older versions available if you encounter file compatibility problems with older DMT files. To execute the DMT in Computer science:

```
$ java -jar /dcs/emp/empublic/projects/dmtWong2002/dmt075.jar
```

A good guide to the Dependency Modelling Tool can be found at `~wmb/public/projects/tools/DMT/dmt.ps.gz` and is well worth printing out as a guide.

The DMT should be useful in the construction and analysis of your coursework as it can help you to organise and further understand your model. There are some example DMT models in the `dmtWong2003/models075` directory that you should use to experiment with the tool.

2 Experiments with the DMT

To see how the DMT can be used, consider the problem of analysing "bugs" in an existing tkeden model. For instance, execute the draughts model in `~empublic/projects/draughtsRawles1997`.
Start a game by clicking the appropriate button on the interface. Now try to change the background colour of the squares by setting

```
bgcol = "red";
```

You will observe that the unoccupied squares on the board retain a blue centre. This can be traced to the fact that the colour that is associated with a draughts piece on the square is set to bgcolor when the square is unoccupied. A suitable quick fix is to set

```
bgcolor = "red";
```

A better fix would be to introduce a separate colour variable and define both bgcol and bgcolor to be equal to it:

```
bgcolor is backgroundcol; bgcol is backgroundcol;
```

To see how this kind of analysis might in principle be supported by the DMT tool, execute the Run.e file in the ~wmb/public/cs405/draughts directory, and simultaneously display the dmt file draughtssq68sde.dmt.

By inspecting the dmt file, you can identify the eden, scout and donald components of all the observables relating to the square with coordinates [6,8]: this is the square that is displayed in the SCOUT screen.

Interrogation shows that the value of Square68 is determined by a definition in terms of the positions of all the white and black pieces:

```
Square68 is checkcol([6,8], bpieces, wpieces);
bpieces is [b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12];
etc
```

In order to get the model of the single square to operate properly, it is necessary to place the submodel in the right observational context: assigning locations to the pieces b1, b2, b3, b4, ... You can find suitable assignments for the relevant variables commented out at the end of the file draughtsdefns.e:

```
b1 = [2,6]; b10 = [4,8]; b11 = [6,8]; b12 = [8,8]; b2 = [4,6]; b3 = [6,6];
b4 = [8,6]; b5 = [1,7]; b6 = [3,7]; b7 = [5,7]; b8 = [7,7]; b9 = [2,8];
bcrowned = [0,0,0,0,0,0,0,0,0,0,0,0];a

etc
```

Once you have done this, you can experiment with setting values for the variables bgcol and bgcolor and the observables b11, w11 etc that determine the positions of pieces to demonstrate that there are problems in the model. It is useful to interpret these problems with reference to the DMT model.

2.1 Supplementary exercise

Consider the dependencies that are required in modelling the states, relationships between states and transitions between states in a statechart. Investigate how these are implemented in the digital watch model:

```
/dcs/acad/wmb/public/projects/misc/digwatch/watchcarlos
```

which is essentially the model discussed in the Lecture T4. Contrast this with the statechart models that are generated by SIT, a statechart editor that was implemented by Rob Carpenter in his third year project:

```
/dcs/acad/wmb/public/projects/tools/editors/sit
```

To inspect a sample statechart generated by sit, enter:

```
tkeden radio.all
```

and include the file radio_picture.e to obtain a simple visualisation of a radio to accompany the statechart. Finally, explore how you might use the DMT to display the structure of the dependency structure associated with statechart states.

3 Digital Watch Model

The following exercise is based around the digital watch model developed by Chris Roe. This was derived as a variant of the digital watch model discussed in lecture T4.

The model is obtained by executing Run.e in digitalwatchRoe2001. As it stands, the model is incomplete in as much as the extra clock mechanis is not implemented.

To make the watch tick, set the variable 'input' to 8.

There are some basic exercises in the use of the watch that you might like to attempt to familiarise yourself with the watch operation: these are displayed in the text box at the bottom of the screen.

To get a new question, change the value of qeustionNo (initially set to 1).

When you have done all four exercises you will have exercised much of the functionality of the watch.

There are three exercises in modifying the functionality of the model that you can consider: they require some effort, but you can crib by looking at the answers and trying to understand them if you like.

1. adapt the updating mechanism of the watch so that pressing the brown button loops cyclically through the `update_sec`, `update_min`, etc functions, and adapt the statechart accordingly.
2. add the missing functionality of the extra clock.
3. add a small display to represent two runners running a race (use donald lines that extend in length towards a finishing line when the stopwatch is started), and then demonstrate how to use the split time function on the watch to time the 'runners'.

For more background on this model you can see papers in the `papers` directory of empublic, notably 069.pdf. This discusses the many different interpretations of state that interaction with the model supports.

3.1 Further study

For more ideas about EM and interfaces, with particular reference to the potential role of LSD in specification, you might like to look at the following models and associated references:

- Leung / [YP]Yung's 2-view editor for DoNaLD. See

  `~wmb/public/projects/tools/editors/2vde`

  for a tkeden prototype 2-view editor for DoNaLD that allows diagrams and scripts to be constructed by direct manipulation, and see

  `~empublic/papers/039.pdf`

  a paper describing the design of a somewhat enhanced version of such an editor that includes an LSD specification of the interface.

- Adzhiev / [YP]Yung's conceptual design of an interface for a CSG modeller. See

  `~wmb/public/projects/misc/CSG`

  for an extensive LSD specification (csg.lsd) which describes the conceptual interface to a fictitious CSG modeller, together with an 'automatically generated' tkeden implementation (csg.e.tk). And

also see

`~empublic/papers/038.pdf`

for a discussion of the role of the LSD specification in the context of geometric modelling using function representations.