02285 AI and MAS
Mandatory Assignment 2
Due: Tuesday, 1st of April at 13.00

Mikkel Birkegaard Andersen

Formalities

Part of this assignment is assessed individually, so, per the DTU Study Handbook, the following rules apply:

- Discussing the exercises with your fellow students *is* allowed.
- Your answers must be your own, *from the moment you start writing down anything.*
- You must hand in your solutions individually.
- Indicate on the first page of your hand-in who you had discussions with.
- *All* parties involved in a breach of these rules will be held responsible.

For the common part, you may collaborate as much as you like in the group, whether on discussion, implementation and or report-writing. For this part, we require only one of you to hand in the report, though *do* remember to indicate all group members. A group may not collaborate with other groups on implementation and report-writing, though it may discuss the exercise with people outside the group. If you do this, make sure to indicate this in the report.

Your solutions are to be handed in via Campusnet no later than Tuesday, 1st of April at 13.00, in `.pdf`-format. Make sure your answers are clearly legible and use the proper notation ($\neq, \neg, \Delta, \dots$).

Individual Part

Exercise 1 (Hierarchical Planning)

The following is a classical planning specification of a domain in which there are a number of trucks and airplanes and packages needing to be delivered to specific locations. Trucks can move within a particular city and planes can fly between airports in different cities.

```
Types:
   physobject : object
   package : physobject
   vehicle : physobject
   truck : vehicle
   plane : vehicle
   location : object
   city : object
   airport : location
 Predicates:
   At(o : physobject, l : location)
   InCity(l : location, c : city)
   InVehicle(p : package, v : vehicle)
 Actions:
   Load(p : package, v : vehicle, location : l)
     Pre: At(p, l), At(v, l)
     Eff: InVehicle(p, v), ¬At(p, l)
   Unload(p : package, v : vehicle, location : l)
     Pre: InVehicle(p, v), At(v, l)
     Eff: At(p, l), ¬InVehicle(p, v)
   Drive(t : truck, x : location, y : location, city : c)
     Pre: At(t, x), InCity(x, c), InCity(y, c)
     Eff: ¬At(t, x), At(t, y)
   Fly(a : plane, x : airport, y : airport)
     Pre: At(a, x)
     Eff: ¬At(a, x), At(a, y)
```

The notation `type1 : type2` indicates type inheritance; `type1` is a specialisation of `type2`. Everything is an `object`, both `trucks` and `planes` are `vehicles`, etc. In the classical planning formulation, goals are of the form `At(`$\mathtt{p}_1$`,`$\mathtt{l}_5$`)`, where $\mathtt{p}_1$ is a package and $\mathtt{l}_5$ is a location.

a) Write a set of HTN methods and operators for this domain. Remember to specify top-level tasks corresponding to classical goals. You may make modifications to the domain, e.g. introduce new predicates like `Destination(p : package, l : location)`, but you don't have to. Remember, that tasknames do not have to mention all variables, but method and operator signatures do. This was unclear in the original lecture slides, but has been explicitly stated in the new version.

b) Draw the fully refined decomposition tree for the problem corresponding to the following classical planning problem:

```
Constants:
  P1, P2 : package
  T1, T2 : truck
  A1 : plane
  DTU, KU, KTH, SU: location
```

```
  CPH, STO : airport
  Copenhagen, Stockholm : city
Init:
  At(P1, DTU), At(P2, DTU), At(T1, CPH), At(T2, STO), At(A1, STO),
  InCity(DTU, Copenhagen), InCity(KU, Copenhagen), InCity(CPH, Copenhagen),
  InCity(KTH, Stockholm), InCity(SU, Stockholm), InCity(STO, Stockholm)
Goal:
  At(P1, SU), At(P2, KU)
```

In the tree, indicate what methods have been used to decompose tasks, unless it is completely unambiguous. It is enough to write constants (that is, you can forget about binding constraints). Note that there may be several possible decompositions, depending on how you design your hierarchy. Just show one of them.

c) Can your hierarchy produce optimal plans? Does it guarantee them?

Common Part

Exercise 2 (Playing a one-sided game)

For the next phase of the project, your task is to implement a team consisting of 8 `Explorer`-agents. They should be able to explore the environment and coordinate their positioning such that the team achieves a good score.[1] There should also be 8 `Explorer`-agents on the opposing team, though they will not be doing anything. Node weights and edge weights should be in the default range, as specified in `config.dtd`.[2] There are not going to be any achievements in this phase, nor the final third phase starting in three weeks, so your strategies cannot rely on buying upgrades. As with Mandatory 1, you are to do the following:

a) Implement a program that can do the above. You may use any language you like.

b) Write no more than 5 pages detailing your solution. You should use the same overall structure as you did for the previous assignment. If you wish, you can add a *brief* introduction summarising what is in the report. The guidelines for report writing from Mandatory 1 still apply.

c) Have one of the group-members hand in two files, a `zip`-file with the source to your program, and the mini-report in `pdf`-format (and not in a `zip`) via CampusNet. *Both the individual and common parts of the assignment have the same deadline.*

d) Be ready to demonstrate your solution on your own computer at the exercise class on the 1st of April.

[1] What is considered a good score is deliberately left vague. You may approach the problem in any way that you like. And, of course, there may be plenty of room for improvements later.

[2] So if you do not specify them in the `simulation`-tag, they will have the correct default values.