**Artificial Intelligence I**, *prof. Pasquale Caianiello 22.01.17*

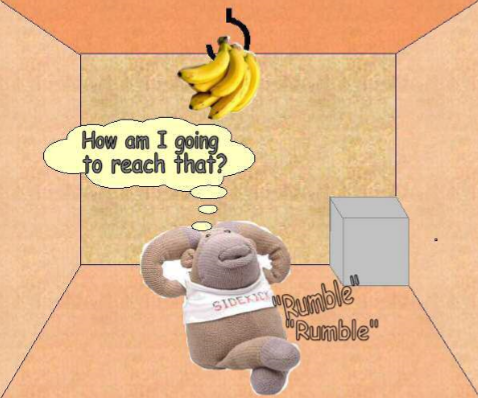
Homework 3: Planning. *Presented by: Yuna Frolov*

**Monkey and Banana Problem**

The **monkey and banana problem** is a famous toy problem in artificial intelligence, particularly in logic programming and planning.

**The problem:**

A monkey is in a room. Suspended from the ceiling is a bunch of bananas, beyond the monkey's reach. However, in the room there is a box. The ceiling is just the right height so that a monkey standing on a box could take the bananas. The monkey knows how to move around, carry other things around and reach for the bananas. What is the best sequence of actions for the monkey?

 ->>>>> 

**\_\_init\_\_.py**

**MonkeyBanana.pl:**

The file starts with definition of the facts about the monkey, banana and box and locations.

Then the possible actions the monkey can do in order to achieve the goal. A monkey can:

**Go (Monkey, To)** – monkey can go to a location

**push(Monkey, Box, To)** - monkey can push the box to a location

**climb\_on(Monkey, Box)** - monkey can climb on a box

**climb\_off(Monkey, Box)** - monkey can climb off a box

**grab(Monkey, Banana)** – monkey can grab a banana

Then we have definitions of locations the objects can be in:

**location(Object, Loc, [])** - defines the initial locations

**location(Monkey, Loc, Actions)** – monkey location according to actions

**location(Box, Loc, Actions)** – box location according to actions

**location(Banana, Loc, Actions**) – banana location according to actions

Then we have the functions to verify the goal:

**has\_banana** – checks if the monkey has a banana

**on\_top\_of\_box** – checks if monkey is on top of a box

Finally, we have the definition of the goal state:

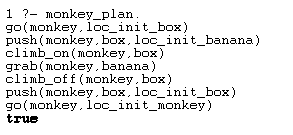
**goal\_state(Actions)** – monkey need to have the banana, and monkey and box have to be in their initial place.

The plan requires a list, defined by **writeList([]).**

Then, **monkey\_plan** achieves the shortest plan for the monkey to get the bananas using iterative deepening (depth first search), each time taking an action and validating if it is possible. Finally, reversing the actions taken and inserting them in the list to be printed.

**\_\_init\_\_.py**

**A working run:**



Monkey goes to a location of the box, pushes the box to the banana location, climbs on the box, grabs the banana, climbs off and then pushes box to initial location and goes back to it’s initial location, while holding bananas, thus reaching the goal.

**\_\_init\_\_.py**

In my research I found use in Kevin Tindall’s implementation from GitHub, and made changes according to the requirements of the homework.