# Assignment for Internship or thesis project on discovering decision strategies using meta-level reinforcement learning

Clever cognitive strategies are essential to good decision-making. An important part of our research in the Rationality Enhancement Group seeks to understand how people can learn clever strategies for thinking and decision-making and to leverage machine learning to assist them in this process. One approach we would like to explore is developing (cognitively-informed) Bayesian program induction methods for discovering a curriculum of rational decision strategies that people can be taught by an intelligent tutor. Our approach to strategy discovery combines meta-level reinforcement learning with Bayesian program induction. This assignment therefore has two parts: The first parts tests your ability to develop a simple meta-level RL algorithm and the second part assesses your ability to learn about Bayesian program induction.

## Part I: Meta-level Reinforcement Learning

Let’s assume we have a simple environment where the agent repeatedly chooses between two gambles of the form (x1 with probability p and y1 with probability 1-p) vs. (x2 with probability p and y2 with probability 1-p) where p~Uniform([0.5,1]) and x1,x2,y1,y2 ~ Uniform([-10,10]). On each trial the probability and the payoffs of both gambles are different and known to the agent. The agent has 1000 time steps to earn as much money as possible and is equipped with two different decision strategies: the Lexicographic heuristic (choose based on the payoff for the most probable outcome) and the Equal-Weight heuristic (which averages the payoffs and chooses the gamble with the higher average payoff). Each heuristic performs a sequence of operations and each operation takes one time step. Executing the Lexicographic heuristic takes less time than executing the Equal-Weight heuristic because it performs fewer operations. Develop a meta-level RL algorithm that learns to adaptively choose between the two heuristics according to the value of p so that the agent obtains as much reward per timestep as possible. Plot learning curves showing how the learners performance changes over time and how their preference for the two heuristics evolves over time for p>=0.9 versus p<=0.6.

If you are not sure how to approach this problem, feel free to take a look at our previous work on metacognitive reinforcement learning.

## Part II: Bayesian program induction

Write a WebPPL program ([http://webppl.org](http://webppl.org/)) that computes the posterior probability that any given set of strings (e.g., aab, bbaa, aaaaab) was generated by Program 1 versus Program 2. Here each candidate program is represented by a probabilistic context-free grammar.  Write your code in such a way that it can be easily extended to handle a larger space of potential programs. Test whether your code is working and be prepared demonstrate and explain the result.

1. P(Program 1)=0.7
2. P(Program 2)=0.3
3. Program 1:
4. P(S-->aSb)=0.5
5. P(S-->bSa)=0.2
6. P(S-->a)=0.1
7. P(S-->b)=0.2
8. Program 2:
9. theta ~ Beta(0.1,0.1)
10. P(S-->Xa)=theta
11. P(S-->bY)=1-theta
12. P(Y-->aY)=0.4
13. P(Y-->b)=0.6
14. P(X-->Xb)=0.2
15. P(X-->a)=0.8