Kiel University
Christian-Albrechts-Universität zu Kiel

# Assignment 1
# Autonomous Learning
May 4th, 2020

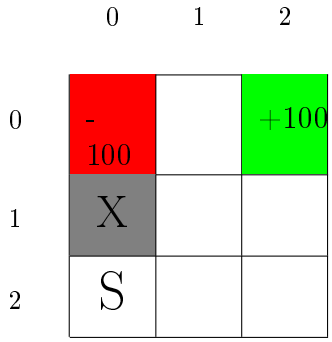
## Assignment 1 - StarCraft II Framework in Python

A. Install, (if not already done) **Python 3.6** → www.python.org
**Recommendation:** Use Anaconda (www.anaconda.com) for managing your Python packages and installations. To do this, install Anaconda/Miniconda and create a new virtual environment named *pysc2* where you can install all other packages. Follow the instructions in *PySC2_Env_Setup.pdf*.
All information on Anaconda can be found in the documentation.
An introduction to virtual environments and Miniconda can be found here.

B. Install **pysc2** → www.github.com/deepmind/pysc2
Follow the description given in *README.md*

## Assignment 2 - Markov Decision Process

Given is the following Markov Decision Process:




- The agent starts in State (2,0) marked with a big S
- Cells with an X cannot be used
- The agent has 4 possibilities for actions: (N, O, S, W)
- The agent carries out the action he has chosen with a probability of 70% and goes in the corresponding direction. With a probability of 10% in each case he goes in one of the other directions
- If the agent can't get any further in one direction, he stops

- The agent gets a reward of -1 in each step
- If instead the agent leaves one of the coloured fields and reaches the final state, then he will receive the Reward that is on the field

A. Compute the V-Values $V^k(s)$ for all states $s \in S$, $k = 5$, $\gamma = 0.95$

B. Determine from the V-Values of the previous task the optimal Policy $\pi(s)$ for all states $s \in S$

## Assignment 3 - Basic Agent

A. Create a Python project (for example with PyCharm) in which all following tasks (sheets) are included. The project should be named after the group name (choose one for your team!). When creating the project, follow the instructions in *SLS_ Boilerplate.pdf*.

B. Implement an agent that solves the minigame "MoveToBeacon"!

   (a) In each step the agent has 8 possibilities to choose a direction. (N, NO, O, SO, S, SW, W, NW)
   (b) The size of the playground is $64 \times 64$
   (c) The agent chooses that action in each step which minimises the distance to the goal.

C. The agent should be able to be started by a Python file (RunBasicAgent.py). The file should be on the top level of the project. Copy the script *runScript.py* from the boilerplate and adapt it accordingly.

## Assignment 4 - Q-Learning

A. Extend the agent from the previous tasks by a Q-Learning component.

- An episode is 1920 frames (pre-configuration of the minigame-map)
- It is allowed to select the marine in the first step of a run.
- An episode continues until the agent reaches the target or the run is over.
- For the modelling of the state space only the x and y distance from the marine to the beacon should be used. Furthermore, the state space shall be divided into areas:

$$x_1 < s_1 < x_2$$

$$y_1 < s_1 < y_2$$

  This serves to limit the state space.
- The agent receives a reward of 1 when he reaches the target. (use the already implemented reward function of pysc2)
- The exploration strategy will be $\epsilon$-greedy with a linear decreasing $\epsilon$.
- The Q-table should be created with pandas (https://pandas.pydata.org/) and saved in Pickle format.

B. Create two graphs with the help of the tensor board which show the learning progress of the agent.

- Extend the *summarize* method in the runner class so that the graph represents a moving average of rewards with a sliding window = 50.
- The second graph should show the utilised Epsilon.

C. The graphs showing your training progress are part of the assignment.

D. The trained Q-Learning agent should be able to be started by a Python file (RunQLAgent.py). The agent should read the trained Q-table (trained by you) and use it without further learning.

E. The training of the Q-Learning agent should be able to be started by a Python file (TrainQLAgent.py). The agent should create a new Q-table.