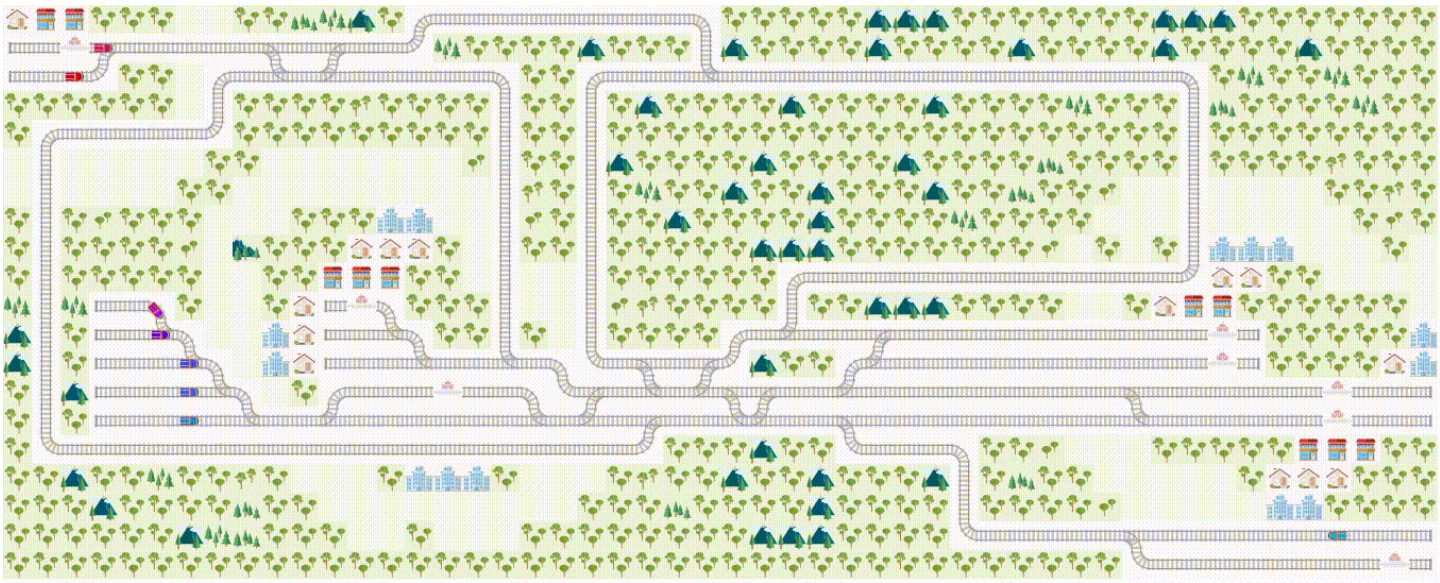
Introductory assignment

Background information

What is the Flatland Challenge?

> The Flatland challenge aims to address the problem of train scheduling and rescheduling by providing a simple grid world environment and allowing for diverse experimental approaches.
> *—Competition background*

The problem is a simplified version of what railway operators face in real life: the daunting task of scheduling trains to ensure that every one of them reaches its destination on time and without incurring any collision. **The goal** of the challenge is to gather teams from around the globe who test various machine learning approaches to find which methods work best.



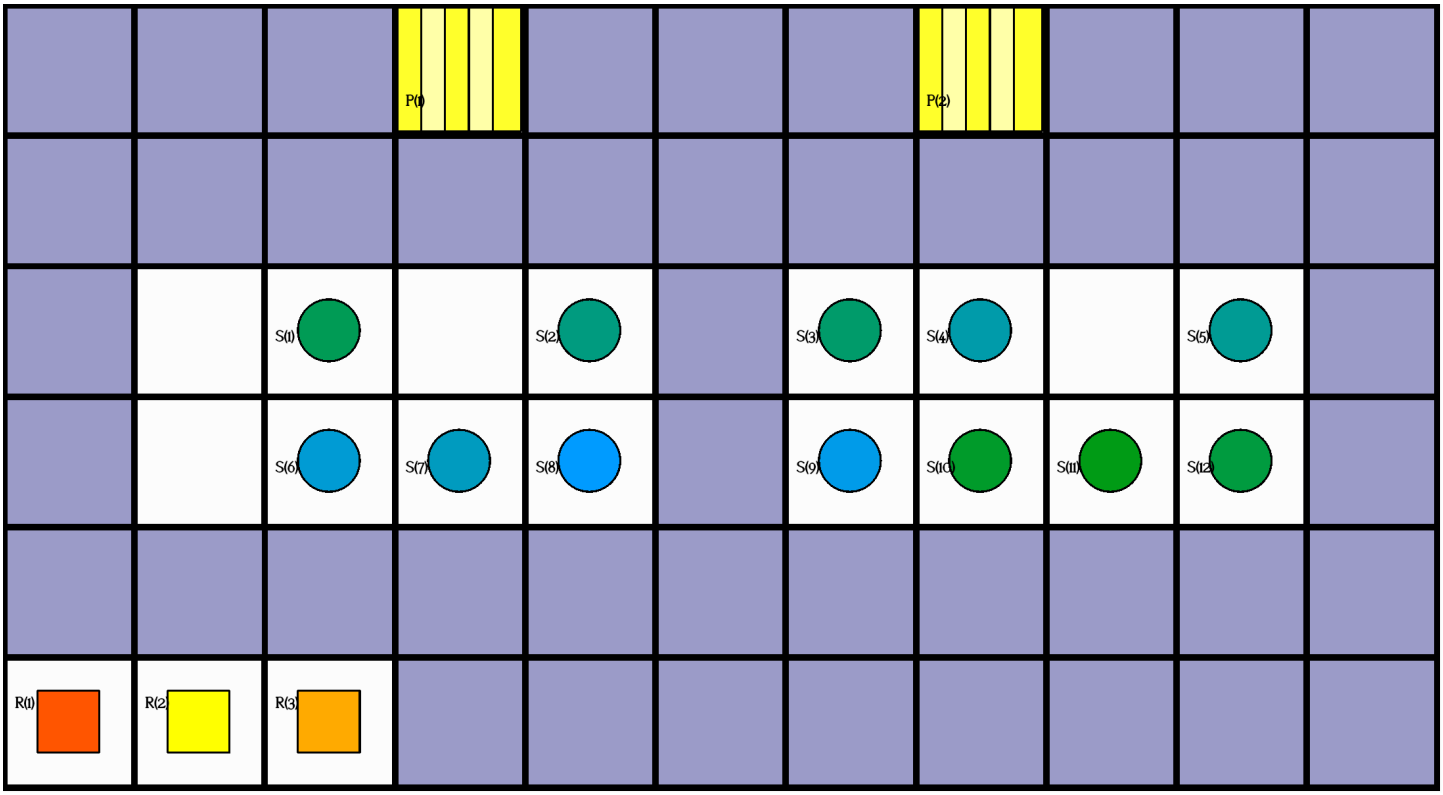
The initial expectation of the creators of the competition was that this would be solved using primarily a reinforcement learning approach. However, answer set programming (ASP) shows promise, as it has been successfully implemented in similar problems, such as multi-agent path finding problems.

Developing and maintaining a professional rapport with those in charge of the competition would be mutually beneficial.

What is asprilo?

> *asprilo* is an benchmarking framework to study typical scenarios in intra-logistics and warehouse automation with multiple mobile robots.
> *—Potassco about asprilo*

The asprilo environment allows users to simulate the movement of agents within a simplified grid environment. Realistic applications of such a representation include warehouses or production floors. Agents within the environment represent robots who have tasks to complete and must do so without interfering with other robots.



The above implementation can be encoded in ASP. First, each node within the environment is established. After that, anything occupying the grid—such as robots, shelves, and other objects—are then defined after that. Each initialization is represented by an `init()` predicate, within which can be found several crucial pieces of information, including the object type and its location. A snippet of ASP code defining the above implementation can be seen below:

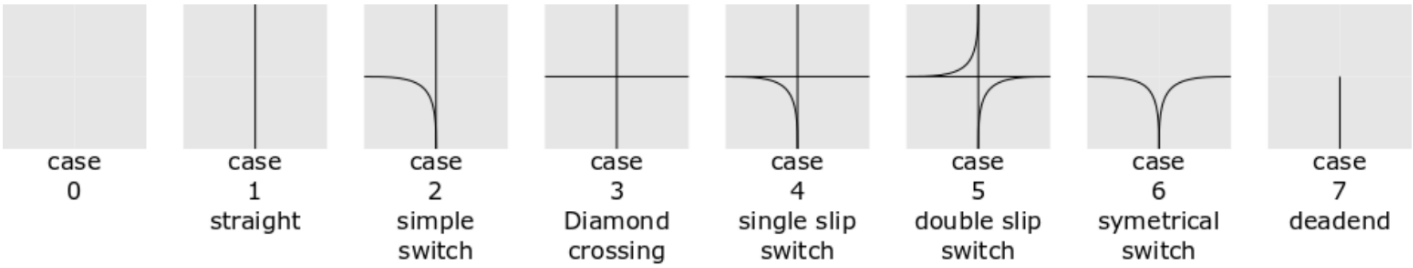
```
% initialize nodes
init(object(node,1),value(at,(1,1))).
init(object(node,2),value(at,(1,2))).
init(object(node,3),value(at,(1,3))).
… % etc.

% initialize robots
init(object(robot,1),value(at,(3,4))).
init(object(robot,2),value(at,(5,4))).
init(object(robot,3),value(at,(7,4))).
… % etc.

% initialize shelves as destinations
init(object(shelf,1),value(at,(1,1))).
init(object(shelf,2),value(at,(2,1))).
init(object(shelf,3),value(at,(3,1))).
```

Various approaches can then lead to solutions wherein agents traverse the grid to complete their tasks. Agents can do so by moving from one cell to the next during any given time step. Certain constraints must be considered, such as moving into unoccupied spaces. In principle, the tasks that asprilo represents bear similarity to those in the Flatland challenge—each agent must navigate a grid-like world, beginning at its origin and ending at its destination, without interfering with others doing the same. However, there are differences between the two which indicate that asprilo alone would not suffice as an ASP implementation for Flatland.

What is a track transition?

The major difference between the asprilo environment and the Flatland environment is the freedom of movement. **In asprilo**, there is a path connecting each cell to its four neighboring cells (to the north, east, south, and west). An agent may move from its current cell to any of the four neighboring cells, provided no constraint impedes its ability to move, such as an obstacle or another agent already present in the neighboring cell. **In Flatland**, the ability for an agent to proceed to an adjacent cell is determined by the type of track in that cell. The track acts as a guide that dictates all possible paths an agent may take. For instance, if an agent is traveling along a straight track, it cannot move to the left or to the right—it must go straight.

The possible paths an agent may take based on the track type present in that cell is known as a track transition.



In Case № 1, for example, the train must continue straight. It does not have the option to move to the left or to the right.

In Case № 2, for example, a train approaching this cell from beneath would have the **decision** to continue forward to the north, or to follow the curved path to the west. A train approaching from the west, however, may only follow the curved path; it cannot travel to the north. Consequently, there are four possible paths for this type of track. The track can be rotated in any direction, as well as flipped horizontally, leading to eight possible configurations. Considering the number of possibilities, encoding a reliable transition function to describe possible paths for each track type, regardless of its orientation, may play a critical role in a working environment.

Assignment

This assignment asks two broad questions:

1. Generally speaking, how could the Flatland environment be represented in ASP?
2. How could the problem of track transitions (i.e. allowing only legal paths for agents) be solved in ASP?

To answer both of these questions, it would be beneficial to investigate the Flatland challenge in more depth, paying close attention to the attributes within the environment—the trains, the tracks, the stations, etc.

- How each of these could and should be represented in ASP?

Consider elements from the asprilo framework. Which aspects can be transferred; which aspects should be adapted and how could they be changed to fit this problem?

Then, attempt to represent the environment below in ASP. The red triangle represents the

train facing east, and the blue square represents its destination station.