Flatland Encoding

> Week 2

Progress

The goal for this week was to reduce a Flatland environment into graph that considers only decision nodes (i.e. switch tracks).

Comments

👍 Positive feedback

The overall graph representation looks good. We are glad to see that you have been able to create a custom environment in Flatland that doesn't automatically close out by itself.

⚠️ Making improvements

Relative actions

To be more like Flatland, we need to embrace relative actions:

> - Move forward
> - Turn left
> - Turn right

Using these as opposed to absolute cardinal directions will bring this more in line with the way the competition environment is structured. This also means we need to retain the current orientation (direction) of the train at each time step.

Reducing redundancies

There are likely redundant qualities about the way the transitions are written. Perhaps we don't need all of the additional zeros at the end:
`value(transitionEast,((15,18,north),0,0,0))).`

If you know the direction of the transition and which cell you're going to (and the distance), that should be all you need.

Automatically-generated transitions

The process of manually defining all transitions becomes too cumbersome at larger scales. If there is a way to automate that process, that will save time as environments become more complicated.

Next steps

1. Consider improvements to the representation
2. Develop an encoding for actions and use them to solve simple instances, such as the environment from the first assignment
3. In the encoding, you will want to:
    i. Generate a choice rule
    ii. Ensure that the preconditions for the transitions apply
    iii. Assign the consequence at the next time point if the action should occur

Take a look at past asprilo projects to get an idea of how time steps are incorporated into projects. Here is some pseudocode that may set you in the right direction:

- You have a predicate that defines that train's position
    - `at(ID,Position,Time).`
    - `occur(Action,Time)`
    - These are the two things that depend on time
- Then a choice rule is applied to this
    - `1 {  occur(Action,Time) } :- precondition(Action,Time).`

For now, work on transitions for a minimal example, ignoring the long stretches of track. Then, when you include the long stretches, you can make the assumption that the train is taking only one step (just to generate the path). After that is successful, see how you can create the path and track the position of the track.