Computer and Information Sciences
University of Strathclyde

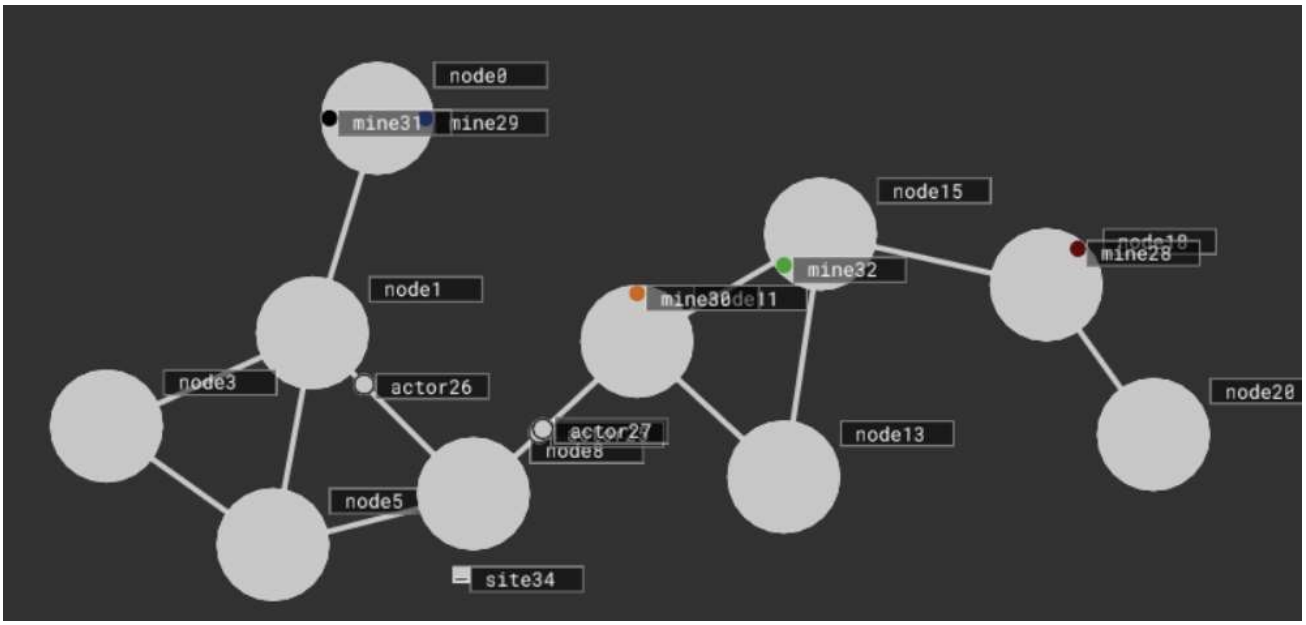
Assignment 2 – Temporal Networks – Worth 25% of Overall Mark

Submission deadline: 02/12/22

Online submission through MyPlace

Background

CraftBots is a light-weight simulation environment with properties that can be configured to increase difficulty or introduce particular problem features. The simulation is written in Python and should run on low-end machines. In assignment 1, you used PDDL to create domains, a problem, and generate plans. In this assignment, you will create and modify simple temporal networks (STNs).



For a reminder, the simulation is shown above, which models the same scenario setup that we will use in this coursework. The scenario includes 3 actors (solid grey circles) moving between 10 locations (large white circles) connected by edges (blue lines). The aim of the scenario is for the actors to use the resources to construct as many buildings as possible. After part 1, you should have created a domain, a problem, and be able to generate a valid plan. **However, if you do not have a plan, a sample plan is available on MyPlace.**

Your task will be to take the initial plan and first parse it into Python so that it can be read (a skeleton function will be provided). You will also create a function that will write your STN into a DOT graph. The next task is to check STN consistency using the Floyd-Warshall algorithm and produce the all-pairs shortest path . You should also remove all dominated edges. The final output of all your tasks will be a DOT file that can be visualised to make a STN graph.

A skeleton Python file will be provided (assignment.py), and you should use this as the basis of your assignment. You must not change the values returned or the headers.

Created October 2022
By Dr Andrew Abel, modified from a version from Dr Michael Cashmore

Computer and Information Sciences
University of Strathclyde

1: Parsing an STN from a PDDL Plan (10 marks)

The first part of the coursework is to parse an STN from a PDDL file.

- The plan should be generated using your domain from the first assignment, or using the example domain and problem craftbots_domain.pddl and craftbots_problem.pddl (will be provided on MyPlace).
- You should read a plan file and generate an STN, listing nodes and edges
- A function header will be provided in assignment2.py (on MyPlace), and you should complete the `make_stn(plan)` function, returning the nodes and edges.

Each action in the plan should be associated with 2 nodes. These nodes correspond to the start and end of each action. Edges, representing the orderings between nodes, should be formed from:

- Action duration - two nodes represent the start and end of the same action, and so must be ordered.
- Causal support - one node is associated with an effect that achieves the condition of the other node, and must be ordered before.
- Interference - one node violates the condition of another node, and so must be ordered before or afterwards (depending upon how the nodes are ordered in the plan).

2: Print an STN graph in DOT format (5 marks)

For this task you must complete a function to print the STN in DOT format.

- Nodes must be labelled by the action name, parameters, and whether they are the start or end of the action.
- Edges must be labelled with the time bounds of the action in the form [lower, upper].
- You can find detail on DOT here: https://graphviz.org/doc/info/lang.html.
- You must complete the `print_stn(nodes, adj_list)` function, and write the data to a temp_network.dot output.
- It should be possible to visualise your STN through VSCode (more detail in the labs).

3: Check STN consistency (5 marks)

In this part of the assignment the Floyd-Warshall algorithm should be used to check the consistency of the STN plan.

- If the plan is inconsistent, then the method should detect this.
- If the plan is consistent, the STN should be converted into an all-pairs shortest path graph.
- You should complete the `floyd_warshall()` function.
- You will need to test this yourself by modifying the plan file.

4: Removing all dominated edges (5 marks)

In this part of the assignment redundant edge checking should be implemented to identify and remove all dominated edges.

- Redundant edges should be removed from the network so that it can be efficiently executed.
- An edge is redundant if it is dominated by another edge (as described in the lectures)
- You should complete the `make_minimal()` function

Created October 2022
By Dr Andrew Abel, modified from a version from Dr Michael Cashmore

Computer and Information Sciences
University of Strathclyde

Submission Instructions

You must submit 3 files:

1. Your completed Assignment.py file containing all the functions that you have completed. You will lose marks if the method headers are changed, or you do not use the assignment.py layout.

2. One STN graph, temp_network.dot, which is the optimised and consistent STN

3. One plan file, which your .dot file is based on

Created October 2022
By Dr Andrew Abel, modified from a version from Dr Michael Cashmore