INF2D – Reasoning and Agents

Coursework 2: Symbolic Planning

TA: Rimvydas Rubavicius (rimvydas.rubavicius@ed.ac.uk)

TA: Jay Park (jay.jh.park@ed.ac.uk)

Learning Objectives

- Gain hands-on experience of designing ‘effective & efficient’ formal representations of planning problems
  - Learn how to code in PDDL, a declarative language for formally expressing planning problems

- Understand how performance of a planning algorithm may be affected by different factors, including:
  - Parameters to the planning algorithm
  - Your design choices!

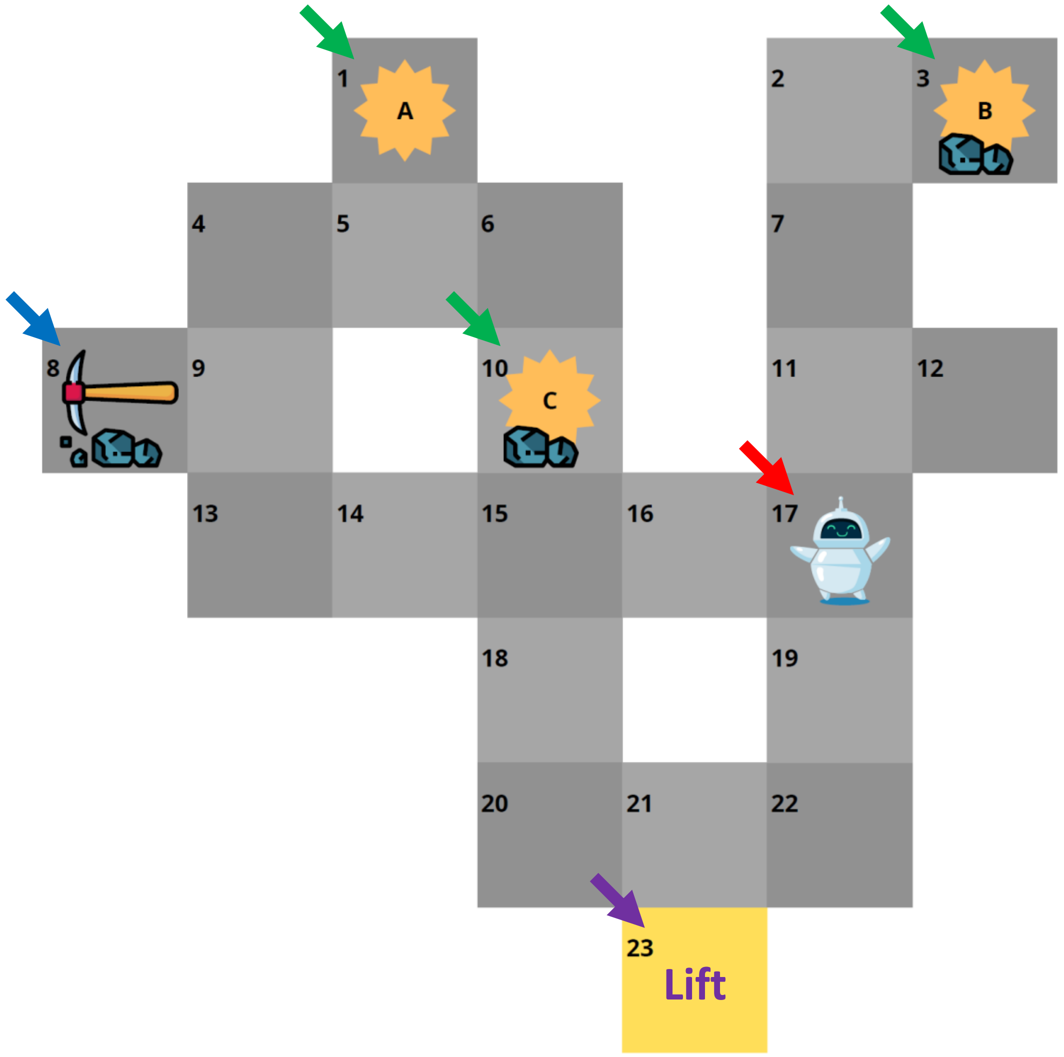
Coursework outline

- High-level goal: Design symbolic planning domains that model real-world practical scenarios involving a miner robot
  - Open-ended task requiring your own decisions about abstraction
  - No single ‘correct’ design… but there are such things as *inefficient* designs!

- CW2 is broken down into three tasks:
  - Modelling (35%)
  - Experiment (15%)
  - Extensions (50%)

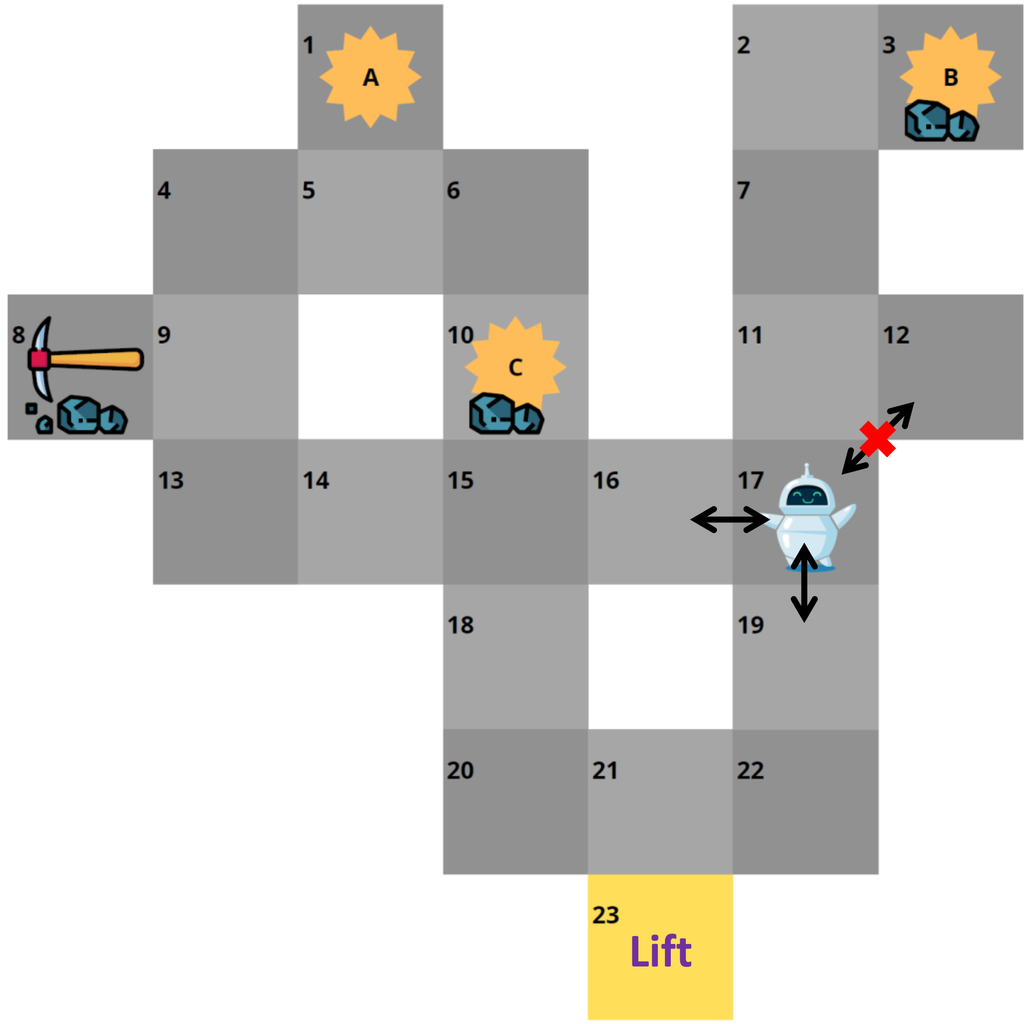
Task 1: Modelling

- Scenario
  - A MineBot is tasked to mine *all* of the ores present in a mine
  - An ore can be either directly picked up (A), or may be 'blocked' by rocks (B, C)
    - If an ore is blocked, MineBot first needs to 'unblock' it with a hammer
  - An ore is considered 'mined' if it is placed in the lift that is turned on
    - The lift is turned off at the beginning
    - To turn on the lift, MineBot must be present in the cell where the lift is placed



Task 1: Modelling

- Scenario (cont'd)
  - The layout of a mine consists of cells (numbered squares in the figure), in which an object may or may not be present
  - A MINEBOT can move between two adjacent cells
    - No diagonal movements!
  - A MINEBOT can pick up at most one object – iff it is not already holding anything



Task 1: Modelling

- Programming in PDDL (Planning Domain Definition Language)
    - A declarative programming language
        - Specifies *what* problems to solve, NOT *how* to solve problems
        - Solving of the planning problems is entirely delegated to the Metric FF planner
    - A PDDL representation of planning problems consists of:
        - One PDDL domain file, defining the “universal” aspects of problems
        - One or more PDDL problem files, each instantiating a particular planning problem

Task 1: Modelling

- Anatomy of a PDDL domain file
  - (Example in handout)

```
(define (domain blocks-world)
    (:requirements :adl)

    (:types table block)

    (:predicates
        (On ?x - block ?y - object)
        (Clear ?b - object)
    )

    (:constants Table - table)
```

(cont'd in next slide…)

Some domain name of your choice

Similar to importing packages in other languages; would only need minimal modifications when necessary

Declaration of object 'types' in this domain

Declaration of predicates, by their names and arguments

Declaration of 'constant' entities that will be present in all problem instances in the domain

Task 1: Modelling

- Anatomy of a PDDL domain file
    - (Example in handout)

Declaration of an action schema, by its name, parameters, preconditions and effects

(cont'ing from previous slide…)

```
    (:action MOVE-TO-TABLE
        :parameters (?b - block ?x - block)
        :precondition (and (On ?b ?x) (Clear ?b) (not (= ?b ?x)))
        :effect (and (On ?b Table) (Clear ?x) (not (On ?b ?x)))
     )
)
```

Implementation of:
    Action(MoveToTable(b, x),
        Precond: $On(b, x) \wedge Clear(b) \wedge b \neq x$)
        Effect: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x)$)

Task 1: Modelling

- Anatomy of a PDDL problem file
  - (Example in handout)

```
(define (problem block-problem)
    (:domain blocks-world)
    (:objects
        A B C - block
    )

    (:init
        (On A B) (On B Table) (On C Table)
        (Clear A) (Clear C)
    )
    (:goal (and
        (On A B)
        (On B C)
    ))
)
```

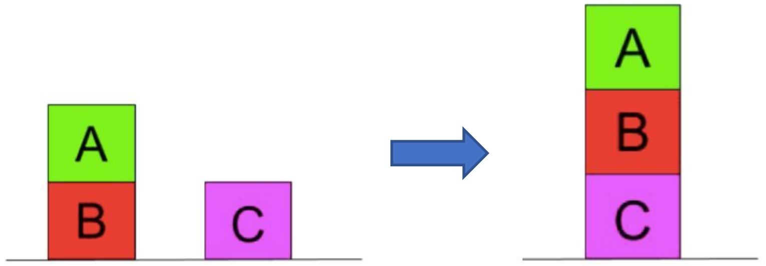
Some problem name of your choice

Specification of the planning domain for the problem (defined earlier)

Objects existing in the scope of the problem and their types

Initial state specification

Goal specification



Task 1: Modelling

- Testing your domains & problems
  - You can test your PDDL domain & problem files locally before submitting by running the Metric FF planner included in the handout (binary executable with the name `ff`)

  - To run the planner, execute the following command on your shell, in the directory you unzipped the handout:

```
./ff -o {domain_file_name} -f {problem_file_name}
```

Task 2: Experiment

- Now that we have our first suite of planning domain & problems, let’s conduct some experiments…

- Task 2.1: Design a harder problem
  - The planning problem instance encoded in Task 1.2 is not challenging enough for the Metric FF planner
  - Can you come up with a harder problem that would actually distress the Metric FF to a greater extent?
    - Make it challenging enough that you can later observe noticeable differences in planner performance in the next task
    - Justify your design choice in your report

Task 2: Experiment

- Task 2.2: Extensive evaluation of planner performance
  - By default, the Metric FF planner runs a faster – yet incomplete – heuristics-based algorithm (enforced hill-climbing) to solve a PDDL planning problem

  - If the Metric FF fails to find a legitimate plan at the first attempt, then it falls back to a more thorough heuristics-based best-first search to try again…
    - … using the following evaluation function for a state s:

$$f(s) = w_g g(s) + w_h h(s)$$

where $g(s)$ is the actual cost accumulated so far, $h(s)$ is the *estimated* cost hereafter from s to the goal, and w_g (default: 1) & w_h (default: 5) are integer weight parameter

Task 2: Experiment

- Task 2.2: Extensive evaluation of planner performance (cont'd)
  - Experiment question: How does our choice of (w_g, w_h) affect the planner performance?
  - Design a suite of experiments to evaluate the effect of setting different (w_g, w_h) values on the performance. Analyse the results in your report.
  - To start a single run with an experiment configuration, execute the following command on your shell:

```
./ff -E -g {value of w_g} -h {value of w_h} -o {domain_file_name} -f {problem_file_name}
```

Flag for disabling the default hill-climbing search algorithm

Task 3: Extensions

- One could argue the domain we have designed in Task 1 is rather too simple and still misses out many aspects of the real-world scenarios…

- Incrementally extend your domain to accommodate the following scenarios, and write problem files as specified in the handout:
  - Task 3.1: A MINEBOT now does not have unlimited power and holds a rechargeable battery with some quantitative energy capacity (hint: numeric fluents)
  - Task 3.2: Now there might be a natural gas leakage causing a fire on a cell, disabling MINEBOT to enter the cell before putting off the fire with a fire extinguisher object
  - Task 3.3: Now there might be 'larger' ores that are too heavy for a MINEBOT to lift by itself; such larger ores must be lifted and carried by TWO MINEBOTs.

Task 3: Extensions

- Extend your domain (cont'd):
  - Task 3.4: Extra challenge; freely motivate, design and implement another real-world-like aspect of your own, describe and justify in your report

- **A word of caution!!!**
  - **DO NOT ATTEMPT** this subtask unless:
    - The CW so far was a breeze, prior tasks took less than 12 hours to perfect
    - Your time could be better spent on other important tasks (e.g. refining answers to previous parts, other CWs, catching up on sleep)

  - Seriously, we will be incredibly picky when marking this part; we expect answers leading to an exciting, research-grade question to warrant a meaningful mark

Submission

- We will use Gradescope for both submitting and marking
- Follow the submission instruction in Gradescope as specified in the handout, having your submission files named accurately
- Autograder will verify the submitted PDDL files are syntactically valid and plans are generated in a reasonable timeframe
  - You can submit as many times as you want, up until submission deadline
  - Your latest submission will be marked
  - For late submission policy, consult the handout pdf
- Export your report as a pdf file

Getting support

- Labs
  - Weeks 7-9 in AT 6.06, at 11am-1pm
  - Exploit the lab sessions if you feel you could use some help from our demonstrators or fellow coursemates

- Piazza
  - Whenever you have questions, ask away; public questions are encouraged and may help other fellow students as well, though you can choose to post private questions only visible to course staff

- Please do not ask anyone to just give away solutions!
- Also, do not give away your solutions to anyone!

- Any questions?