make them reflect on their own learning, so this is good! In addition, many people who liked ungrading commented that it reduced the stress of trying to get every test case right.

There were many reasons people liked ungrading, chiefly:

1. Independence and autonomy: people felt more responsible for their own learning and appreciated the additional autonomy and independence that ungrading provided.
2. Reduced the stress of trying to get everything right and figure out what the staff wanted to see.
3. Allowed people to explore and take a few more risks, because if they failed, learning still occurred.

More generally, we saw several people either not submitting a self evaluation or not really taking the time to reflect on their learning. In this assignment, we make the expectations clearer: students are marked based on their self evaluation.

Collaboration

Assignment 2 can be completed individually or in pairs. We encourage pairs to work together to learn from each other; not to simply split the tasks for efficiency. But we will not monitor this -- it's your responsibility. You can submit different solutions or the same solution, we just want to encourage collaboration.

For the student works in pairs, you must submit an individual short self evaluation.

We encourage students to derive their own tests for the tasks and share them with others to help with learning.

The Domain - Bubble Tea robot

Your task is to model a robot that makes bubble tea. The owner of the company has just a few hard-coded recipes, but would prefer that customers are able to flexibly specify their own bubble teas.

The robot is able to do three things: (1) add ingredients to a cup; (2) mix the ingredients that are in the cup; and (3) heat the ingredients that are in the cup.

There are four main types of ingredients: (1) tapioca balls; (2) syrup; (3) ice; and (4) the tea itself. The model can add any of these to the cup at any point. Syrup and tapioca balls can be flavoured.

Customers can specify whether they would like their drink mixed or not (modelling as a PDDL predicate in a goal). If the tea and syrup are mixed, the drink is considered to be "mixed". However, if the tapioca balls are mixed, the mixer blends them into a syrup, so there are no longer tapioca balls in the drink.

When ice is added to the cup, the drink is considered to be iced. Otherwise, it is not iced (modelled as a PDDL goal).

The cup can be heated in a microwave. Typically, customers like heated tapioca balls (modelled as a PDDL goal), otherwise the tapioca balls are too hard. If the ice is heated, it melts so the drink is no longer iced. Customers can also specify that they would like the tea or the syrup iced as well (modelled as a PDDL goal).

The robot can make just one bubble tea at a time.

Tasks

You can use the classical propositional STRIPS subset of PDDL + typing, negative preconditions and conditional effects. You are responsible for creating the domain and problems files, specifying any types, predicates and actions (including action parameters, preconditions and effects), initial states, and goal states.

Task 1 [0 marks]

As a first task, model PDDL actions (can be one action or multiple) for adding the ingredients into a drink. Only one of each ingredient can be added into a single drink.

For this, you will also need to model the PDDL predicates, constants, etc.

Once you have actions modelled, specify problems to test the following:

A. A customer can create an iced drink with the tea, mango tapioca balls, and mango syrup.
B. A customer cannot create an iced drink with the tea, ice, and two different flavours of syrup.

Task 2 [0 marks]

The second task is to model heating the cup in the microwave, and mixing the ingredients of the cup.

Once you have actions modelled, specify problems to test the following:

A. A customer can create a mixed drink with the unheated tea, heated mango tapioca balls, unheated mango syrup, and no ice.

B. A customer can create a mixed drink with no tapioca balls, unheated mango syrup, unheated lime syrup, and ice. (Think carefully about this. If you have modelled the mixing properly, this should be possible).

You may need to change your actions from Task 1 for this. That's fine -- you should only submit the final versions.

Task 3 -- Challenge task [0 marks]

This task is a more challenging task that will take more time than the first two tasks, but you may wish to complete it if you want to learn more about PDDL.

The owners developing the robot have noticed that if the cup is heated in the microwave, the cup becomes too hot to handle. Fortunately, they have just upgraded the robot so that it can use two cups at a time.

Create a new version of your model that has two cups.

It is strongly recommended that you model this in a new domain file.

You should have the same actions as before, but this time, the actions apply to one of the cups. When heated in the microwave, the cup should be noted as being hot. In addition, the robot can tip all ingredients from one cup into the other, between either cup. This will tip all ingredients.

Once you have actions modelled, specify problems to test the following:

A. A customer can create a mixed drink with the unheated tea, heated mango tapioca balls, unheated mango syrup, no ice, and an unheated cup. This is the same as Task 2.a, but with an unheated cup.
B. A customer can create a mixed drink with no tapioca balls, unheated mango syrup, heated lime syrup, ice, and an unheated cup. This is the same as Task 2.b, but with an unheated cup and with heated lime syrup.

Task 4 -- Self evaluation [10 marks]

For each question, write a brief self evaluation that includes how many marks you believe that you have earnt in ths assignment. Your self evaluation for each question must include:

- Self evaluated marks for each question, with a breakdown of:
    - 3 marks for Task 1
    - 5 marks for Task 2
    - 2 marks for Task 3

- A brief overview of why you have earnt this. As part of your overview, consider the following:
    - whether you can successfully generate correct plans for the goals;
    - explain how you checked the correctness of your plans;
    - whether you believe that your model is robust -- that is, whether it can generate correct plans for goals other than those specified above and why you believe it is robust;
    - whether your model is clear, concise, and at the 'right' level of abstraction -- that is, the model contains only those aspects that are needed to solve the problem;
    - what you learnt about modelling and abstraction; and
    - list any assumptions that you have made.

For each task (1-3), keep this to 2-3 paragraphs.

Note that completing the tasks without submitting a self evaluation will not attract ANY marks. You must tell us the evidence that you use to self-evaluate yourself.

Ambiguous details?

You may have questions about the above description, such as: "what should happen if this happens, but then that -- what should the behaviour be?".

The answer: just choose a sensible behaviour and model it. We don't care if the model does exactly what our model does -- we care that you are learning about modelling and abstraction. You should be able to make a sensible assumption about the behaviour and model it. You will still learn the same amount!

Debugging tips and other resources

If the planner is failing to return a plan, but you think it should, debugging is more difficult than in imperative programming languages, because the model is declarative and there are no standard PDDL debuggers.

There are two techniques to help debug the model:

1. **Simplify the goal**: If your goal contains multiple predicates, try removing them systematically to see which goal (or combinations of goals) are preventing the solution. This helps to identify which actions you may have modelled incorrectly.

2. **Use preconditions as goals**: If tip 1 does not work, write out the plan that you expect it to find. Then, create a new problem instance, take the first action of your expected plan, and set its precondition as the goal. If it fails, you have identified where your model is not doing what you expect. If it does not fail, move to next action in the plan, setting its precondition as the goal, and iterate.

Don't forget there are a host of other resources to help with model building and debugging: https://canvas.lms.unimelb.edu.au/courses/151398/pages/links-to-ai-tools?module_item_id=4570190

Submission Instructions

Your submission has three parts: a session saved on planning.domains; a PDF file submission on Canvas; and a submission certification form.

planning.domains submission

Copy the domain and problem files into a session at https://tinyurl.com/comp90054.

From the menu, select Session --> Save --> Read only (link) to get the session URL. See a video below showing how to do this.

PDF report submission

Your submission should contain a report that includes:

1. Your unimelb student ID and student email address, and the name of anyone partner that you completed the assignment with.
2. A link to the URL of your planning.domain session at https://tinyurl.com/comp90054
3. The PDDL domain file(s), cut and paste into the PDF report.
4. The PDDL problem files, cut and paste into the PDF report.
5. The output generated by the problem files. That is, the plan if it generates a plan, or the output specifying that no plan was found.
6. Your self evaluation for each question.
7. An optional appendix, if you want to add additional details, such as additional problem files or solutions that you tried that didn't work out.