COMP90054 AI Planning for Autonomy - Project 2 Self Evaluation - PDDL modelling

Arya Araban – 1439683 – aaraban@student.unimelb.edu.au

Planning domains link

Problem 1:

self-evaluated mark: 3/3

In this problem, we aim to model the addition of ingredients into a cup. The ingredients can be either flavored (such as tapioca-balls and syrup) or unflavored (such as ice and syrup), which presents a challenge in creating a domain that accurately represents this. To address this challenge, we consider all possible ingredients as one set of constants of type “ingredient” and their flavors as another set of constants of type “flavor”, with possible flavors being mango and lime. We then model two separate actions: one for adding an unflavored ingredient into the cup and another for adding a flavored ingredient. The main difference between these actions is that adding a flavored ingredient requires an extra parameter for flavor.

The problem files for the tasks assume that all ingredients are initially outside of the cup, and we specify a goal using predicates. Using this model, I was able to obtain correct answers for both problem files: problem11.pddl produces a plan while problem12.pddl does not. To verify the correctness of the generated answers, I analyzed the output and went through each step based on the rules specified in the project description. This model is robust to unseen problems that aim to achieve the addition of ingredients into the cup, and has been tested for many different combinations. I also believe the model to be clear and abstract enough, since we aren’t introducing any unnecessary actions, and that the predicates and goals are also straightforward.

During the process of solving this problem, I came to the realization that a significant amount of effort and consideration must be invested in accurately modeling specific problems. In general, the task of modeling these problems is more complex than it initially appears. One of the main challenges I encountered while working on this problem was comprehending the syntax of PDDL and determining how to effectively utilize this syntax to achieve the desired goals.

Problem 2:

self-evaluated mark: 5/5

In this problem, we aim to incorporate the actions of mixing and heating the cup into the domain and achieve two tasks that utilize these actions. To model these two actions, several assumptions were made:

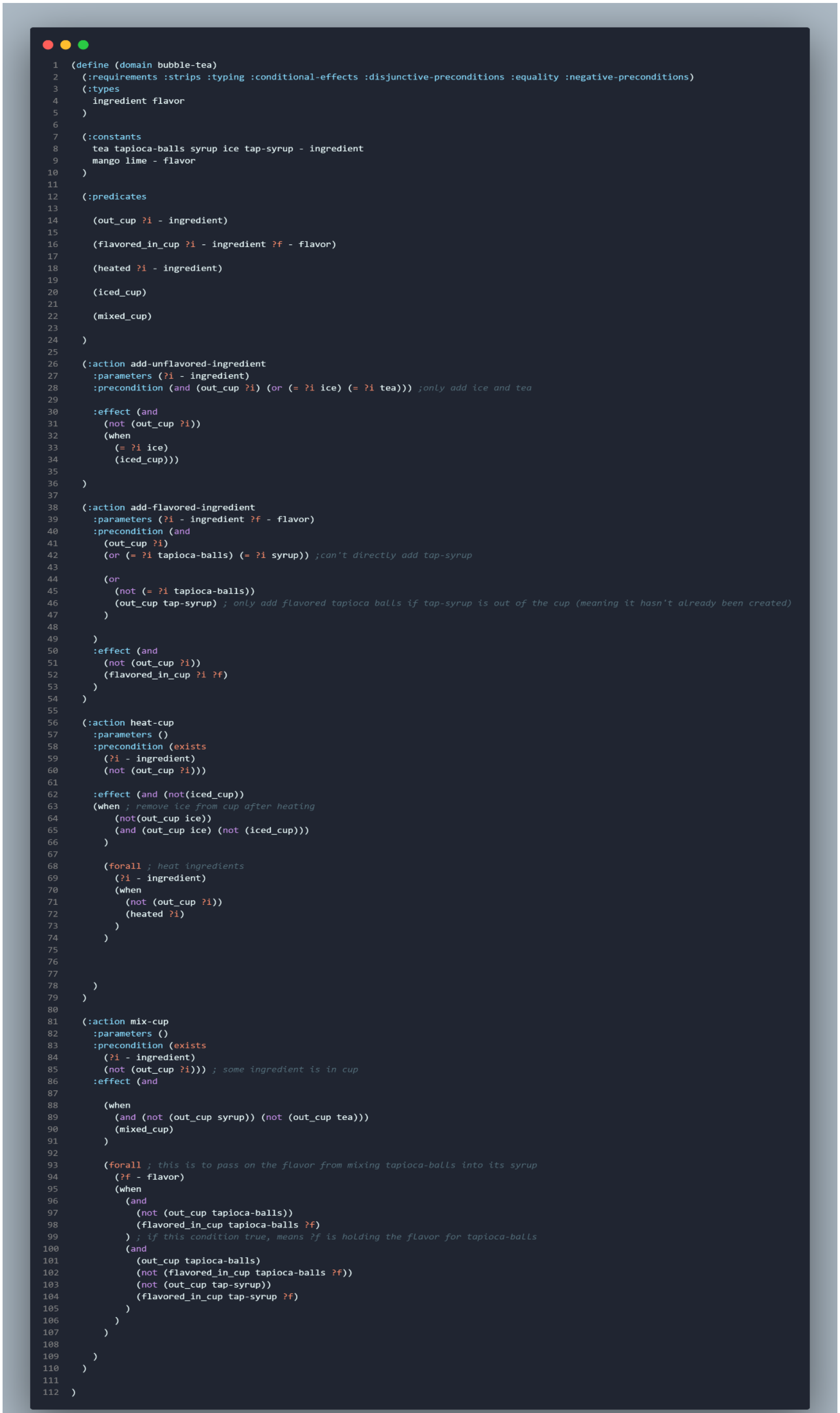
1- As in the previous problem, only one of each ingredient can be added to the cup.
2- If the cup contains ice, then heating the cup will cause the ice to melt and be removed from the cup, resulting in the cup no longer being considered "iced".
3- A cup is considered "mixed" only if the mixing action occurs when both tea and syrup are present in the cup.
4- Mixing tapioca-balls of any flavor will create a syrup of the same flavor. This syrup is distinct from the one that can be added directly, meaning that it is possible to have two syrups in the cup.
5- After tapioca-balls are converted into syrup, no additional tapioca-balls can be added.

Given these assumptions, it is impossible to find a plan for the first task of this problem (problem21.pddl), since if we examine the goal conditions and attempt to devise a plan, we will find that no plan exists. However, if we were to omit the third assumption and allow a cup to be considered "mixed" whenever mixing occurs, then a plan would be found. For the second task of this problem (problem22.pddl), a plan will be found, since we have assumed that two syrups can be inside the cup if one of those syrups comes from mixing the tapioca-balls. Similar to my answer to problem1, I analyzed the output and went through each step based on the rules specified in the project description in order to verify the solution's correctness.

I am confident that the model I have developed is both concise and unambiguous. This is due to the fact that every action and predicate has been explicitly defined, allowing the model to generate solutions for previously unseen problems. Furthermore, it has come to my attention that as the complexity of PDDL problems increases, it becomes increasingly important to exercise caution in the modeling process. This should be so to ensure that the model is at the right level of abstraction, so that the model doesn't contain non-useful information.

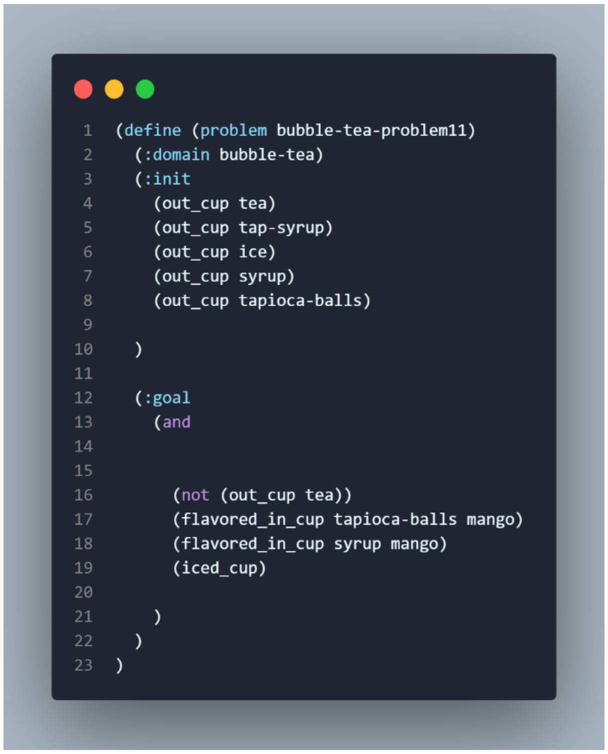
Problem 1 & 2 Files / Solutions:

bubble_tea_domain.pddl:

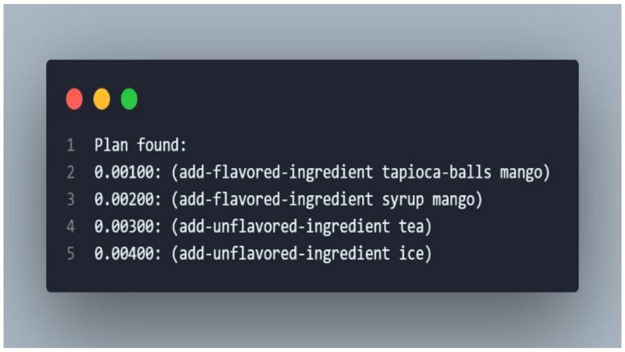


```
(define (domain bubble-tea)
  (:requirements :strips :typing :conditional-effects :disjunctive-preconditions :equality :negative-preconditions)
  (:types
    ingredient flavor
  )

  (:constants
    tea tapioca-balls syrup ice tap-syrup - ingredient
    mango lime - flavor
  )

  (:predicates

    (out_cup ?i - ingredient)

    (flavored_in_cup ?i - ingredient ?f - flavor)

    (heated ?i - ingredient)

    (iced_cup)

    (mixed_cup)

  )

  (:action add-unflavored-ingredient
    :parameters (?i - ingredient)
    :precondition (and (out_cup ?i) (or (= ?i ice) (= ?i tea))) ;only add ice and tea

    :effect (and
      (not (out_cup ?i))
      (when
        (= ?i ice)
        (iced_cup)))

  )

  (:action add-flavored-ingredient
    :parameters (?i - ingredient ?f - flavor)
    :precondition (and
      (out_cup ?i)
      (or (= ?i tapioca-balls) (= ?i syrup)) ;can't directly add tap-syrup

      (or
        (not (= ?i tapioca-balls))
        (out_cup tap-syrup) ; only add flavored tapioca balls if tap-syrup is out of the cup (meaning it hasn't already been created)
      )

    )
    :effect (and
      (not (out_cup ?i))
      (flavored_in_cup ?i ?f)
    )
  )

  (:action heat-cup
    :parameters ()
    :precondition (exists
      (?i - ingredient)
      (not (out_cup ?i)))

    :effect (and (not(iced_cup))
    (when ; remove ice from cup after heating
        (not(out_cup ice))
        (and (out_cup ice) (not (iced_cup)))
      )

      (forall ; heat ingredients
        (?i - ingredient)
        (when
          (not (out_cup ?i))
          (heated ?i)
        )
      )



    )
  )

  (:action mix-cup
    :parameters ()
    :precondition (exists
      (?i - ingredient)
      (not (out_cup ?i))) ; some ingredient is in cup
    :effect (and

      (when
        (and (not (out_cup syrup)) (not (out_cup tea)))
        (mixed_cup)
      )

      (forall ; this is to pass on the flavor from mixing tapioca-balls into its syrup
        (?f - flavor)
        (when
          (and
            (not (out_cup tapioca-balls))
            (flavored_in_cup tapioca-balls ?f)
          ) ; if this condition true, means ?f is holding the flavor for tapioca-balls
          (and
            (out_cup tapioca-balls)
            (not (flavored_in_cup tapioca-balls ?f))
            (not (out_cup tap-syrup))
            (flavored_in_cup tap-syrup ?f)
          )
        )
      )

    )
  )

)
```

Problem11.pddl:

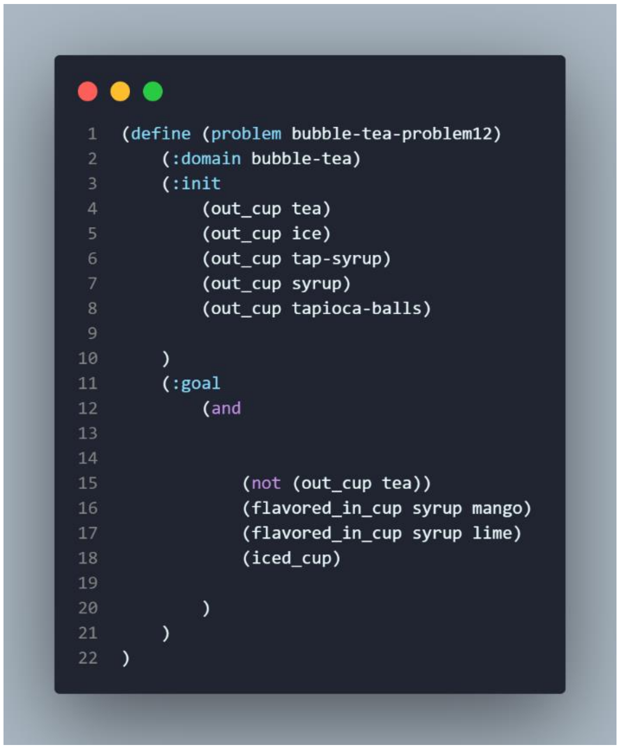


```
(define (problem bubble-tea-problem11)
  (:domain bubble-tea)
  (:init
    (out_cup tea)
    (out_cup tap-syrup)
    (out_cup ice)
    (out_cup syrup)
    (out_cup tapioca-balls)

  )

  (:goal
    (and


      (not (out_cup tea))
      (flavored_in_cup tapioca-balls mango)
      (flavored_in_cup syrup mango)
      (iced_cup)

    )
  )
)
```

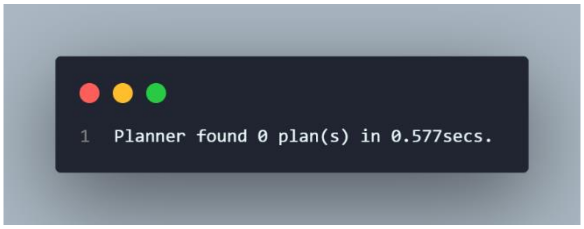


```
Plan found:
0.00100: (add-flavored-ingredient tapioca-balls mango)
0.00200: (add-flavored-ingredient syrup mango)
0.00300: (add-unflavored-ingredient tea)
0.00400: (add-unflavored-ingredient ice)
```

Problem12.pddl:

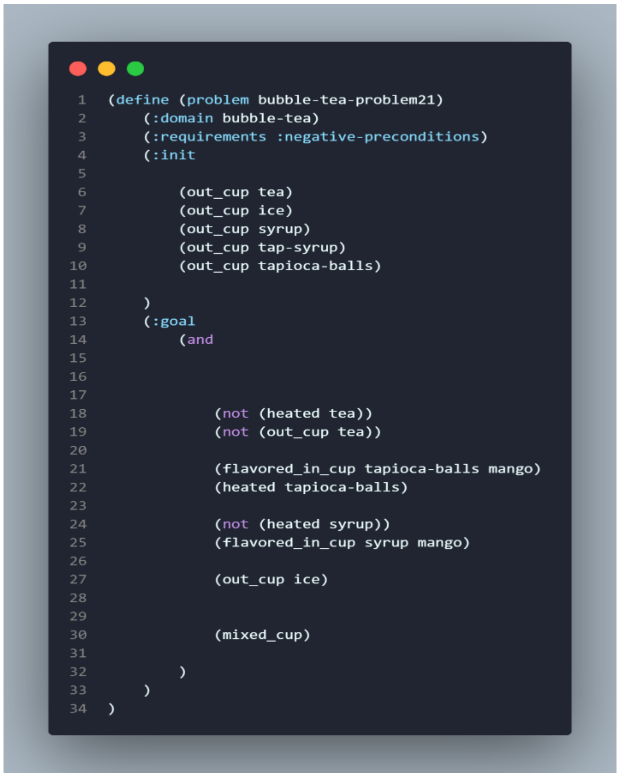


```
(define (problem bubble-tea-problem12)
    (:domain bubble-tea)
    (:init
        (out_cup tea)
        (out_cup ice)
        (out_cup tap-syrup)
        (out_cup syrup)
        (out_cup tapioca-balls)

    )
    (:goal
        (and


            (not (out_cup tea))
            (flavored_in_cup syrup mango)
            (flavored_in_cup syrup lime)
            (iced_cup)

        )
    )
)
```

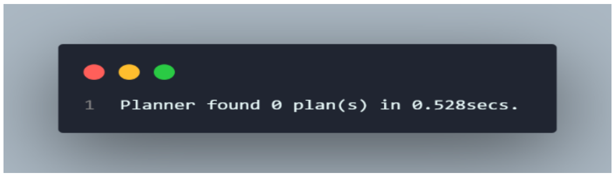


```
Planner found 0 plan(s) in 0.577secs.
```

Problem21.pddl:

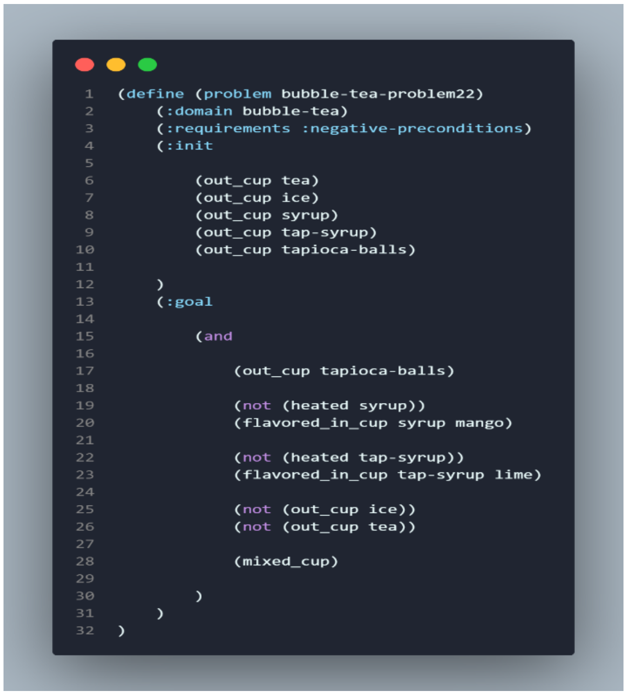


```
(define (problem bubble-tea-problem21)
    (:domain bubble-tea)
    (:requirements :negative-preconditions)
    (:init

        (out_cup tea)
        (out_cup ice)
        (out_cup syrup)
        (out_cup tap-syrup)
        (out_cup tapioca-balls)

    )
    (:goal
        (and



            (not (heated tea))
            (not (out_cup tea))

            (flavored_in_cup tapioca-balls mango)
            (heated tapioca-balls)

            (not (heated syrup))
            (flavored_in_cup syrup mango)

            (out_cup ice)


            (mixed_cup)

        )
    )
)
```

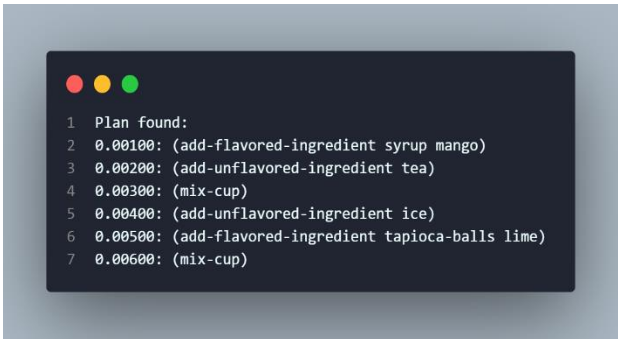


```
Planner found 0 plan(s) in 0.528secs.
```

Problem22.pddl:



```
(define (problem bubble-tea-problem22)
    (:domain bubble-tea)
    (:requirements :negative-preconditions)
    (:init

        (out_cup tea)
        (out_cup ice)
        (out_cup syrup)
        (out_cup tap-syrup)
        (out_cup tapioca-balls)

    )
    (:goal

        (and

            (out_cup tapioca-balls)

            (not (heated syrup))
            (flavored_in_cup syrup mango)

            (not (heated tap-syrup))
            (flavored_in_cup tap-syrup lime)

            (not (out_cup ice))
            (not (out_cup tea))

            (mixed_cup)

        )
    )
)
```



```
Plan found:
0.00100: (add-flavored-ingredient syrup mango)
0.00200: (add-unflavored-ingredient tea)
0.00300: (mix-cup)
0.00400: (add-unflavored-ingredient ice)
0.00500: (add-flavored-ingredient tapioca-balls lime)
0.00600: (mix-cup)
```

Problem 3:

self-evaluated mark: 2/2

The third problem presented a greater challenge as it involved a more complex scenario in which two cups were utilized instead of one. Nevertheless, due to the concision of the domain model, not many modifications were required to adapt it to this new scenario. In the domain file we specify two new constants of type cup, and we also modify each of the actions so that they take in a new parameter “cup” which specifies on which cup the action should be performed on. A new action has been introduced to facilitate the transfer of ingredients from one cup to another. All assumptions from Problem 2 remain applicable to each cup of this problem. In addition, the following assumptions have also been introduced:

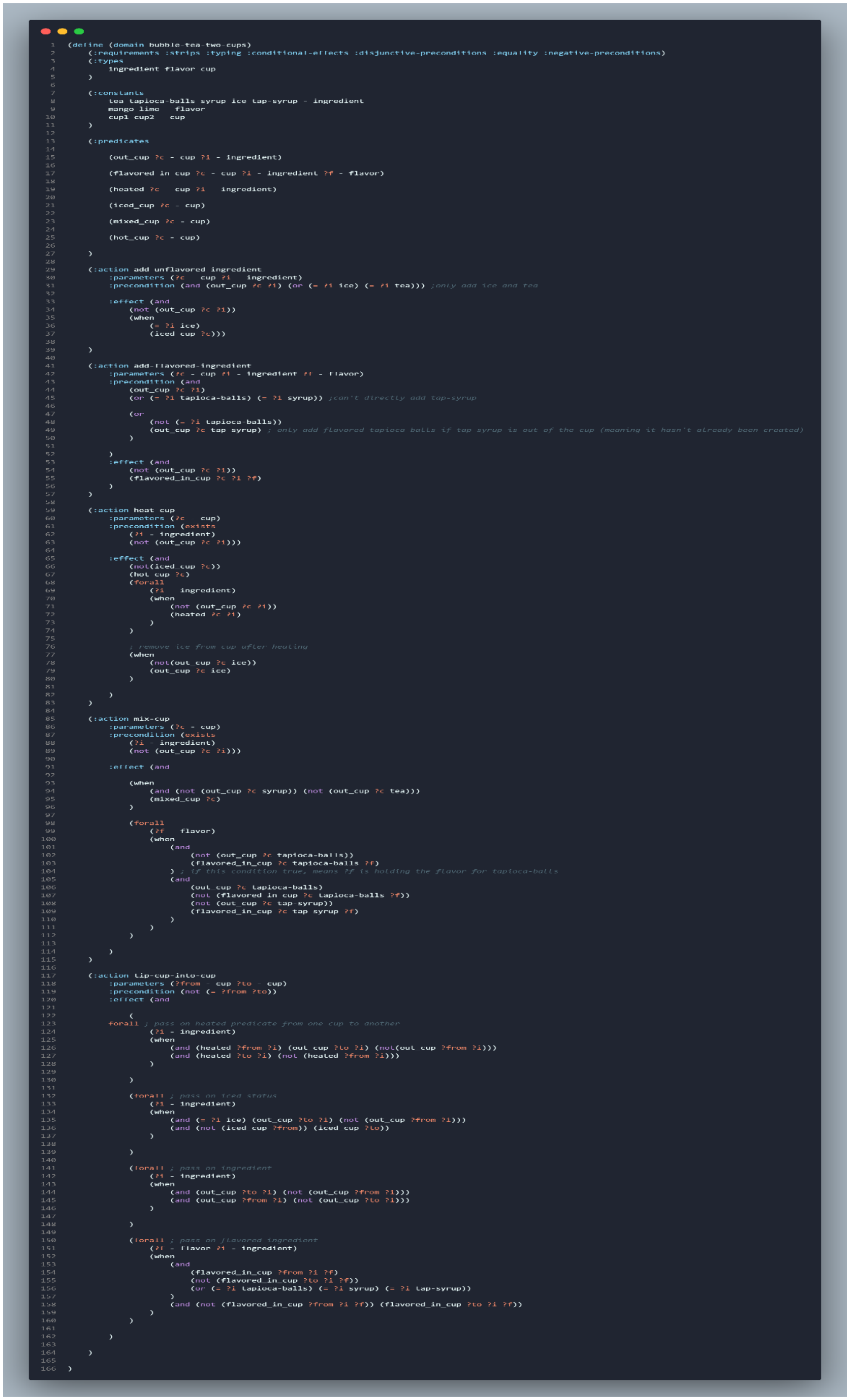
1- A new predicate has been added to track whether the cup itself has been heated.
2- When tipping one cup into the other, only the ingredients of the first cup which are not already in the second cup will be transferred.
3- The heat of the transferred ingredients will be retained.
4- If Ice is among the transferred ingredients, then the “iced” status of the cup will be transferred as well.

Based on these assumptions, the model is able to identify plans for both tasks associated with this problem. As before, in order to check the correctness of these solutions, I analyzed the output and went through each step based on the rules specified in the project description.

I am also rather confident that the model will be able to find plans for unseen problems related to the domain, since we have clear actions, while also having clear assumptions for those actions.

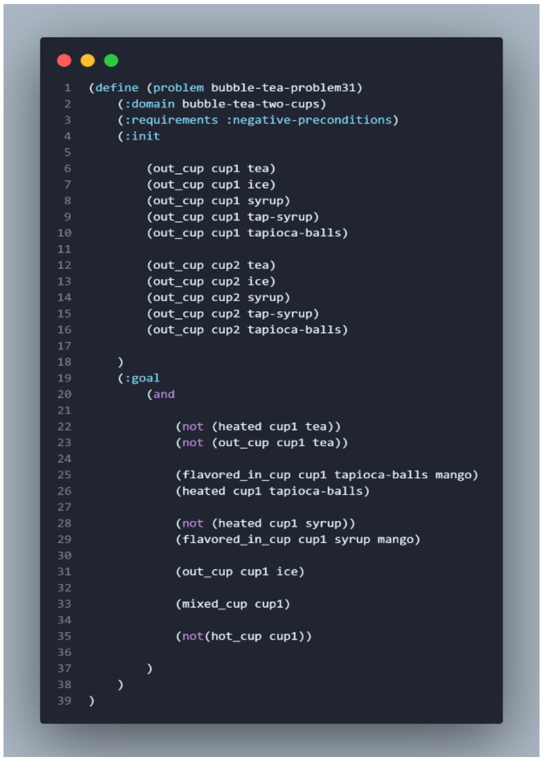
Problem 3 Files / Solutions:

bubble_tea_two_cups_domain.pddl:

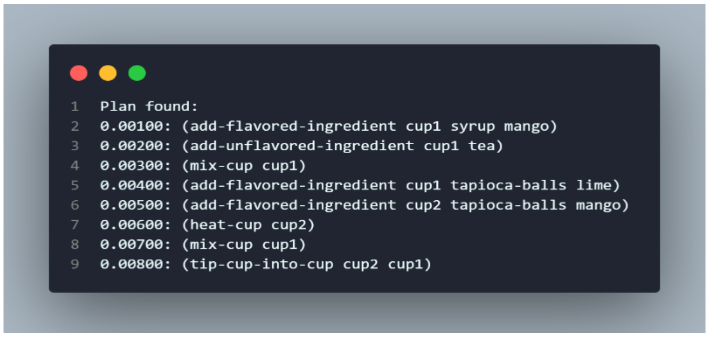


```
(define (domain bubble-tea-two-cups)
    (:requirements :strips :typing :conditional-effects :disjunctive-preconditions :equality :negative-preconditions)
    (:types
        ingredient flavor cup
    )

    (:constants
        tea tapioca-balls syrup ice tap-syrup - ingredient
        mango lime - flavor
        cup1 cup2 - cup
    )

    (:predicates

        (out_cup ?c - cup ?i - ingredient)

        (flavored in cup ?c - cup ?i - ingredient ?f - flavor)

        (heated ?c - cup ?i - ingredient)

        (iced_cup ?c - cup)

        (mixed_cup ?c - cup)

        (hot_cup ?c - cup)

    )

    (:action add unflavored ingredient
        :parameters (?c - cup ?i - ingredient)
        :precondition (and (out_cup ?c ?i) (or (= ?i ice) (= ?i tea))) ;only add ice and tea

        :effect (and
            (not (out_cup ?c ?i))
            (when
                (= ?i ice)
                (iced cup ?c)))

    )

    (:action add-flavored-ingredient
        :parameters (?c - cup ?i - ingredient ?f - flavor)
        :precondition (and
            (out_cup ?c ?i)
            (or (= ?i tapioca-balls) (= ?i syrup)) ;can't directly add tap-syrup

            (or
                (not (= ?i tapioca-balls))
                (out_cup ?c tap syrup) ; only add flavored tapioca balls if tap syrup is out of the cup (meaning it hasn't already been created)
            )

        )
        :effect (and
            (not (out_cup ?c ?i))
            (flavored_in_cup ?c ?i ?f)
        )
    )

    (:action heat cup
        :parameters (?c - cup)
        :precondition (exists
            (?i - ingredient)
            (not (out_cup ?c ?i)))

        :effect (and
            (not(iced_cup ?c))
            (hot cup ?c)
            (forall
                (?i - ingredient)
                (when
                    (not (out_cup ?c ?i))
                    (heated ?c ?i)
                )
            )

            ; remove ice from cup after heating
            (when
                (not(out_cup ?c ice))
                (out_cup ?c ice)
            )

        )
    )

    (:action mix-cup
        :parameters (?c - cup)
        :precondition (exists
            (?i - ingredient)
            (not (out_cup ?c ?i)))

        :effect (and

            (when
                (and (not (out_cup ?c syrup)) (not (out_cup ?c tea)))
                (mixed_cup ?c)
            )

            (forall
                (?f - flavor)
                (when
                    (and
                        (not (out_cup ?c tapioca-balls))
                        (flavored_in_cup ?c tapioca-balls ?f)
                    ) ; if this condition true, means ?f is holding the flavor for tapioca-balls
                    (and
                        (out_cup ?c tapioca-balls)
                        (not (flavored in cup ?c tapioca-balls ?f))
                        (not (out_cup ?c tap-syrup))
                        (flavored_in_cup ?c tap syrup ?f)
                    )
                )
            )

        )
    )

    (:action tip-cup-into-cup
        :parameters (?from - cup ?to - cup)
        :precondition (not (= ?from ?to))
        :effect (and

            (
        forall ; pass on heated predicate from one cup to another
                (?i - ingredient)
                (when
                    (and (heated ?from ?i) (out cup ?to ?i) (not(out cup ?from ?i)))
                    (and (heated ?to ?i) (not (heated ?from ?i)))
                )

            )

            (forall ; pass on iced status
                (?i - ingredient)
                (when
                    (and (= ?i ice) (out_cup ?to ?i) (not (out_cup ?from ?i)))
                    (and (not (iced cup ?from)) (iced cup ?to))
                )

            )

            (forall ; pass on ingredient
                (?i - ingredient)
                (when
                    (and (out_cup ?to ?i) (not (out_cup ?from ?i)))
                    (and (out_cup ?from ?i) (not (out_cup ?to ?i)))
                )

            )

            (forall ; pass on flavored ingredient
                (?f - flavor ?i - ingredient)
                (when
                    (and
                        (flavored_in_cup ?from ?i ?f)
                        (not (flavored_in_cup ?to ?i ?f))
                        (or (= ?i tapioca-balls) (= ?i syrup) (= ?i tap-syrup))
                    )
                    (and (not (flavored_in_cup ?from ?i ?f)) (flavored_in_cup ?to ?i ?f))
                )
            )

        )

    )

)
```

Problem31.pddl:

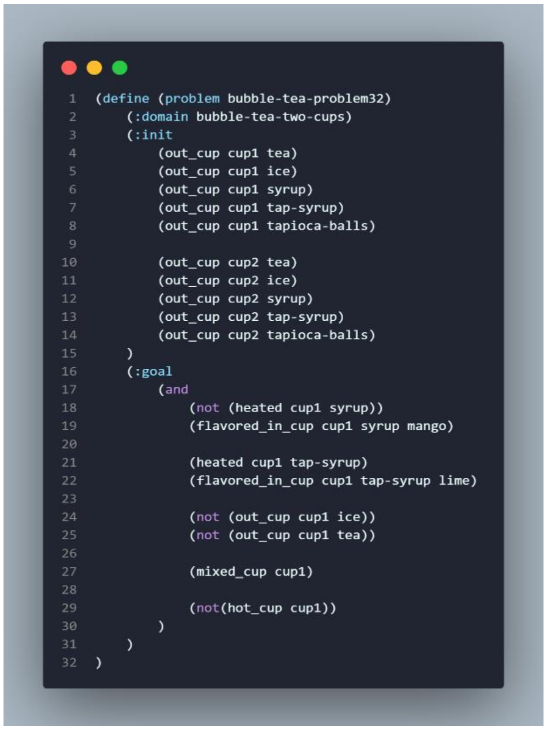


```
(define (problem bubble-tea-problem31)
    (:domain bubble-tea-two-cups)
    (:requirements :negative-preconditions)
    (:init

        (out_cup cup1 tea)
        (out_cup cup1 ice)
        (out_cup cup1 syrup)
        (out_cup cup1 tap-syrup)
        (out_cup cup1 tapioca-balls)

        (out_cup cup2 tea)
        (out_cup cup2 ice)
        (out_cup cup2 syrup)
        (out_cup cup2 tap-syrup)
        (out_cup cup2 tapioca-balls)

    )
    (:goal
        (and

            (not (heated cup1 tea))
            (not (out_cup cup1 tea))

            (flavored_in_cup cup1 tapioca-balls mango)
            (heated cup1 tapioca-balls)

            (not (heated cup1 syrup))
            (flavored_in_cup cup1 syrup mango)

            (out_cup cup1 ice)

            (mixed_cup cup1)

            (not(hot_cup cup1))

        )
    )
)
```

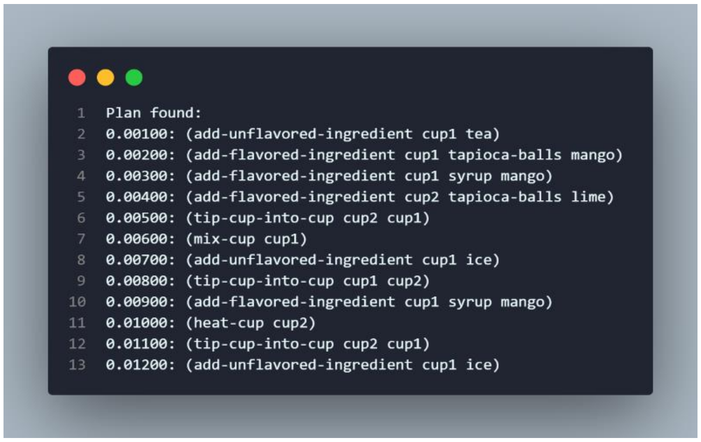


```
Plan found:
0.00100: (add-flavored-ingredient cup1 syrup mango)
0.00200: (add-unflavored-ingredient cup1 tea)
0.00300: (mix-cup cup1)
0.00400: (add-flavored-ingredient cup1 tapioca-balls lime)
0.00500: (add-flavored-ingredient cup2 tapioca-balls mango)
0.00600: (heat-cup cup2)
0.00700: (mix-cup cup1)
0.00800: (tip-cup-into-cup cup2 cup1)
```

Problem32.pddl:



```
(define (problem bubble-tea-problem32)
    (:domain bubble-tea-two-cups)
    (:init
        (out_cup cup1 tea)
        (out_cup cup1 ice)
        (out_cup cup1 syrup)
        (out_cup cup1 tap-syrup)
        (out_cup cup1 tapioca-balls)

        (out_cup cup2 tea)
        (out_cup cup2 ice)
        (out_cup cup2 syrup)
        (out_cup cup2 tap-syrup)
        (out_cup cup2 tapioca-balls)
    )
    (:goal
        (and
            (not (heated cup1 syrup))
            (flavored_in_cup cup1 syrup mango)

            (heated cup1 tap-syrup)
            (flavored_in_cup cup1 tap-syrup lime)

            (not (out_cup cup1 ice))
            (not (out_cup cup1 tea))

            (mixed_cup cup1)

            (not(hot_cup cup1))
        )
    )
)
```



```
Plan found:
0.00100: (add-unflavored-ingredient cup1 tea)
0.00200: (add-flavored-ingredient cup1 tapioca-balls mango)
0.00300: (add-flavored-ingredient cup1 syrup mango)
0.00400: (add-flavored-ingredient cup2 tapioca-balls lime)
0.00500: (tip-cup-into-cup cup2 cup1)
0.00600: (mix-cup cup1)
0.00700: (add-unflavored-ingredient cup1 ice)
0.00800: (tip-cup-into-cup cup1 cup2)
0.00900: (add-flavored-ingredient cup1 syrup mango)
0.01000: (heat-cup cup2)
0.01100: (tip-cup-into-cup cup2 cup1)
0.01200: (add-unflavored-ingredient cup1 ice)
```

Upon examination of the solution for problem32, it becomes evident that the proposed plan is not optimal due to the presence of redundant actions. Nevertheless, taking into account the established assumptions, particularly the second assumption, it can be concluded that the plan remains valid.