Kings College London

4CCS1IAI Introduction to Artificial Intelligence

AI Planning Application Coursework
Deliverable 1: Planning Model

Wonjoon Seol (k1631098)
Munkhtulga Battogtokh (k1631010)
Britton Forsyth (k1630500)
Eugene Fong (k1630435)

Supervised by
Dr.Dan Magazzeni
Dr.Lela Koulouri

April 21, 2017

1 Deliverable 1: Planning Model

1.1 Introduction

We have considered a simplified model of restaurant staff management for our domain, and have picked sample managing problem instances that we have identified as common to most average scale restaurants. Deliverable section 1 focuses on describing this planning model, whereas deliverable section 2 on the other hand focuses on planning solutions that our application can offer.

Extension of our model in the Planning Domain Definition Language (PDDL) could well be meaningful for commercial use to speed-up service, and reduce human-mistakes, and in the long-run cut labour cost for traditional style restaurants with human service (as opposed to self-service or take-away food stores). While there are already technologies commercially utilised, by for example McDonalds, to simplify self-service operations, there are not yet many corresponding alternatives for those who value the warmth of a human service. This unsatisfied demand has been the creative incentive behind our application.

For practicality of the application, we have used our self-experiences working in the Korean restaurant Bibimbap, located in Charlotte street, as reference to suggest the essentials of the domain: the objects (e.g. tables, or group of customers), and the actions (e.g. to seat customers, or to take order from them).

The managing domain generally concerns maximum customer satisfaction as goal, and the correct steering of the staff to achieve this. From among several options, for simplicity and practicality, we assume that minimising the time with which the customers are served ensures our goal: customer satisfaction. This suits the abilities, and the purpose of the planner OPTIC, which is for domains where time is essential.

Samples for more particular problem scenarios are offered in section 2. They include thematic scenarios like receiving large groups that exceed any table capacity, or overwhelming number of customer groups for very few staff.

In conclusion, as result of experimenting with suitability of the PDDL syntax, and capability of the OPTIC planner as well as of analysing the restaurant managing domain in real-life, we would say that we have constructed this application to be interesting for both reality, and planning.

1.2 Breakdown of domain model in PDDL

The spirit of our domain model is to keep minimalistically simple (yet so rich) while capturing the most essential of the elements that constitute managing a restaurant. In order to do this, we have defined set of object types, predicates, functions, and durative actions as enabled by PDDL2.1 syntax. From here will follow brief pseudocode descriptions for each individual instances of these in our model.

- Types:

  group - object that corresponds to a group of customers
  table - object that corresponds to a table on which customers may be seated
  staffmember - object that corresponds to a member of staff of the restaurant, e.g. a waiter

- Predicates:

  (waiting-table ?g - group)
  States that the group of customers given as parameter is waiting to be seated.
  (table-available ?t - table)
  States that the table given as parameter is available for customers to be seated on.

(member-available ?m - staffmember)
States that the member of staff given as parameter is available to take appropriate actions.
(seated ?g - group ?t - table)
States that the given group is seated on the given table
(waiting-order ?g - group)
States that the given group is waiting to order
(ordered ?g - group ?t - table)
States that the given group on the given table has ordered
(served ?g - group)
States that the given group has been served their orders
(not-served ?g - group)
States that the given group has not been served their orders.
(eaten ?g - group)
States that the given group has eaten their dishes
(need-clean ?t - table ?g - group)
States that the given table where the given group has dined requires cleaning
(group-complete ?g - group)
States that the given group has completed dining from the resturant and leaving

- Functions:

(people-count ?g - group)
Returns the number of people in the given group
(table-capacity ?t - table)
Returns how many people the given table can accomodate
(table-id ?t - table)
Returns an identifying number for the given table.

- Durative actions:

(:durative-action seat-group)
Takes a staffmember, a group, and a table as parameters. The staffmember lets the group of customers take their seats on the given table if the table capacity can accomodate the number of people in the group. Duration is 30 seconds per customer.
(:durative-action let-decide-order)
Takes a group and a table as parameters. Given that the group is seated, action is to give them time to decide their orders. Duration is 30 seconds per customer.
(:durative-action take-order)
Takes a staffmember, a group, and a table as parameters. Given that the group has decided their orders, and are seated, the staffmember takes the table's orders. Duration is 30 seconds per person.
(:durative-action serve)
Takes a staffmember, a group, and a table as parameters. The staffmember serves the orders to the group on the given table. Duration is 100 seconds per a customer seated at the table.
(:durative-action let-eat)
Takes a group as a parameter. Action is to give the group enough time to finish consuming their orders.
(:durative-action take-payment)
Takes a staffmember, a group, and a table as parameters. The staffmember takes payment from the given group seated on the given table, after which the customers leave. Duration is 60 seconds per customer
(:durative-action clear-table)
Takes a staffmember and a tables as parameters. The staffmember cleans the given table where the given group is seated. The table is now ready to take a new group. Duration is 30 seconds per table.

1.3 Summary of the planning model

We have told in the introduction that our application is meaningful to certain style of restaurant operation. Here, we highlight how AI planning is deemed suitable and used.

The pseudocode above suggests that the actions work in chain, and cycles: a group must be seated before ordering, after that they can be served, and so on until they can finally leave the restaurant satisfied. After they have left, the next customers gets seated, and so on in cycles. The way how results of one action satisfy the preconditions of the next action, is taken from the real world, and fits well with PDDL definition syntax. More importantly, the restaurant managing domain has the simple goal of ensuring customer satisfaction, which is easily modelled in planning. There are many possibilities for an initial state (some groups seated, some not, some group with few people etc.), and in reality, it requires quick, and precise planning to make sure no group leaves unsatisfied (and writes a negative online review afterwards), which could utilise the high computational-power of a computer.

However, note that our application is premature. For practical use, extensions such as a way to provide the planner with constantly updated problem-file, or to ensure that the planner does not unnecessarily change the plan midway are required. There are also alternative ways to encode the domain in PDDL to increase for example computational efficiency, or realism. For this purpose, we have been incrementally improving our code (see Appendix - Deliverable 1 for latest version).

2 Appendix

2.1 Domain file

Domain part 1: Restaurant Management Domain

```
(define  (domain restaurant-managing)
    (:requirements :typing :durative-actions :numeric-fluents)
    (:types group table staffmember - object)
    (:predicates
        ;General table − related states
        (waiting-table ?g - group)
        (table-available ?t - table)
        (member-available ?m - staffmember)
        (seated ?g - group ?t - table)
        (waiting-order ?g - group)
        (ordered ?g - group ?t - table)
        ;Serving related states
        (served ?g - group)
        (not-served ?g - group)
        (eaten ?g - group)
        (need-clean ?t - table ?g - group)
        (group-complete ?g -group)
    )
    (:functions
        (people-count ?g - group)
        (table-capacity ?t - table)
        (table-id ?t - table)
    )
    (:durative-action seat-group
        :parameters (?m - staffmember ?g - group ?t - table)
        :duration (= ?duration 30)
        :condition (and (at start (waiting-table ?g))
                        (over all (table-available ?t))
                        (at start (member-available ?m))
                        (at start (<= (people-count ?g) (table-capacity ?t)))
                    )
        :effect (and (at start (not (member-available ?m)))
                    (at end (member-available ?m))
                    (at end (seated ?g ?t))
                    (at end (member-available ?m))
                    (at end (not (table-available ?t)))
                    (at end (not-served ?g))
                    (at end (not (waiting-table ?g)))
                )
    )
    ;Once seated, 30 seconds per person is needed for a group to prepare for ordering
     (:durative-action let-decide-order
        :parameters (?g - group ?t - table)
        :duration (= ?duration (* (people-count ?g) 30))
        :condition (and (over all (not-served ?g))
                        (over all (seated ?g ?t))
                    )
        :effect (at end (waiting-order ?g))
    )
```

Domain part 2: Restaurant Management Domain

```
1   ; In this simplified model, 30 seconds are spent on each person to take order
     (:durative-action take-order
2       :parameters (?m - staffmember ?g - group ?t - table)
3       :duration (= ?duration (* (people-count ?g) 30))
4       :condition (and (at start (member-available ?m))
5                       (at start (seated ?g ?t))
6                       (at start (waiting-order ?g))
7                  )
8       :effect (and (at start (not (member-available ?m)))
9                    (at end (member-available ?m))
10                   (at end (ordered ?g ?t))
11             )
12  )
13  ; Assumes that every goods served are ready and served in 100s per person
14  (:durative-action serve
15      :parameters (?m - staffmember ?g - group ?t - table)
16      :duration (= ?duration (* (people-count ?g) 100))
17      :condition (and (at start (seated ?g ?t))
18                      (at start (ordered ?g ?t))
19                      (at start (not-served ?g))
20                      (at start (member-available ?m))
21                 )
22      :effect (and (at start (not (not-served ?g)))
23                   (at end (served ?g))
24                   (at start (not (member-available ?m)))
25                   (at end (member-available ?m))
26             )
27  )
28  ; Assumes that every goods served are ready and served in 100s per person
29  (:durative-action let-eat
30      :parameters (?g - group)
31      :duration (= ?duration (* (people-count ?g) 100))
32      :condition (at start (served ?g))
33      :effect (at end (eaten ?g))
34  )
35  (:durative-action take-payment
36      :parameters (?m - staffmember ?g - group ?t - table)
37      :duration (= ?duration (* (people-count ?g) 60) )
38      :condition (and (at start (seated ?g ?t))
39                      (at start (member-available ?m))
40                      (at start (eaten ?g))
41                 )
42      :effect (and (at start (not (seated ?g ?t)))
43                   (at start (not (member-available ?m)))
44                   (at end (member-available ?m))
45                   (at end (need-clean ?t ?g))
46             )
47  )
```

Domain part 3: Restaurant Management Domain

```
    (:durative-action clear-table
        :parameters (?m - staffmember ?g - group ?t - table)
        :duration (= ?duration 30)
        :condition (and (at start (member-available ?m))
                        (at start (need-clean ?t ?g))
                   )
        :effect (and (at start (not (seated ?g ?t)))
                     (at end (table-available ?t)
                     (at start (not (member-available ?m)))
                     (at end (member-available ?m))
                     (at end (group-complete ?g))
                     (at end (not (need-clean ?t ?g)))
                )
    )
)
```