



Group 3: Downtown Dining

Amy Hong (19YH94)

Eric Lam (19EWHL1)

Lucy Woloszczuk (19LEW3)

Wesley Lam (19WESW)

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

December 13, 2021

Abstract

In this project, logic will be used to model the selection of an ideal restaurant among a party that includes 2-8 members. This model will use a priority system to account for both the dietary restrictions (higher priority) and the preferences (lower priority) for each group member.

Propositions

Party size: N - The number of people in the party, N can contain values 2-8

Time: t - The hour that the party needs to eat, can contain values 0 - 23

Weekday: d - The weekday that the party needs to eat, can contain values 1-7

Allowable - $A(r)$: The set of all restaurants that are allowed.

Person (restrictions, preferences): Represents each person in the group students

Restrictions (dietary, price): The list of conditions that a restaurant must satisfy

Dietary: all restrictions related to food - can be vegetarian, dairy-free, halal

- V_p : is true if a person is vegetarian, p represents the person
- D_p : is true if a person is lactose intolerant
- H_p : is true if a person requires halal food
- V_r : is true if a restaurant has vegetarian options, r represents the restaurant
- D_r : is true if a restaurant has dairy-free options
- H_r : is true if a restaurant has halal options

Price: Preferred price sign displayed by Google (\$, \$\$, \$\$\$, \$\$\$)

- P_{rj} : is true if the dollar sign on Google matches the person's preferred price, where j represents the number of dollar signs

Rating: How many stars does the person require the restaurant to have on Google - can be 1, 2, 3, 4, 5

- R_{pk} : is true if the number of stars is greater than or equal to k , where k is the number of stars

Seating: Does the person want patio seating?

- S_p : is true if patio seating is wanted, false if not

Location: The location of the person or restaurant

-
- L_{pd} : Is true if the person is located at location "d", where "d" is the distance from the starting location.

Price: Price sign displayed by google (\$, \$\$, \$\$\$, \$\$\$\$)

- P_{rk} : is true if the restaurant bill per person is lower than j, where j is the bill price per person

Rating: How many stars the restaurant has on Google - can be 1, 2, 3, 4, 5

- R_{rk} : is true if the number of stars is equal to k, where k is the number of stars

Seating: Is patio seating available?

- S_r : is true if patio seating is available, false if not

Location: The location of the restaurant is on Princess

- L_{ra} : Is true if the restaurant is located at location "a", where "a" is the address.

Open: Is the restaurant considered open?

- O_r : Is true if the restaurant is currently open.

Open Time: Is the restaurant open during the hour where the party eats?

- OT_{rt} : Is true if the restaurant is open at time t (The hour that the party needs to eat)

Open Day: Is the restaurant open during the weekday where the party eats?

- OD_{rd} : Is true if the restaurant is open on the day, d (The day that the party needs to eat)

Weather: The current weather conditions

- *Sunny*: Is true if it is sunny
- *Raining*: Is true if it is raining
- *Snowing*: Is true if it is snowing

Constraints

Rating Constraints:

- Each restaurant can only have one rating at a time
 $k = 1 = 1 \text{ star}, k = 2 = 2 \text{ star}, \dots$
 $\forall r (R_{r,k} \rightarrow \neg R_{r,k+1} \wedge \neg R_{r,k+2} \dots \wedge \neg R_{r,k-1} \dots)$

Price Constraints:

- Each restaurant can only have one price at a time
 $k = 1 = \$, k = 2 = \$\$, k = 3 = \$\$\$, k = 4 = \$\$\$\$$
 $\forall r (P_{r,k} \rightarrow \neg P_{r,k+1} \wedge \neg P_{r,k+2} \dots \wedge \neg P_{r,k-1} \dots)$

Location Constraints:

- Each restaurant has exactly one unique address a
 $\exists a L_{r,a} \wedge \forall b (L_{r,b} \rightarrow a = b)$
- Each restaurant must be located on n Princess St. where n is the address number
 $L_{r,a} \rightarrow a = "n \text{ Princess St.}"$

Opening constraints:

- Restaurants must be open on the day and hour the party eats.
 $\forall r (O_r \rightarrow (OD_{rd} \wedge OT_{rt}))$

Dietary Constraints:

- All allowable restaurants must not violate any dietary constraints. For example, if a person in the group is vegetarian all allowable restaurants must be vegetarian.
 $\forall r \exists p (A(r) \rightarrow ((V_p \rightarrow V_r) \wedge (D_p \rightarrow D_r) \wedge (H_p \rightarrow H_r)))$

Weather Constraints:

- If it is sunny outside, it cannot be raining
 $Sunny \rightarrow \neg Raining$
- If it is raining or snowing, patio seating is not available
 $(Raining \vee Snowing) \rightarrow \neg S_r$

Model Exploration

List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.

One of our Service propositions wasn't coming up in any solutions unless specifically required to true (it instead defaults to False). We tried implementing different workarounds but currently stumped.

```

88     @constraint.at_least_one(E)
89     @proposition(E)
90     class Service:
91
92     def __init__(self, data):
93         self.data = data
94
95     def __repr__(self):
96         return f"{self.data}"
97
98
99     # dine-in, take-out, delivery
100    e = Service("Eat-in")
101    t = Service("Take-out")
102    u = Service("Delivery")
103
104
105    # Build an example full theory for your setting and return it.
106    #
107    # There should be at least 10 variables, and a sufficiently large formula to describe it (>50 operators).
108    # This restriction is fairly minimal, and if there is any concern, reach out to the teaching staff to clarify
109    # what the expectations are.
110    def solution():
111        # E.add_constraint(v | g | d | h | ~h | ~v | ~g | ~d)
112        # E.add_constraint(~(~e & ~t & ~u))
113        E.add_constraint((e & (i | o)) | (~e & ~i & ~o)) # eat-in means there must be either indoor or outdoor
114        E.add_constraint(f >> t) # fast-food restaurants have take-out
115
116        # TODO: code for loop to add constraints
117
118        # E.add_constraint(r3 >> (r3 | r4 | r5))
119        # E.add_constraint(r3 & p2)
120
121        return E
122

```

(0.1)

We also began thinking about how we could eliminate restaurants from our solution based on certain constraints. Our idea was that we could automatically create a proposition corresponding to each restaurant in a Google Sheet. We have yet to have our program actually take restaurants from this sheet.

```

22     for i in range(10):
23         exec("m" + str(i) + " = BasicProposition('k" + str(i) + "')"
24         print(eval("m" + str(i)))
25
26     print(m1)

```

To start with, we have been experimenting with the code above in order to get a better sense of how this could be implemented and arbitrarily scaled as necessary. The above code will successfully create 10 BasicPropositions (labelled k0-k9), that currently exist beyond the scope of the loop in which they were created, and which will require more experimentation to ascertain stability.

We also created our own `displaySolution` function so that we could more clearly see which propositions are true. This function returns a list of True propositions for easier readability.

```
126 def displaySolution():
127     if not T.satisfiable():
128         return
129     result = []
130     lis = T.solve()
131     for key in lis.keys():
132         if lis.get(key):
133             result.append(key.__repr__())
134     result.sort()
135     return result
```

Because we chose a project that doesn't come with a strictly-defined ruleset, we spent a significant amount of time discussing what we wanted to be able to accomplish, and so when we get those things fully implemented, we will be able to explore the model more deeply at that time.

Jape Proofs

Proof 1: If the patio is not open during a restaurant's opening hours, then the weather condition is either raining or snowing. The weather can be either sunny or either raining or snowing.

P: The patio is open.

T: The restaurant is open.

Q: It is snowing.

R: It is raining.

S: It is sunny.

1: $(\neg P \wedge T), S \rightarrow P, \neg S \rightarrow (R \vee Q)$ premises

2: $\neg P$ \wedge elim 1.1

3: $\neg P \rightarrow \neg S$ Theorem $P \rightarrow Q \vdash \neg Q \rightarrow \neg P$ 1.2

4: $\neg S$ \rightarrow elim 3,2

5: $R \vee Q$ \rightarrow elim 1.3,4

Proof 2: There does not exist a weather condition that involves snow, but no precipitation. Patios are always closed if it snows. Whenever the patio isn't open it's precipitating.

S(x): The object x, involves snow.

P(y): The object y, allows for patios to open.

R(x): The object x, involves precipitation.

1:	$\neg \exists x.(S(x) \wedge \neg R(x), \forall y.(P(y) \vee S(y))$	premises
2:	actual i	assumption
3:	$P(i) \vee S(i)$	\forall elim 1.2,2
4:	$\neg P(i)$	assumption
5:	$\neg R(i)$	assumption
6:	$P(i)$	assumption
7:	\perp	\neg elim 6,4
8:	$S(i)$	contra (constructive) 7
9:	$S(i)$	assumption
10:	$S(i)$	\vee elim 3,6-8,9-9
11:	$S(i) \wedge \neg R(i)$	\wedge intro 10,5
12:	$\exists x.(S(x) \wedge \neg R(x))$	\exists intro 11,2
13:	\perp	\neg elim 12,1.1
14:	$R(i)$	contra (classical) 5-13
15:	$\neg P(i) \rightarrow R(i)$	\rightarrow intro 4-14
16:	$\forall y.(\neg P(y) \rightarrow R(y))$	\forall intro 2-15

Proof 3: There exists a restaurant with low price and does not satisfy dietary restrictions, and a restaurant with low rating and satisfy dietary restrictions. Concludes restaurant with low rating and satisfy dietary restrictions.

$R1(x)$: The object x is a unique restaurant

$R2(x)$: The object x is a unique restaurant

P: Has a low price

S: Has a low rating

D: Dietary restrictions are satisfied

1: $\exists x.((R1(x) \wedge P \wedge \neg D) \wedge (R2(x) \wedge S \wedge D$	premise
2: $\text{actual } i, (R1(i) \wedge P[xi] \wedge \neg D[xi]) \wedge (R2(i) \wedge S[xi] \wedge D[xi])$	assumptions
3: $R2(i) \wedge S[xi] \wedge D[xi]$	\wedge elim 2.2
4: $D[xi]$	\wedge elim 3
5: $R1(i) \wedge P[xi] \wedge \neg D[xi]$	\wedge elim 2.2
6: $\neg D[xi]$	\wedge elim 5
7: \perp	\neg elim 4,6
8: $R(x) \wedge P \wedge \neg C$	contra (constructive) 7
9: $R(x) \wedge P \wedge \neg C$	\exists elim 1,2-8

First-Order Extension

Predicates

Person(restrictions, preferences): List restrictions is the list of dietary restrictions a person has, list preferences is the list of preferences a person has

Restaurant(x): object x is the name of the restaurant

PartySize(n): n is the number of people in the party, N can contain values 2-8

Time(t): t is the hour that the party needs to eat, can contain values 0 - 23

Weekday(d): d is the weekday that the party needs to eat, can contain values 0-6

Restrictions (dietary, price): List dietary is the list of food-related restrictions, object price is a price value

Dietary(x, y): object x has y, where y is a dietary restriction

Price(x): object x is a price range, can hold values (\$, \$\$, \$\$\$, \$\$\$\$)

Rating(x): object x is the number of rating stars on Google, can be 1, 2, 3, 4, or 5

Seating(x): object x is true or false

Location(p, r, d): Person p is distance d from restaurant r.

Open(x): object x is true or false

OpenTime(t): object t is a time value, measured using hours from midnight.

OpenDay(d): object d contains value of a weekday, can range from 0-6

Time(t): object t is a time value, measured using hours from midnight.

Day(d): object d contains value of a weekday, can range from 0-6

Weather(x): object x defined a weather condition

Constraints

Earlier, we were able to represent some of our constraints using first-order logic. However, now that we are able to represent our proportions in first-order logic, it allows us to better represent our constraints.

- Each restaurant can only have one rating at a time
 $\forall x. (\text{Restaurant}(x) \implies \forall y1. \forall y2. \neg(\text{Rating}(y1) \wedge \text{Rating}(y2)))$
- Each restaurant can only have one price label on Google at a time
 $\forall x. (\text{Restaurant}(x) \implies \forall y1. \forall y2. \neg(\text{Price}(y1) \wedge \text{Price}(y2)))$
- Each restaurant has exactly one unique address a
 $\exists a L_{r,a} \wedge \forall b (L_{r,b} \rightarrow a = b)$
- Each restaurant is on Princess St.
 $\forall x. \exists n. (\text{Restaurant}(x) \implies (\text{Location}(n)))$
- Only one weather condition can occur at once.
 $\forall x1. \forall x2. \neg(\text{Weather}(x1) \wedge \text{Weather}(x2))$
- Restaurants must be open on the day and hour the party eats.
 $\forall x. y. t. d. (\text{Restaurant}(x) \wedge \text{Open}(y) \wedge \text{Day}(d) \wedge \text{Time}(t) \implies (\text{Open-Time}(t) \wedge \text{OpenDay}(d)))$
- All restaurants must not violate any dietary constraints. For example, if a person in the group is vegetarian, all restaurants the party is considering must be vegetarian.
 $\forall x. \exists y \forall p. (Person(p) \wedge Restaurant(x)) \implies ((Dietary(p, y) \implies (Dietary(x, y)))$