# Project Summary

Due to all the restrictions put in place during the COVID-19 pandemic, it has become increasingly more difficult to find restaurants. Our model will determine which restaurant accommodates the most preferences of the user, such as price range, dietary restrictions, and delivery or take out options. Our model will take into account a wide range of personal preferences of the user, and find the best fit restaurant for them to eat at.

Propositions

**low**: Represents the lowest price range, true when the restaurant can provide a meal within the range

**med**: Represents the middle price range

**high**: Represents the highest price range

**restriction\_gluten**: True if the restaurant can provide gluten free meals

**restriction\_vegan**: True if the restaurant can provide vegan meals

**restriction\_lactose:** True when the restaurant can provide lactose-free meals

**restriction\_vegetarian:** True when the restaurant can provide lactose-free meals

**dine\_in**: True when the restaurant allows the user to dine-in

**take\_out**: True when the restaurant has a take-out option

**delivery**: True when the restaurant has a delivery option

**under\_10**: Under a 10 minute walk from Queen’s campus

**10\_to\_20**: Between a 10 to 20 minute walk from Queen’s campus

**over\_20**: Over a 20 minute walk from Queen’s campus

# Constraints

**Price**

* + Lowest price point:
    - (low **∧** ~med **∧** ~high)
  + Medium price point:
    - (med **∧** ~high ~low)
  + High price point:
    - (high **∧** ~low **∧** ~med)

**Dietary restrictions**

Single Dietary restrictions:

* Gluten ^ ~vegan & ~Vegetarian ^ ~Lactose
  + Gluten free will evaluate to true if the restaurant and the user both have gluten free options.
* ~Gluten ^ vegan & ~Vegetarian ^ ~Lactose
  + vegan will evaluate to true if the restaurant and the user both have vegan options.
* ~Gluten ^ ~vegan & Vegetarian ^ ~Lactose
  + Vegetarian will evaluate to true if the restaurant and the user both have Vegetarian options.
* ~Gluten ^ ~vegan & ~Vegetarian ^ Lactose
  + Lactose free will evaluate to true if the restaurant and the user both haveLactose free options.

Two Dietary restrictions:

* Gluten ^ vegan & ~Vegetarian ^ ~Lactose
  + When the user and the restaurant both have gluten free and vegan options
* Gluten ^ ~vegan & Vegetarian ^ ~Lactose
  + When the user and the restaurant both have gluten free and vegetarian options
* Gluten ^ ~vegan & ~Vegetarian ^ Lactose
  + When the user and the restaurant both have gluten free and lactose free options
* ~Gluten ^ vegan & ~Vegetarian ^ Lactose
  + When the user and the restaurant both have vegetarian and vegan options
* ~Gluten ^ vegan & Vegetarian ^ ~Lactose
  + When the user and the restaurant both have vegetarian and lactose free options
* ~Gluten ^ ~vegan & Vegetarian ^ Lactose

Three Dietary restrictions:

* Gluten ^ vegan & Vegetarian ^ ~Lactose
  + When the user and the restaurant both have gluten free, vegan options, and vegetarian options
* Gluten ^ vegan & ~Vegetarian ^ Lactose
  + When the user and the restaurant both have gluten free, vegan and lactose free options
* ~Gluten ^ vegan & Vegetarian ^ Lactose
  + When the user and the restaurant both have vegan,vegetarian and lactose free options

All dietary restrictions

* Gluten ^ Vegan ^ Vegetarian ^ Lactose
  + When the user and the restaurant both have all dietary restrictions

**Dinning**

One Dining Option

* (dine\_in ^ ~take\_out ^ ~delivery)
  + When the user and the restaurant have a dine in option
* (~dine\_in ^ take\_out ^ ~delivery)
  + When the user and the restaurant have a take out option
* (~dine\_in ^ ~take\_out ^ delivery)
  + When the user and the restaurant have a delivery option

Two Dinning Options

* (dine\_in ^ take\_out ^ ~delivery)
  + When the user and the restaurant both have take out and dine in options
* (dine\_in ^ ~take\_out ^ delivery)
  + When the user and the restaurant both have dine in and delivery options
* (~dine\_in ^ take\_out ^ delivery)
  + When the user and the restaurant both have take out and delivery options

All Dinning Options

* (dine\_in ^ take\_out ^ delivery)

**Distance**

* + Under 10:
    - (under\_10 **∧** ~10\_to\_20 **∧ ~**over\_20)
  + 10 to 20:
    - (~under\_10 **∧** 10\_to\_20 **∧ ~**over\_20)
  + Over 20:
    - (~under\_10 **∧** ~10\_to\_20 **∧** over\_20)

# Model Exploration

We have explored our model in the following ways throughout the course of our project:

**How we modified / improved our propositions and constraints:**

* Our model is structured to match the user with the best result based on some search criteria. The user has the option to select a price point, Add dietary restrictions, Add dinning options and add distance options. The appropriate constraints only get added to the encoding when the user has selected that specific option. This way it only solves what it needs to and can give a cleaner theory. The restaurants are then looped through and each restaurant is tested for the theory to determine if its a good fit or not. If it is it is sorted to the top of the list while the restaurants that are not good fits are sorted to the bottom. The list is then displayed and the user can make another search.
* Our original idea was to make a score system to count for each valid restriction and assign the top restaurant with a score so they can be sorted from best to worst fit. The issue we ran into was that our propositions were not set up correctly to use the library and encoding so when we got to that stage it took hours of debugging to figure out where the errors were. Once they constraints were fixed and all functioned properly we completed the example theory method and the main with a prompt to the user for their preferences.
* Initially, three of our propositions for distance were under\_10, under\_20, and over\_20. One of our constraints was that a restaurant must be under a 10 minute walk, under 20 minutes, or over 20 minutes. In order for us to categorize and offer the top pick between the restaurants, the restaurant must be one, and only one, of the propositions. The constraint format we chose to use was:

Under 10:

* + (under\_10 **∧** ~10\_to\_20 **∧ ~**over\_20)

10 to 20:

* + (~under\_10 **∧** 10\_to\_20 **∧ ~**over\_20)

Over 20:

* + (~under\_10 **∧** ~10\_to\_20 **∧** over\_20)

After analyzing these constraints, we realized that the propositions should be under\_10, 10\_to\_20, and over\_20 instead. The constrai

* When starting our project, we had made a proposition for opening time and closing time of the restaurant. However, after careful consideration we decided to take these propositions out since it did not make sense to model whether the time a user wanted to go to a restaurant was within the opening and closing time of the restaurant with logical propositions. It would not have made sense, or would have over complicated the problem, to assign a truth value to a proposition modelling this.
* We also started out with a proposition for the type of cuisine a restaurant offers. This proposition was taken out as well since in order to model it as a logical proposition, we would need a true or false value for each type of cuisine. After exploring the scope of our project, we realized that it would be unnecessary to have each type of a cuisine as a proposition, since other requirements would be of greater importance to the user.
* Our initial plan to represent the price range of a restaurant was to use the proposition mi. The proposition was true when m was less than or equal to i. We realized this was not the best way to represent this criteria. Instead, we replaced it with price\_$, price\_$$, and price\_$$$ to represent three ranges of prices. Each restaurant would be True for one of the propositions and False for the others for clarity.

**How we modified / improved our code:**

* In our code, we started by storing our restaurants in a 2D list. After hearing feedback from peer assessments, we decided that the restaurants could be better modelled if they were stored as a list of restaurant objects. This is because it is easier to store information on each restaurant using an object. This method is more time efficient since there would be less iterating through lists and it is easier to see how everything is stored. As well, we added a csv reader function as a way to read in the restaurant objects.
* Our first approach to our modelling was to add in all of the constraints to check if they satisfied the model. However with this approach, we didn’t have a way of saying that a restaurant satisfied the model or ranking/sorting the restaurants. We changed our approach to the problem and split the constraints up, making a separate function to evaluate each constraint. This allowed us to only encode the necessary constraints.

First-Order Extension

We could extend our model to a predicate logic setting by changing the propositions and constraints to the following:

* A(x): Represents the lowest price range, true when the restaurant can provide a meal within the range
* B(x): Represents the middle price range
* C(x): Represents the highest price range
* G(x): restaurant can provide gluten free meals
* V(x) True if the restaurant can provide vegan meals
* L(x): True when the restaurant can provide lactose-free meals
* P(x): True when the restaurant can provide lactose-free meals
* D(x): True when the restaurant allows the user to dine-in
* T(x): True when the restaurant has a take-out option
* E(x): True when the restaurant has a delivery option
* W(x): Under a 10 minute walk from Queen’s campus
* Y(x): Between a 10 to 20 minute walk from Queen’s campus
* Z(x): Over a 20 minute walk from Queen’s campus

**Extended Price**

* + ∃x(A(x) ∨ B(x) ∨ C(x)) - There exists a restaurant that is in one of the price ranges
  + ∀x(~(A(x) **∧** B(x)) - All restaurants cannot have more than one price range

**Extended Dietary restrictions**

* + ∃x(G(x) **∧** V(x) **∧** L(x) **∧** P(x)) - There exists a restaurant that can accommodate all four types of dietary restrictions
  + ∃x(G(x) ∨V(x) ∨L(x) ∨ P(x)) -There exists a restaurant that can accommodate at least one type of dietary restrictions
  + ∃x(G(x) **∧ (** V(x) ∨L(x) ∨ P(x))) -There exists a restaurant that can accommodate at least two types of dietary restrictions
  + ∃x(G(x) **∧** V(x) **∧ (**L(x) ∨ P(x)) -There exists a restaurant that can accommodate at least three types of dietary restrictions

**Extended Methods of consumption**

* + ∃x(D(x) **∧** T(x) **∧** E(x)) - There exists a restaurant that offers all three methods of consumption
  + ∃x(D(x) ∨ T(x) ∨ E(x)) -There exists a restaurant that offers at least one method of consumption
  + ∃x(D(x) **∧** (T(x) ∨ E(x)) - There exists a restaurant that offers dine-in and one other method of consumption

**Extended Distance**

* + ∃x(W(x) ∨ Y(x) ∨ Z(x)) - there exists a restaurant that is at least one distance
  + ∀x(~(W(x) **∧** Y(x) **∧** Z(x))) - all restaurants must not be three of the options