GRP_19: Event Planner

Alex Panfilov (21MP27)

Derek Peene (20DSP1)

Adam Qian (17AWQ)

George Salib (20GS36)

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

December 5, 2022

Abstract

Our goal is to create an event planner that schedules any number of events within a given time frame based on the availability of any number of people who need to attend. On top of availability, we also take into consideration the weather, preferences of attendees, and time of day (morning, afternoon, evening).
For any set of events, our model will be able to either:

1. Plan a best fit itinerary based on the preferences of attendees where not all events within the possible set have to happen OR
2. Plan a itinerary such that all events in the possible set will happen

Our event planner is intended to be used either by organizations planning a conference or groups of friends that want to decide on activities that work for everybody.

Propositions

Let i represent a person, and j represent a given activity, and k represent a given time. $i, j, k \in Z$.

- $T_{i,k}$ will represent timing, and is true if person i is available at time k.
  - E.g. $T_{2,1}$ is true if person 2 is available at time slot 1.
  - If we define time slots as an hour long each, for every day, we can generate a p x 24, where p is the number of participants we are working with.
- A_k represents time slot availability, and is true if all attendees are available at time k.
  - E.g. If all attendees are available at time slot 2 then A_2 is true.
  - This proposition will help aggregate all the attendees' availability. We can also easily catch unsolvable models where A_k is false for all k.
- $X_{j,k}$ will represent an activity, and is true if the activity j is feasible at a given time k.
  - E.g. $X_{1,1}$ is true if activity 1 (going to the movies) is a possible activity at a time 1.
- Q_j will represent if the activity j is indoors, in which case weather conditions do not matter.
  - E.g. If activity j is indoors, then Q_j is true.
- W_k will represent weather conditions, and is true if the weather conditions are clear for the given time.
  - E.g. W_2 could represent clear weather, and is true if it is clear at time 2.

- $S_{j,k}$ will represent a scheduled activity, and is true if an activity j has been scheduled at time k.
  - E.g. $S_{1,2}$ could represent that activity 1 has been put into the schedule at time 2.

Constraints

1. Availability holds when $A_k \leftrightarrow (T_{1k} \wedge T_{2k} \wedge ... \wedge T_{nk})$.

   This means that an activity can only hold iff every single person is available at a given time, thus everyone has time to attend the event.

   For example, if we say we choose noon time and check that everyone is available at that time, then availability holds, however, even if one person is busy at said time, the availability doesn't hold.

2. An activity holds when $X_{j,k} \leftrightarrow (A_k \wedge (W_k \vee Q_j))$.

   This means that an activity can take place iff everyone is available and either the activity is indoors or the weather is clear.

   For example, if at a given time everyone is free, and the activity can take place without weather concerns (i.e. indoors or weather is clear), then activity holds.

3. Only one event can take place at a given time k, thus, $S_{1,k} \leftrightarrow (X_{1,k} \wedge \neg X_{2,k} \wedge \neg X_{3,k} \wedge ... \wedge \neg X_{n,k})$.

   This means that a scheduled time slot is only valid iff there is at most one event scheduled at said time.

   For example, if at a given time everyone is free, and there is a movie and meeting that needs to be scheduled only one of them can be scheduled and not both.

4. We know that if an activity holds when the weather is not clear, then the activity must be indoors, thus: $(X_{j,k} \wedge \neg W_k) \longrightarrow Q_j$.

5. In addition, We know that if an activity holds and it is not indoors, then the weather must be clear, thus: $(X_{j,k} \wedge \neg Q_j) \longrightarrow W_k$.

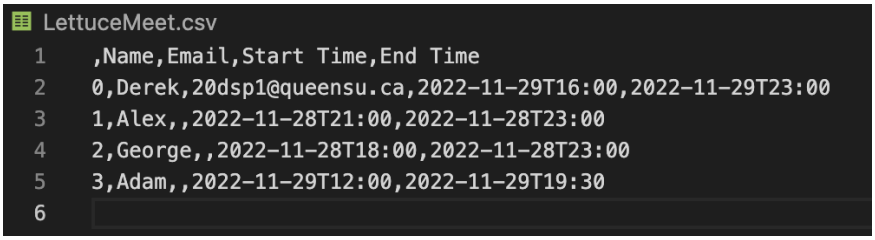
6. Assuming all activities need to be scheduled, the schedule is complete when $(S_{1,k} \wedge S_{2,k} \wedge ... \wedge S_{n,k})$.

   This means that the schedule holds (is complete) iff for all possible activities the scheduled activity proposition holds, meaning that all activities have been scheduled.

   For example, if there is not enough time for all of the activities to take place, then the schedule doesn't hold.

Model Exploration

To begin, in order to get the timetables of availability from the users, we decided that a good way to source this information is by web scraping LettuceMeet. LettuceMeet allows users to interactively select time slots in which they are available, thus, is a good way for each individual member to fill out their availability's. The csv file created by LettuceMeet is then used as data reference in order for the program to evaluate time slot availability's. Such data is stored as seen below:



We wanted the data to be structured as a dictionary associated to each person where the keys are the dates, and the values are the times during that day that a person is free. The implementation of which has been made in a separate python script to execute and is simply called in the *run.py* script.

In addition to sourcing the availability of each individual, we needed to source the weather at a given day and location, thus, to do this, we implemented a weather API where we get the location input from the user and modified the output to create a dictionary in which the keys are the dates and the values are a list containing the time, weather condition, and true or false depending if weather is clear for each time of the day. This allows for easy evaluation of weather conditions by the program. The implementation of the weather can be seen below:

```python
def get_weather():
    apiKey = "7310dd5bc23c33461ad6d71f286c1f05"
    rootUrl = "http://api.openweathermap.org/data/2.5/forecast?"

    cityName = input("Enter Your City: ")

    url = f"{rootUrl}appid={apiKey}&q={cityName}"


    allWeather = requests.get(url).json()

    filtWeather = {}

    list = allWeather['list']

    for dict in list:
        weatherList = dict['weather']
        weatherDict = weatherList[0]
        tempValue = weatherDict['description']
        tempKey = dict['dt_txt']
        filtWeather[tempKey] = tempValue

    dayDict = {}
    for k, v in filtWeather.items():
        if k[:10] in dayDict.keys():
            dayDict[k[:10]].append([k[-9:], v])
        else:
            dayDict[k[:10]] = [[k[-9:], v]]

    for v in dayDict.values():
        for i in v:
            if i[-1][-4:] == 'rain':
                goodWeather = 'F'
            elif i[-1][-4:] == 'snow':
                goodWeather = 'F'
            else:
                goodWeather = 'T'

            i.append(goodWeather)

    return dayDict
```

We decided to source the activities to be scheduled straight from the user using command line input, where the user enters the number of activities, duration per activity, and whether or not each activity is indoors. This information about every activity is then stored in a dictionary for each activity, all put in a list. Implementation can be seen below:

```python
def input_activities():
    print('Enter activities')
    activities = []
    finished_input = False
    while not finished_input:
        activity_name = input('Enter activity name: ').lower()
        activity_duration = int(input('Enter duration of activity in hours: '))
        activity_indoors = input('Is the activity indoors? Y/N: ').lower() == 'y'
        finished_input = input('Add another activity? Y/N: ').lower() == "n"

        activity = {
            'name': activity_name,
            'duration': activity_duration,
            'indoors': activity_indoors
        }

        activities.append(activity)
```

After this, we added all propositions and constraints we had to the code using the Bauhaus python library. This allows the program to use the information we had gathered from the user's availability's as well as the weather conditions and activities, and evaluate them using our constraints to schedule activities when our conditions are satisfied.

Currently, our model only supports fixed-duration activities and fixed blocks of time. For example, our days are split into chunks of 3 hours because we couldn't get any more granularity with our weather API. We tried changing our model so that we could split our day into 1-hour time slots and allow activities/events to span over multiple time slots. This involved changing how we modelled a few of our propositions and constraints:

- We need a way to store the length of an activity. The only way we came up with to do this using a simple true/false proposition is by creating a proposition for if the event is 1 hour, 2 hours, 3 hours, etc.

  Proposition $L1_j$ holds if activity j is of length 1.
  Proposition $L2_j$ holds if activity j is of length 2.
  Proposition Ln_j holds if activity j is of length n.

- With these new propositions, we have to add more constraints.

  An event can only have one length.
  $L1_j \rightarrow (\neg L2_j \wedge \neg L3_j \wedge \neg L4_j \wedge ... \wedge \neg Ln_j)$

  We need consecutive blocks of availability to schedule an event with a length greater than 1 hour.
  $L2_j \rightarrow (X_{j,k} \leftrightarrow ((A_k \wedge A_{k+1}) \wedge ((W_k \wedge W_{k+1}) \vee Q_j)))$
  $Ln_j \rightarrow (X_{j,k} \leftrightarrow ((A_k \wedge A_{k+1} \wedge ... \wedge A_{k+n}) \wedge ((W_k \wedge W_{k+1} \wedge ... \wedge W_{k+n}) \vee Q_j)))$

  Our condition for a complete schedule has now changed because now an activity can be scheduled on more than one time slot. For example, if

activity 1 is 2 hours, our condition for a complete schedule becomes: $(S_{1,k} \wedge S_{1,k+1} \wedge ... \wedge S_{n,k})$

Implementing these adjusted propositions and constraints in code proved much harder than we anticipated. It complicates the user input process quite a lot, and a lot of extra proposition have to be added for model length. The extra constraints were quite long and it took a lot of trial and error to properly translate them into code.

Once we were able to implement this new model in code, we ran into a problem with nonsensical solutions. We tested our model with a 3-hour availability window and a single 2-hour event, which should yield 2 solutions. We weren't able to achieve this, so it is clear that we either made an error in our modelling or implemented it incorrectly (or both). We decided to revert the model in our code back to our original fixed-time-slot version.

First-Order Extension

Propositions
The most central aspect of our event planner project is time, so our first-order extension should be modelled around time.

Let t be the integer representation of a time slot (i.e. 17-Sept-2022 6:00AM corresponds to t=1).

- Original proposition: $T_{i,k}$
  Extension: $T_i(t)$
  Such that $T_i(t)$ holds when person i is available at time t

- Original proposition: A_k
  Extension: $A(t)$
  Such that $A(t)$ holds when all attendees are available at time t

- Original proposition: $X_{j,k}$
  Extension: $X_j(t)$
  Such that $X_j(t)$ holds when activity j is feasible at time t

- Modelling whether or not an activity is indoors is a unique challenge since our predicate logic extension is based on time. We could introduce a variable x that represents an activity and use $Q(x)$ for indoors/outdoors, but we don't think it's worth it to complicate the model for just this proposition.

  Instead:
  Original proposition: Q_j
  Extension: $Q_j(t)$
  Such that $Q_j(t)$ holds when an activity j is indoors at time t. A single activity cannot be indoors and outdoors at different times, so $Q_j(t)$ will always be the same regardless of t. This is inefficient but saves us from having to create another variable for just this proposition.

- Original proposition: W_k
  Extension: $W(t)$
  Such that $W(t)$ holds when the weather is clear at time t.

- Original proposition: $S_{j,k}$
  Extension: $S_j(t)$
  Such that $S_j(t)$ holds when activity j has been scheduled for time t.

Examples of Constraints

- For availability to hold, we know that the timing of all attendees must line up for a given time.

$$\exists t.(A(t) \leftrightarrow (T_1(t) \wedge T_2(t) + ... + T_i(t))$$

- The constraint for when an activity holds does not differ much from the original.

$$X_j(t) \leftrightarrow (A(t) \wedge (W(t) \vee Q_j(t)))$$

- Only one event can be scheduled at any given T. Let j, g be integers that correspond to activities. Therefore for every possible t, if $S_j(t)$ and $S_g(t)$, then it would imply that $g = j$.

$$\forall t.(\exists j.S_j(t) \wedge \exists g.S_g(t)) \longrightarrow (j = g)$$

- The schedule is complete when all events have been scheduled.

$$\forall j.\exists t.S_j(t)$$

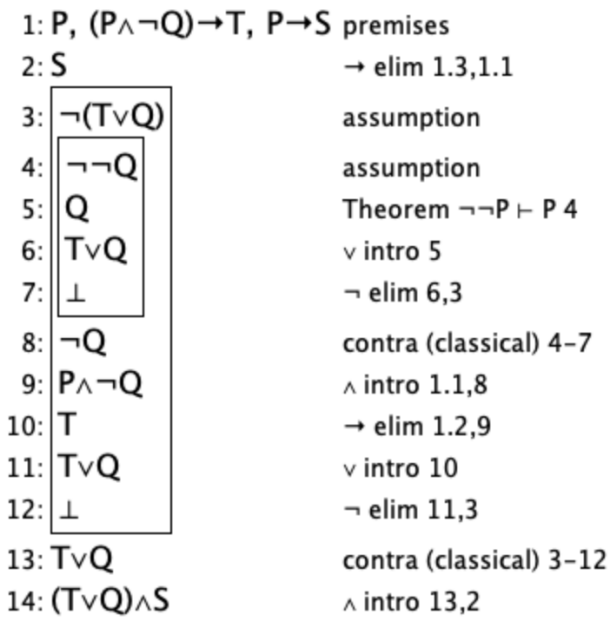
Jape Proofs

*Note that the propositions used refer to the SAME activity, time and person, thus, i,j,k are omitted.

1. We know from our constraints that if an activity holds then either the weather is clear, or the activity is indoors. Thus, if we assume that we have a schedule comprised of one activity, where schedule holds if that activity holds, with the premise that the activity holds, we can conclude that it must be true that the schedule holds and either the weather is clear or the activity is indoors.

   Thus, we get the premises: X, $(X \wedge \neg Q) \longrightarrow W$, $X \longrightarrow S$
   And the following conclusion: $(W \vee Q) \wedge S$.

   In order for this proof to be applicable in Jape, let X be P and let W be T (distinct from the letters used in our propositions), i.e. P is an activity, Q is if an activity is indoors, T is weather, and S is the schedule, such that the Jape proof can be seen below:

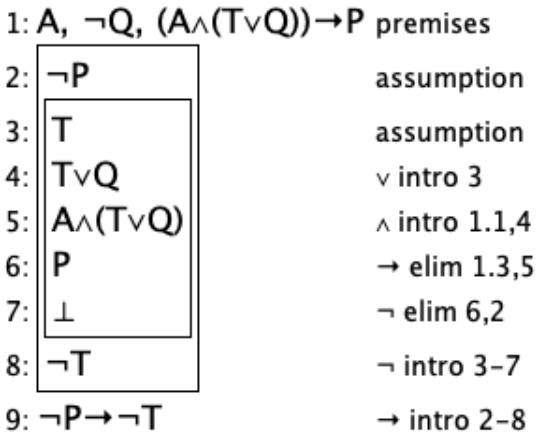


Therefore, through Jape, we have proved that if we assume that an activity holds, in addition to our constraint that if an activity holds and it is not indoors, then the weather must be clear, as well as a schedule that is bound to one activity, then it must be true that the schedule holds, and either the weather is clear or the activity is indoors.

2. Knowing that an activity holds when their is availability and clear weather or the activity is indoors, we can use Jape to prove that if we assume that there is availability, and that the activity is not indoors, then if the activity doesn't hold, that means that the weather is not clear and did not permit such activity to take place.

   Thus, let the premises be: A, $\neg Q$, $(A \wedge (W \vee Q)) \longrightarrow X$
   And the following conclusion: $\neg X \longrightarrow \neg T$.

   In order for this proof to be applicable in Jape, let X be P and let W be T (distinct from the letters used in our propositions), i.e. P is an activity, Q is if an activity is indoors, T is weather, and A is availability, such that the Jape proof can be seen below:

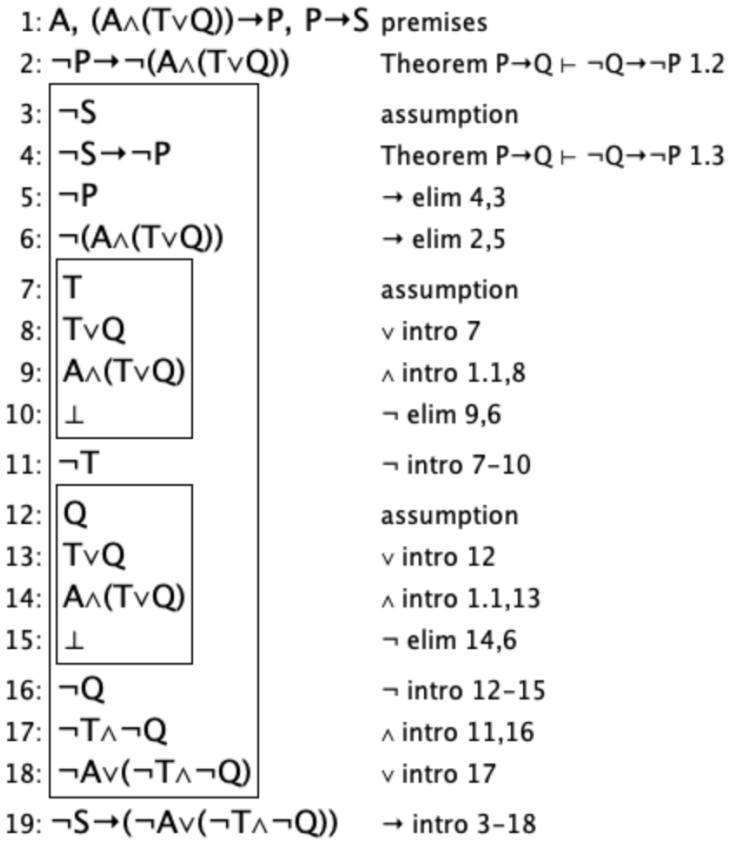


Therefore, through Jape, we proved that if we assume that availability holds, and that an activity is not indoors, in addition to the constraint of when an activity holds, if such activity doesn't hold, then this implies that the weather also doesn't hold.

3. An activity holds if there is availability and either the weather is clear, or the activity is indoors. Assuming availability holds, and we have a schedule comprised of one activity, where schedule holds if that activity holds, we can conclude that if the schedule doesn't hold, it must be true that availability doesn't hold, nor that the weather is clear and the activity is not indoors.

   Thus, we get the premises: A, $(A \wedge (W \vee Q)) \longrightarrow X$, $X \longrightarrow S$
   And the following conclusion: $\neg S \longrightarrow (\neg A \vee (\neg W \wedge \neg Q)))$.

   In order for this proof to be applicable in Jape, let X be P and let W be T (distinct from the letters used in our propositions), i.e. P is an activity, Q is if an activity is indoors, T is weather, A is availability, and S is the schedule, such that the Jape proof can be seen below:



Therefore, through Jape, we have proved that if we assume that availability holds, in addition to our activity constraint, then if the schedule doesn't hold, it must be true that either availability or weather and indoors doesn't hold.