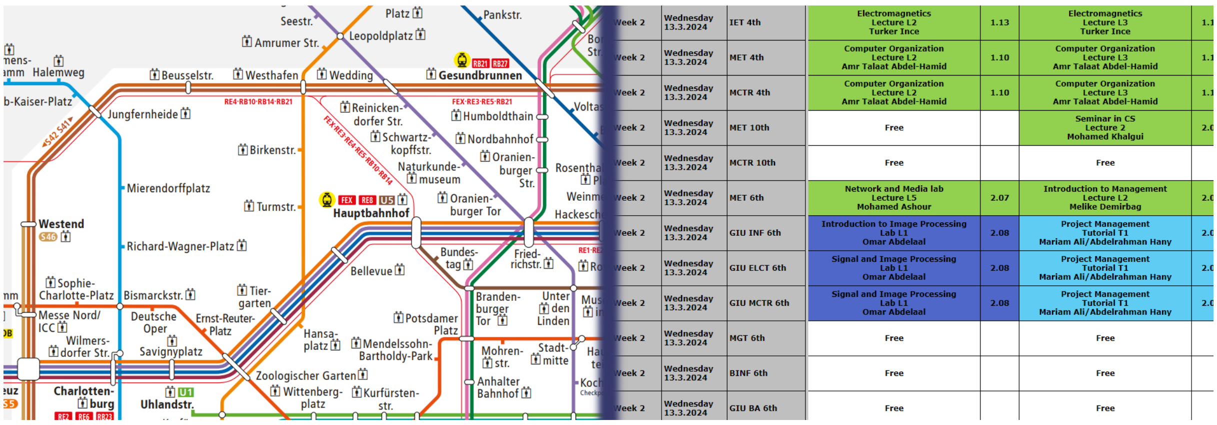
CSEN B303: Concepts of Programming Languages, Spring Term 2024
Prolog Project: Plan My Berlin Semester

Due: 5th May 2024



1 Overview

Since GIU Berlin students are punctual, and always try to arrive to their academic slots on time, the purpose of this project is to plan the transportation plan for each day a student needs to be on campus. You will also need to take into consideration the different majors, the different transportation lines and modes, as well as any transportation union strikes.

A trip plan details the journeys that a student belong to particular group will need to take from their home to the university campus. The trip takes into consideration that the student might be able to reach multiple stations from their home, in addition to being able to reach the campus from multiple stations. The plan consists of a journey for each day that a student needs to be on campus, as they need to arrive on time for the earliest scheduled slot for that day. The journey of each day details the duration of the entire journey from a station close to home, to a station close to the campus. Additionally, the journey for each day consists of which station to start a ride on a line on and the end station on that line, after which the student either reaches the campus, or transfers to another line to continue their journey. Each days journey is constrained by a maximum duration, a maximum number of rides and transfers, and which modes of transportation are available due to strikes.

2 Facts

2.1 Slots

The file `slots_kb.pl` posted on the CMS provides an example of a knowledge base of slots information. The data is encoded using the following predicates:

- `slot(Slot_Num, Start_Hour, Start_Minute)` indicates that a slot whose number is `Slot_Num` starts at `Start_Hour:Start_Minutes`. Please note that the time is in the 24 hour format. **Example:** `slot(4, 14, 15)`.
- `scheduled_slot(Week_Num, Week_Day, Slot_Num, Course_Name, Group)` indicates that a slot for `Course_Name` is scheduled for `Group` during a slot whose number is `Slot_Num` on `Week_day` of week `Week_Num`. **Example:** `scheduled_slot(6, mon, 2, concepts_of_programming_lang, met_4)`.

2.2 Transportation

The file `transport_kb.pl` posted on the CMS provides an example of a knowledge base of transportation information. The data is encoded using the following predicates:

- `line(Line_Name, Line_Type)` indicates that `Line_Name` is of type `Line_Type`. **Example:** `line(u6 ubahn)`, `line(s5, sbahn)`.
- `unidirectional(Line)` indicates that `Line` is unidirectional (i.e. connections between stations are one way). **Example:** `unidirectional(s42)`.
- `campus_reachable(Station)` indicates that campus is reachable from `Station`.
  Example: `campus_reachable(borsigwerke)`.
- `strike(Line_Type, Week_Num, Week_Day)` indicates that there is a strike on `Week_Day` of a week `Week_Num` for all lines of `Line_Type`. **Example:** `strike(ubahn, 3, wed)`.
- `connection(Station_A, Station_B, Duration, Line)` indicates that `Station_A` is connected to `Station_B` on `Line`, and the time to go between them is `Duration`.
  Example: `connection(hermannplatz, rathaus_neukoelln, 1, u7)`.

3 Data Structures

The below data structures are utilized in the predicates described below. You are allowed to create additional data structures.

- `day_timing(Week, Day)` encodes a timing on `Day` of week `Week`.
- `journey(Week_Num, Week_Day, Start_Hour, Start_Minute, Total_Duration, Routes)` encodes a journey on `Week_Day` of week `Week_Num` which starts at `Start_Hour`:`Start_Minute`, takes a total duration of `Total_Duration` minutes, and consist of taking `Routes`.
- `route(Line, Start_Station, End_Station, Duration)` encodes a route on `Line` from `Start_Startion` to `End_Station` which takes takes `Duration` minutes. A route is a continuous trip on a transportation line, where intermediate connections are not shown.

4 Predicates

Your implementation **must** contain the below predicates as described. Read the description of all of them before you start your implementation. It is recommended to implement the predicates in each section in the order they appear in.

You are allowed -and will need, based on your approach- to implement additional helper predicates. You are also allowed and encouraged to use any built-in predicates or libraries, with the **exception** of `assert` and `retract`.

4.1 Slots

4.1.1 group_days/2

`group_days(Group, Day_Timings)` holds if `Day_Timings` is the list of `day_timing(Week, Day)` (see section 3) on which students belonging to `Group` have at least one slot on.

Example Query

```
?- group_days(met_4, Day_Timings).
Day_Timings = [day_timing(1, mon), day_timing(1, tue), day_timing(1, wed), day_timing
    (1, sat), day_timing(2, mon), day_timing(2, tue), day_timing(2, wed), day_timing
    (2, thu), day_timing(..., ...)|...].
```

4.1.2 day_slots/4

`day_slots(Group, Week, Day, Slots)` holds if `Slots` is the list of all slots for `Group` on `Day` of week `Week`.

Example Query

```
?- day_slots(met_10, 2, wed, Slots).
Slots = [2, 3, 5].
```

4.1.3 earliest_slot/4

`earliest_slot(Group, Week, Day, Slot)` holds if `Slot` is the earliest slot for `Group` on `Day` of week `Week`.

Example Query

```
?- earliest_slot(met_10, 2, wed, Slot).
Slot = 2.
```

4.2 Transportation

4.2.1 proper_connection/4

`proper_connection(Station_A, Station_B, Duration, Line)` holds if `Station_A` and `Station_B` are connected on `Line`, and the time to go between them is `Duration`. All while taking into consideration the bidirectionality or lack thereof of `Line`.

Example Queries

```
?- proper_connection(albertinenstr, antonplatz, 1, m4).
true.
?- proper_connection(antonplatz, albertinenstr, 1, m4).
true.
?- proper_connection(gesundbrunnen, wedding, 2, s42).
true.
?- proper_connection(wedding, gesundbrunnen, 2, s42).
false.
```

4.2.2 append_connection/6

`append_connection(Conn_Source, Conn_Destination, Conn_Duration, Conn_Line, Routes_So_Far , Routes)` holds if `Routes` (see section 3) is the result of appending a connection from `Conn_Source` to `Conn_Destination` on `Conn_Line` which takes `Conn_Duration` minutes, to `Routes_So_Far`.

Example Queries

```
?- append_connection(friedrichstr, oranienburger_tor, 1, u6, [route(u5, schillingstr,
    alexanderplatz, 2), route(s5, alexanderplatz, friedrichstr, 4)], Routes).
Routes = [route(u5, schillingstr, alexanderplatz, 2), route(s5, alexanderplatz,
    friedrichstr, 4), route(u6, friedrichstr, oranienburger_tor, 1)].
?- append_connection(friedrichstr, hauptbahnhof, 2, s5, [route(u5, schillingstr,
    alexanderplatz, 2), route(s5, alexanderplatz, friedrichstr, 4)], Routes).
Routes = [route(u5, schillingstr, alexanderplatz, 2), route(s5, alexanderplatz,
    hauptbahnhof, 6)] .
```

4.2.3 connected/8

`connected(Source, Destination, Week, Day, Max_Duration, Max_Routes, Duration, Routes)` holds if it is possible by following a sequence of `Routes` (see section 3) to reach `Destination` from `Source` on `Day` of `Week`, where the combined `Duration` of `Routes` does not exceed `Max_Duration`, and the number of `Routes` does not exceed `Max_Routes`.

Example Queries

```
?- connected(westhafen, leopoldplatz, 1, mon, 10, 3, Duration ,Routes).
Duration = 3,
Routes = [route(u9, westhafen, leopoldplatz, 3)].
?- connected(westhafen, leopoldplatz, 1, tue, 10, 3, Duration ,Routes).
Duration = 3,
Routes = [route(s41, westhafen, wedding, 2), route(u6, wedding, leopoldplatz, 1)] ;
Duration = 9,
Routes = [route(s41, westhafen, gesundbrunnen, 4), route(u8, gesundbrunnen, osloer_str
    , 3), route(u9, osloer_str, leopoldplatz, 2)] ;
Duration = 3,
Routes = [route(u9, westhafen, leopoldplatz, 3)].
?- connected(westhafen, leopoldplatz, 1, wed, 10, 3, Duration ,Routes).
false.
```

4.2.4 connected/10

`connected(Source, Destination, Week, Day, Max_Duration, Max_Routes, Duration, Prev_Stations , Routes_So_Far, Routes)` holds same as **connected/8** in addition to `Prev_Stations` being a list of all the previously visited stations, which should not be visited again, and `Routes_So_Far`, containing the traversed routes before reaching `Source`, where the route from `Source` to `Destination` is to appended to.

Example Query

```
 ?- connected(amrumer_str, leopoldplatz, 1, mon, 2, 1, 2, [westhafen], [route(u9,
    westhafen, amrumer_str, 1)], [route(u9, westhafen, leopoldplatz, 3)]).
true.
```

4.3 Time Conversion

4.3.1 mins_to_twentyfour_hr/3

`mins_to_twentyfour_hr(Minutes, TwentyFour_Hours, TwentyFour_Mins)` holds if `TwentyFour_Hours`:`TwentyFour_Mins` is the twenty-four hour representation of `Minutes` since midnight.

Example Query

```
?- mins_to_twentyfour_hr(795, TwentyFour_Hours, TwentyFour_Mins).
TwentyFour_Hours = 13,
TwentyFour_Mins = 15.
```

4.3.2 twentyfour_hr_to_mins/3

`twentyfour_hr_to_mins(TwentyFour_Hours, TwentyFour_Mins, Minutes)` holds if `Minutes` since midnight is equivalent to the twenty-four hour formatted `TwentyFour_Hours`:`TwentyFour_Mins`.

Example Query

```
?- twentyfour_hr_to_mins(14, 30, Minutes).
Minutes = 870.
```

4.3.3 slot_to_mins/2

`slot_to_mins(Slot_Num, Minutes)` holds if `Minutes` since midnight is equivalent to the start time of a slot whose number is `Slot_Num`.

Example Query

```
?- slot_to_mins(3, Minutes).
Minutes = 735.
```

4.4 Main Predicate

It is recommended to implement this predicate after all the above predicates.

4.4.1 travel_plan/5

`travel_plan(Home_Stations, Group, Max_Duration, Max_Routes, Journeys)` holds if `Journeys` (see section 3) is a valid plan as described in section 1.

Example Queries

```
?- travel_plan([friedrichstr, universitaetsstr], met_10, 24, 2, Journeys).
Journeys = [journey(1, wed, 10, 6, 24, [route(s1, friedrichstr, schoenholz, 16), route
    (s25, schoenholz, tegel, 8)]), journey(1, thu, 11, 56, 19, [route(u6, friedrichstr
    , borsigwerke, 19)]), journey(1, sat, 10, 8, 22, [route(12, friedrichstr,
    naturkundemuseum, 6), route(u6, naturkundemuseum, borsigwerke, 16)]), journey(...,
     ..., ..., ..., ..., ...)|...].
?- travel_plan([ostkreuz, wuehlischstr_gaertnerstr, samariterstr], met_4, 60, 3,
    Journeys).
Journeys = [ journey(1, mon, 11, 43, 32, [route(u5, samariterstr, unter_den_linden,
    12), route(u6, unter_den_linden, borsigwerke, 20)], journey(1, tue, 9, 32, 58, [
    route(m13, wuehlischstr_gaertnerstr, seestr, 48), route(u6, seestr, borsigwerke,
    10)]), journey(1, wed, 10, 0, 30, [route(s42, ostkreuz, gesundbrunnen, 14), route(
    s25, gesundbrunnen, tegel, 16)]), journey(2, mon, 7, 53, 37, [route(m13,
    wuehlischstr_gaertnerstr, warschauer_str, 4), route(u1, warschauer_str,
    hallesches_tor, 9), route(u6, hallesches_tor, borsigwerke, 24)]), ..., journey(12,
     wed, 9, 58, 32, [route(s8, ostkreuz, bornholmer_str, 18), route(s25,
    bornholmer_str, tegel, 14)].
```

5 Notes

5.1 Assumptions

The following assumptions are made for simplicity:

- All connections are bidirectional (work both ways) with the same duration, with the exception of connections on a line which is indicated to be unidirectional.
- Upon reaching a station, it is always possible to take a connection. In other words, there is no need to wait for a transportation vehicle to arrive.
- Transferring from a line to a different line is instant.
- The student is attempting to arrive on campus exactly at the start time of the first slot of the day.
- The student only needs to arrive on campus for academic slots, without considering exams, quizzes or other academic activities.
- Calculated times do not need to cross the same day boundary.

5.2 Implementation Details

- If for at least one day it is not possible to arrive on campus, `travel_plan` is `false`.
- Days where the group does not have any slots are not included in `Journeys`.
- While building the route, it is crucial to
  - check if the next station to be visited was already visited **before** the recursive call. This is to avoid cycles.
  - enforce the `Max_Duration` and `Max_Routes` constraints **before** the recursive call. This is to avoid needlessly deep searches.
  - When transferring from one line to another at a station, this station is considered **both** the end of the earlier route, and the start of the latter route.

5.3 Debugging Tips

- While debugging, it is useful to only use a subset of the provided knowledge bases.
- Avoid having needlessly high `Max_Duration` and `Max_Routes` while testing, as they result in deeper searches, and thus, longer query execution times.
- Initially avoid testing routes containing connections to many lines for simplicity. However, you should test more complex routes once the basics of your implementation are there.

6 Guidelines

- This project should be done **in teams of 4-5**. Please note that in case of unassigned members, some teams of 4 *might* be assigned an additional member.
- Please use the following link to register your team by ***Wednesday 10th of April at 11:59 PM*** `https://forms.gle/kYjH7jPSWKbm7vLG6`. In case you missed the deadline, you will be assigned randomly to a team. There will be **no changes** allowed after the deadline.
- You can consult the manual and search online. However, all work done in this project must be done by the team members and the team members only. **All team members should work on this project equally** and no work should be done by anyone outside the team. Do not discuss approaches for completing the project with colleagues. This also means that copying code from online resources or ChatGPT is not allowed. Team evaluations *might* be conducted at the end in order to verify that. All files **will** be checked for plagiarism.
- Each team should submit a single `.pl` file, containing the team's full project implementation. The submitted file **must** abide by the following rules:
  - The file should be named in the following format "**Team_TeamNumber.pl**", for example: "**Team_7.pl**".
  - Your file should **not** include any additional facts, or include any additional files except `:- ['transport_kb', 'slots_kb'].`
  - You should include a clear documentation per implemented predicate. You should write each predicate's documentation above the predicate's implementation.
- It is the team's full responsibility to successfully submit a valid `.pl` file before the project's deadline using the following link: `https://forms.gle/5G644aM4QcnWKxbB8`