Course Schedule Building

Adebayo Braimah

Problem

- Scheduling can be a non-trivial problem
  - Nurse scheduling problem (NSP)[1]
    - Complex constraints: availability, nurse-to-patient ratios, skill set requirements, work/rest hours
    - Balancing needs: healthcare facility for sufficient staffing, personal/professional preferences of staff
  - University course scheduling[2]
    - Complex constraints: student course requirements (including pre- and corequisites), and overall course load
    - Balancing needs: maximize academic performance, graduate in a sufficient amount of time

[1] C. Dodaro and M. Maratea, “Nurse Scheduling via Answer Set Programming,” in Logic Programming and Nonmonotonic Reasoning, vol. 10377, M. Balduccini and T. Janhunen, Eds., in Lecture Notes in Computer Science, vol. 10377. , Cham: Springer International Publishing, 2017, pp. 301–307. doi: 10.1007/978-3-319-61660-5_27. ACCESSED: May-07-2024.
[2] Ana Y.F. de Lima, Briza M. D. de Sousa, Daniel P. Cardeal, Jessica Y. N. Sato, Lorenzo B. Salvador, and Bruna Bazaluk, “LUNCH: an Answer Set Programming System for Course Scheduling.” ACCESSED: May-07-2024.

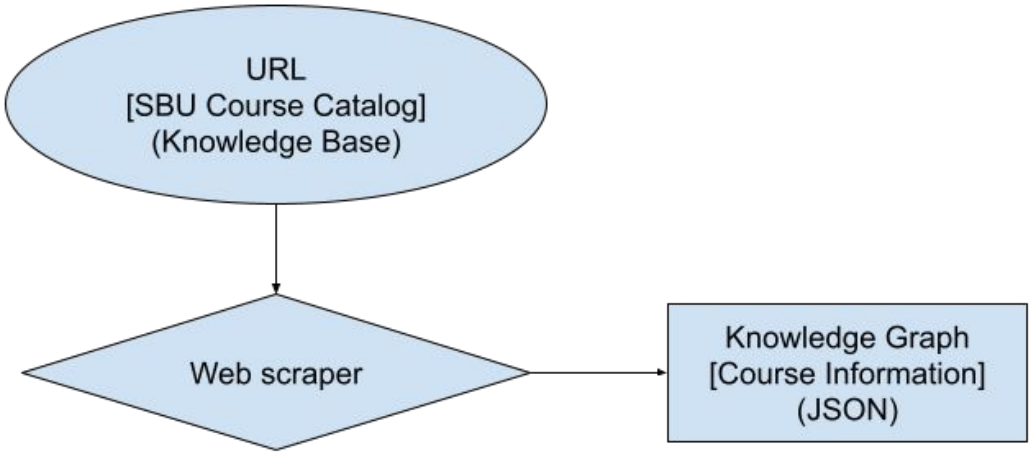
Problem

- State of the Art (SOTA)
  - LUNCH[2]
    - Automated
    - Creates a schedule for one semester
    - Requires significant user input
  - SOLAR[3]
    - Manual/not automated
    - Creates a schedule for one semester
    - Does not require too much user input

[2] Ana Y.F. de Lima, Briza M. D. de Sousa, Daniel P. Cardeal, Jessica Y. N. Sato, Lorenzo B. Salvador, and Bruna Bazaluk, “LUNCH: an Answer Set Programming System for Course Scheduling.” ACCESSED: May-07-2024.
[3] Division of Information Technology – Stony Brook University. Schedule Builder. https://it.stonybrook.edu/services/schedule-builder, 2016. ACCESSED: Mar-11-2024.

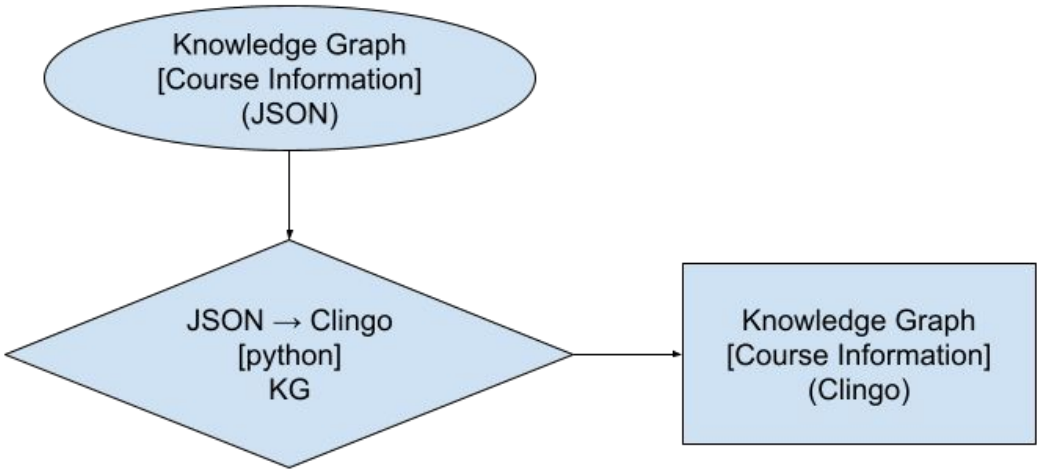
Design

- 3 major components:
  - **Knowledge graph construction from SBU course catalog.**
  - Converting JSON knowledge graph to Clingo atoms and predicates.
  - Query the knowledge base of course requirements.



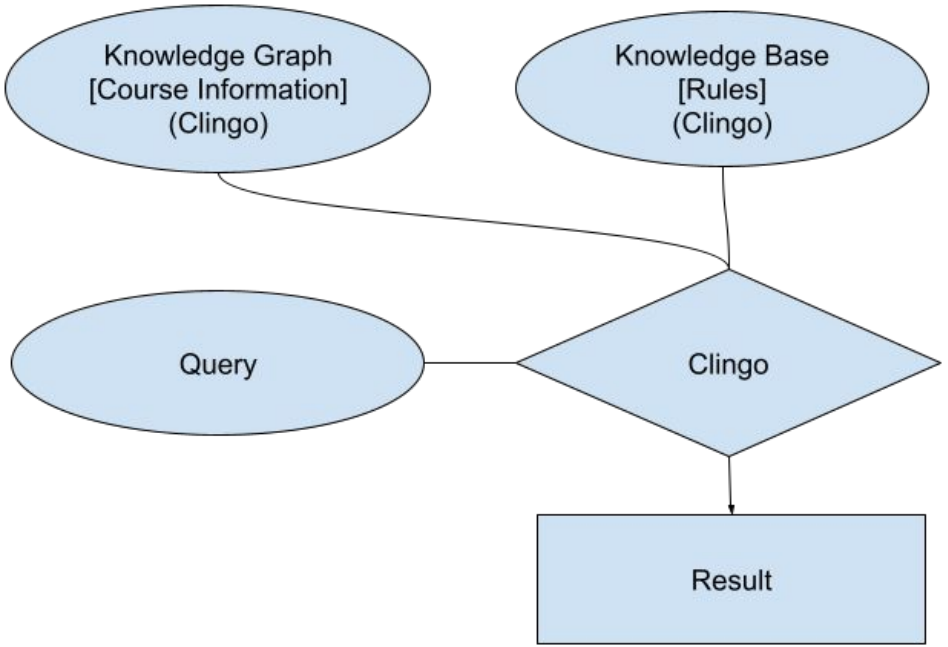
Design

- 3 major components:
  - Knowledge graph construction from SBU course catalog.
  - **Converting JSON knowledge graph to Clingo atoms and predicates.**
  - Query the knowledge base of course requirements.



Design

- 3 major components:
  - Knowledge graph construction from SBU course catalog.
  - Converting JSON knowledge graph to Clingo atoms and predicates.
  - **Query the knowledge base of course requirements.**



Queries

- The most interesting query of this project:

```
% Define courses (subset from code)
course(cse214, 4, "Undergraduate", 1, 1).
course(cse215, 4, "Undergraduate", 1, 1).

% Prerequisites for CSE214
:- course(cse214, 4, "Undergraduate", 1, 1),
   not course(cse114, _, "Undergraduate", _, _).

% Prerequisites for CSE215
:- course(cse215, 4, "Undergraduate", 1, 1),
   not course(ams151, _, "Undergraduate", _, _),
   not course(mat125, _, "Undergraduate", _, _),
   not course(mat131, _, "Undergraduate", _, _).
```

```
% Graduation requirements
% 1.
% Required intro courses
required_intro_course(cse220).
required_intro_course(X) :- course(X, _, _, _, _), X = cse215; X = cse150.
required_intro_course(X) :- course(X, _, _, _, _), X = cse114; X = cse214; X = cse216.
.
.
.
```

```
% Semester limits: at least 12 credits and at most 18 credits per semester
:- semester(Sem), SemCredits = #count { Credits, Course : schedule(Course, Sem), course(Course, Credits, "Undergraduate", _, _) }, SemCredits <
min_credits_per_semester.
:- semester(Sem), SemCredits = #count { Credits, Course : schedule(Course, Sem), course(Course, Credits, "Undergraduate", _, _) }, SemCredits >
max_credits_per_semester.

% Ensure all scheduled courses comply with seasonal offerings
:- schedule(Course, Sem), course(Course, _, "Undergraduate", Spring, Fall),
   Sem \ 2 = 1, Fall = 0.
:- schedule(Course, Sem), course(Course, _, "Undergraduate", Spring, Fall),
   Sem \ 2 = 0, Spring = 0.

% Count total credits and major credits
total_credits(Total) :- Total = #sum { Credits, Course : schedule(Course, _), course(Course, Credits, "Undergraduate", _, _) }.
major_credits(Major) :- Major = #sum { Credits, Course : schedule(Course, _), course(Course, Credits, "Undergraduate", _, _) }.

% Graduation requirements
:- total_credits(Total), Total < 120.
:- major_credits(Major), Major < 80.

% Objective to minimize the number of semesters
#minimize { 1, Sem : schedule(_, Sem) }.

#show schedule/2.
```

Testing & Evaluation

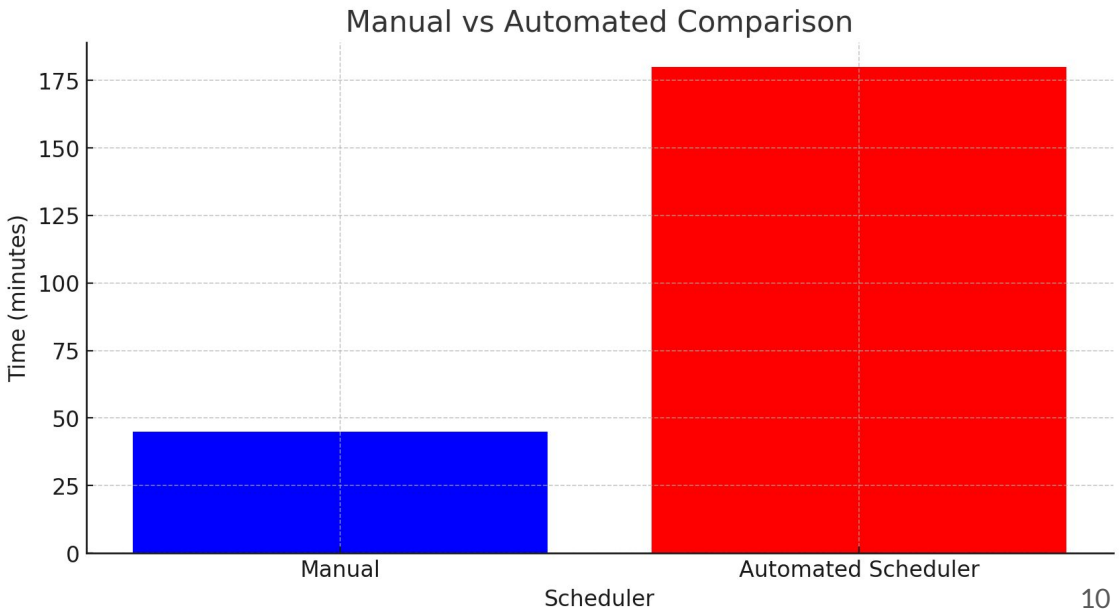
Testing

- Unit tests were implemented
  - Implemented using PyTest
  - Mainly on utility functions
- Testing of the program was difficult as it did not terminate

Testing & Evaluation

Evaluation

- Program was evaluated against a user (me)
  - Create a schedule spanning 12 semesters
  - Satisfies SBU CSE undergraduate requirements
- Comparison:
  - **Me**: 45 min.
  - **Automated Scheduler**: 3+ hours
    - Ended after user termination



Challenges & Summary

- SBU’s course catalog is not easily accessible for analysis
  - No API access available
  - Written in JavaScript
- Satisfying the constraint with such a large number of courses was very difficult
  - Due to:
    - Grounding overhead
    - Combinatorial explosion
- This approach is more appropriate for checking if degree requirements are/were satisfied

Acknowledgements

- Prof. Annie Liu – Project idea, and discussions
- Prof. Paul Fodor & Geoffrey Churchill – Discussions