Università della Svizzera italiana

Software Institute

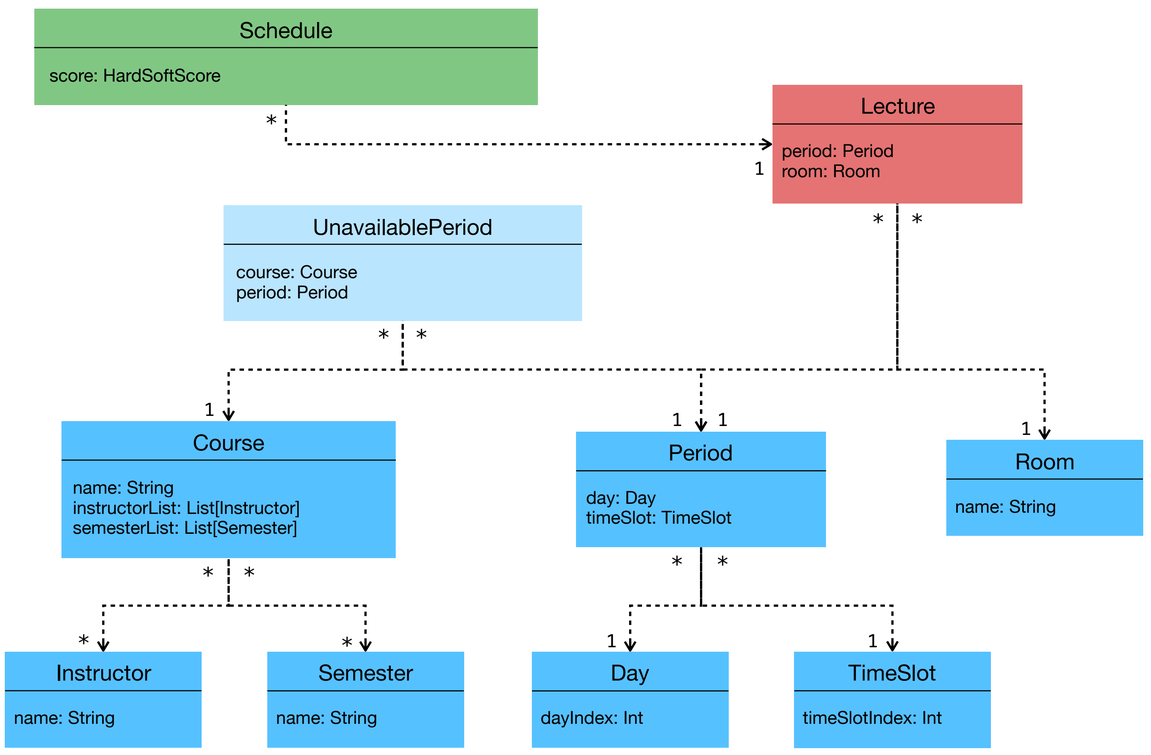
University Timetable Scheduler

Student: Aron Fiechter **Advisor:** Prof. Dr. Michele Lanza **Assistants:** Dr. A. Mocci, Dr. L. Ponzanelli

Background

Creating and managing timetables of courses is an issue for many institutions because of the various constraints that need to be respected in the planning. This becomes harder when the data needed to describe the problem and its constraints is not well curated and stored in several different places, a situation that often leads to a long and tedious manual work.

We automate the process using OptaPlanner, a state of the art constraint solver that expects a description of the **Solution** of the problem together with a mechanism to compute its score. The **Solution** contains a list of **PlanningEntities** (e.g., lectures) with customizable variables. The solver optimizes the score of the Solution by modifying the values of the variables. Finally, **ProblemFacts** specify the constant aspects of each problem instance (e.g., courses, rooms).



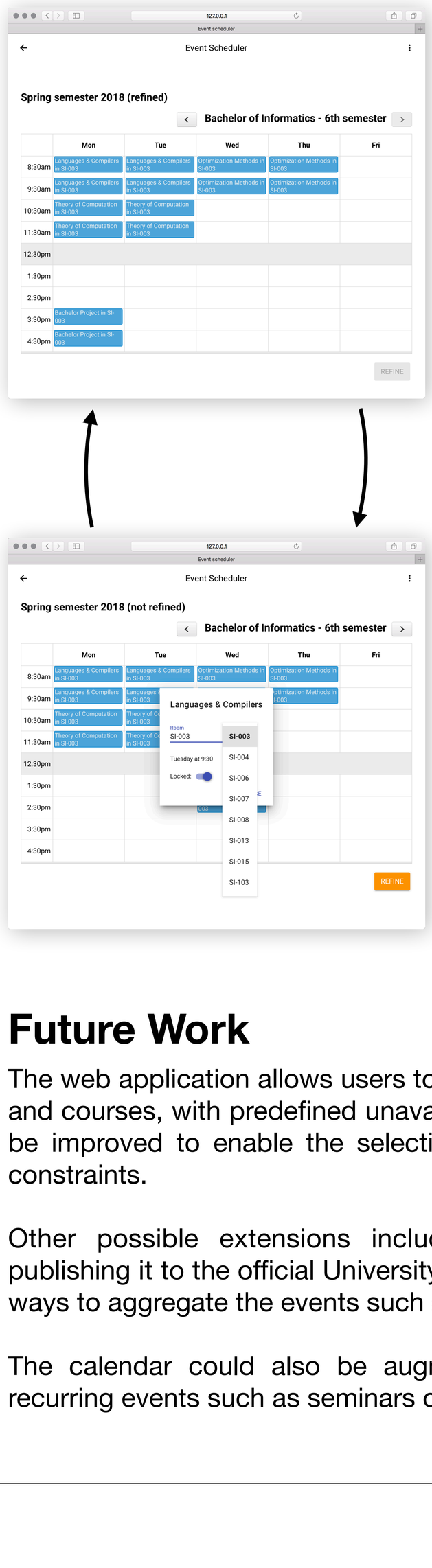
Constraints

The constraints include room capacity, room occupancy, the fact that an instructor cannot teach two courses at the same time, the conflicts between lectures of the same semester.

On the back end, all these constraints are given to the solver by specifying them using the language below.

```
createSchedule {
  courses {
    name
    instructors
    semesters
    studentSize
    numberOfLectures
    minDifferentDays
  }
  rooms {
    name capacity
  }
  unavailablePeriods {
    course period
  }
}
```

```
// Room capacity: A room's capacity should not be
// less than the number of students in the course.
// Each student above the capacity counts as 1 point
// of penalty.
rule "roomCapacity"
  when
    $room : Room($capacity : capacity)
    Lecture(
       room == $room,
       course.studentSize > $capacity,
       $studentSize : course.studentSize
    )
  then
    scoreHolder.addHardConstraintMatch(
       kcontext,
       ($capacity - $studentSize)
    );
end
```



Initial Solution

The input data is converted to the format needed by the solver, then the solver runs for some seconds and finally returns the computed schedule.

The schedule is shown as a weekly view separated by semester. Each lecture shows the course name and the room in which it takes place.

Editing and Refinement

Using the web application, once a schedule is opened, it is possible to move lectures to different periods and different rooms, and to lock lectures in place to prevent the solver from moving them if the schedule is sent to the backend for refinement.

In that case, the solver keeps existing locked lectures in place and builds a new improved solution around them.

Technologies

Polymer TypeScript FullCalendar

Scala GraphQL Sangria OptaPlanner

Future Work

The web application allows users to create a schedule for a given set of rooms and courses, with predefined unavailable periods. As a future work, the UI can be improved to enable the selection of specific courses, rooms, and other constraints.

Other possible extensions include approving a suitable schedule and publishing it to the official University calendar. This calendar may offer different ways to aggregate the events such as by room, instructor, user, or by day.

The calendar could also be augmented with the possibility to add non-recurring events such as seminars or conferences.

codelounge

REVEAL