Abstract

This article proposes a system for generating possible *University Classes Schedules*. It uses multi-agent negotiation to find satisfactory solutions to the problem, while trying to consider *personal preferences* of the represented people and institutions.

1 Implementation

1.1 University Classes

A class is an en event, that brings together a *group of students*, and a *professor* in certain *classroom* in order to learn/teach the specified *discipline*. It happens periodically, usually weekly, at the established *day of week* and *time*.

```
data Class time = Class { classDay        :: Day
                        , classBegins     :: time
                        , classEnds       :: time
                        , classDiscipline :: Discipline
                        , classGroup      :: GroupRef
                        , classProfessor  :: ProfessorRef
                        , classRoom       :: ClassroomRef
                        }
  -- redefined 'System.Time.Day' – no 'Sunday'
data Day = Monday | Tuesday | Wednesday
         | Thursday | Friday | Saturday
  deriving (Eq, Ord, Enum, Bounded, Ix, Read, Show)
```

The classes are negotiated by the interested parties: 1) students / groups, 2) professors, 3) classrooms. Each negotiation participant has a *timetable*, holding a schedule for one week, that repeats throughout the academic period. The *timetable* is actually a table: the columns represent days of week; the rows – discrete time intervals. Actual timetable structure may vary, as can be seen in figure 1.

```
class (Ord t, Bounded t, Show t) => DiscreteTime t where
  toMinutes   :: t -> Int
  fromMinutes :: Int -> t
class (DiscreteTime time) => Timetable tt e time | tt -> time
                                                 , tt -> e
                                                 , e  -> time
  where listEvents :: tt -> [e]
        eventsOn   :: tt -> Day -> [e]
        eventsAt   :: tt -> time -> [(Day, e)]
        eventAt    :: tt -> Day -> time -> Maybe e
```

	Mon	Tue	Wed	Thu	Fri	Sat
08:30 – 09:00						
09:00 – 09:30						
09:30 – 10:00						
10:00 – 10:30						
10:30 – 11:00						
11:00 – 11:30						
11:30 – 12:00						
⋮ ⋮						

(a) Timetable without recesses.

	Mon	Tue	Wed	Thu	Fri	Sat
08:30 – 09:10						
09:15 – 09:55						
10:05 – 10:45						
10:50 – 11:30						
11:40 – 12:20						
12:25 – 13:05						
13:15 – 13:55						
⋮ ⋮						

(b) Timetable with recesses.

Figure 1: Possible *timetable* structures.

One should distinguish the resulting timetables, shown in figure 1 and the timetable entity that holds an agent during the negotiation. The first one is immutable and is the result of agent's participation in the negotiation. The set of such timetables, produced by every the participant, is the **university schedule** for given academic period.

During the negotiation, an agent's inner timetable gets changed on the fly, in order to record agreements made. This means that we are dealing with *side effects*, that need to be explicitly denoted in Haskell. The following definition leaves it free to choose the monad abstraction for those effects.

$$
\begin{aligned}
&\textbf{class}\ (\mathit{DiscreteTime}\ \mathit{time}, \mathit{Monad}\ m) \Rightarrow \\
&\quad \mathit{TimetableM}\ \mathit{tt}\ m\ e\ \mathit{time} \mid \mathit{tt} \to \mathit{time} \\
&\qquad\qquad\qquad\qquad\quad , \mathit{tt} \to e \\
&\qquad\qquad\qquad\qquad\quad , e \to \mathit{time} \\
&\quad \textbf{where}\ \mathit{putEvent} \quad :: \mathit{tt} \to e \to m\ \mathit{tt} \\
&\qquad\quad\ \mathit{delEvent} \quad :: \mathit{tt} \to e \to m\ \mathit{tt} \\
&\qquad\quad\ \mathit{ttSnapshot} :: (\mathit{Timetable}\ \mathit{ts}\ x\ \mathit{time}) \Rightarrow \mathit{tt} \to m\ \mathit{ts}
\end{aligned}
$$

1.2 Negotiating Agents

As it was mentioned before, the schedule is formed in a negotiation between *professors*, *groups* and *classrooms*. To distinguish those three types of participants, agent's role is introduced. The role: 1) identifies the kind of person/entity, represented by the agent; 2) defines agent's reaction on the messages received; 3) defines agent's goal.

A *representing agent* is a computational entity, that represents a *real person or object* in it's virtual environment. In current case, it represents one's interests in a *negotiation*. Such an agent must

(1) pursue the *common goal* – it must consider the common benefits, while being egoistic enough to achieve it's own goal;

(2) respond to the messages received in correspondence with (1);

(3) initiate conversations (send messages, that are not responses), driven by (1);

(4) become more susceptible (less egoistic) with passage of time.

1.2.1 Common Goal

1.2.2 Messaging

1.3 Coherence

1.3.1 Contexts

1.3.2 Obligations

1.3.3 Preferences

1.3.4 External

1.3.5 Decision

1.4 Agent

Here follows *agents* implementation.