Department Timetabling with OPL Studio

Author: Paul Ahern

2 April 2007

"When to the sessions of sweet silent thought,
I summon up remembrance of things past,
I sight the lack of many a thing I sought,
And with old tears new wail my dear time's waste."

- William Shakespeare, sonnet 30

Qualification: MSc Intelligent Systems for Business and Manufacturing

Institution: National University of Ireland, Cork.

Department: Computer Science, Science Faculty, UCC

Submitted: April 2007

Department Head: Gregory Provan

Supervisor: Marc van Dongen

Course: CS5001

Report version: 1.0

Declaration: I confirm that, except where indicated through the proper use of citations and references, this is my own original work and that I have not submitted it for any other course or degree.

Abstract

University Timetabling is about coordinating the activities of teachers and students to provide a service.

It is important to maximise the use of scarce resources, rooms and teachers' time, while not asking too much from the individuals involved.

Timetables for the Computer Science department are currently worked out by hand. This requires considerable experience and takes up a lot of time.

This project is about using Constraint-Based programming techniques to generate timetables for the department automatically.

Project success is measured by the production software which can generate valid timetables. This provides a foundation for future work to add additional functionality.

Timetable drawn up at the department-level in UCC are impacted by many factors beyond the control of an individual department. Such as room availability and modules provided by other departments. These remove many of the specific timetabling decisions from the scope of this project.

One aspect of the OPL Studio 3.7 package which is not optimised is data entry. A relatively large amount of mutable data needs to be passed to that tool. A separate optimisation problem to be solved is to minimise the amount of keying inolved in entering that data.

Contents

Chapter 1

Introduction

Thanks to all those who have helped me in researching the problem and compiling this report. Particularly Leslie Brooks, who has the thankless task of drawing up timetables for the Computer Science department.

MOTIVATION

Module CS5202, Constraint-Based Programming, is a required module for the MSc Intelligent Systems for Business and Manufacturing. This module requires use of the OPL Studio version 3.7 product.

This project CS5001 proposal also requires the use of OPL Studio, to automate the generation of university timetables. Combining this project with the CS5202 module seemed like a good idea at the time.

In this project it is a given that OPL Studio 3.7 will be used to try to solve the problem. Part of the purpose of the project is to determine how practical an approach this is.

This report is intended to be of use to students starting future iterations of the MSc Intelligent Systems for Business and Manufacturing. The software developed and lessons learned in the course of this project could aid further research in this area.

1.1 Problem Specification

Build a tool, based on the OPL Studio 3.7 product, which can generate timetables for the Computer Science department of UCC.

1.1.1 General Timetabling Issues

Any set of activities involving the shared use of scarce resources needs a schedule. An optimum schedule is one which makes the best, or most efficient, use of the scarce resources.

Any university timetable must take into account the availability of various resources:

- Lecturers
- Rooms
- Students

The standard physical constraints must be respected:

- No lecturer can teach more than one class at a time.
- No room can accommodate more than one class at a time.
- No student can attend more than one class at a time.

Each resource has attributes or capacities which need to be respected:

- Module Taught (for lecturers)
- Size (for rooms)
- Course Taken (for students)

Unless a new institution or department is being created timetables are not created from scratch. More common is the need to modify an existing timetable. The amended timetable should take new or changed constraints into account while minimising the disruption.

1.1.2 Timetabling at UCC

The timetable evolves over the years. If the curriculum doesn't change then the timetable remains the same. The departments do not own or control most of the rooms used for lectures. These have to be allocated by the Room Booking section.

Departments draw up their own timetables, but their courses often include modules provided by other departments. The timing of the classes for such modules is agreed at a faculty level. From the departmental timetable point of view, their position is fixed.

Year One computer Science, Term 2

Time	Mon	Tue	Wed	Thur	Fri
09:00	CS1100	MA1054	CS1100	MA1054	CS1100
10:00		CS1101			
11:00		MA1003	MA1003	EC1401	CS1102
12:00				EC1401	
13:00			CS1101		
14:00		CS1102	MA1054		
15:00			MA1015		
16:00	MA1015				
17:00					

Computer Science Other Department

Figure 1.1: Sample Manual Timetable

1.2 Goals

In addition to analysing the problem and modelling it in OPL Studio 3.7 there are some software engineering requirements:

1. A human-readable data structure to hold the specific timetabling requirements of the department in such a way as to minimise the amount of data entry keying.
2. An application to translate this data into a form which OPL Studio 3.7 can process.
3. An application which can display the timetables generated by OPL Studio 3.7 in a more human-readable form.

1.3 Document Outline

This document has the following structure:

Introduction This introductory section.

Background Terminology used, literature review, descriptions of the technologies and methodologies used in the project.

Details Analysis of the problem, Design and Implementation of the Tool.

Conclusion What has been achieved in the course of this project.

Appendices Supporting technical documentation, sample data as well as a Glossary and Bibliography.

Chapter 2

Background

2.1 Nomenclature

This report uses the following terms to define the structure and requirements of the timetable:

Class Individual timetable entry. A meeting of a group of students and a lecturer in a room at a specific time.

FixedClass Timetable entry which cannot be changed (e.g. a Class supplied by another department).

KnownClass Timetable entry which can be changed, if necessary to find an overall solution.

Module Set of Classes about a subject.

Course Set of Modules attended by a group of students.

CourseGroup Set of courses which are available to students enrolled for a specific year.

Room Venue for Classes.

Lecturer Teacher of Classes.

2.2 Literature Review

A review of the literature on generating timetables reveals earlier attempts to use the constraint-based approach, notably [Azevedo94], [Gueret96] and the more recent [Gavanelli06].

Seeing the approaches adopted in these cases is useful as examples of how timetabling requirements may be expressed as constraints. However, the timetabling requirements of each university are radically different in their details. Each institution deals with even the common timetabling issues in different ways.

For instance:

- Faculties at the New University of Lisbon control their own classrooms and their classes can be of varying lengths [Azevedo94].

- Timetables at Institute of Applies Mathematics in France use the quarter of an hour as their unit of time, give the students a break of one time unit between classes. While the full group attends lectures half- or one third-groups attend practical classes (some of which are given by students) [Gueret96].
- Students from different years can attend the same classes at Ferrara University. Too many optional courses are offered for all possible variations to be accommodated within a timetable, so fixed patterns of available hours are used and each course assigned to one of these sets of hours [Gavanelli06].

This makes it difficult to directly apply the lessons of these earlier attempts in the current project. It is worth noting that the peculiarities of each institution's approach to timetabling drive the shape of the solution adopted.

2.3 Problem Requirements

The normal, physical constraint, timetabling requirements apply:

- No room should have two classes scheduled at the same time.
- No lecturer should have two classes scheduled at the same time.
- No group of students should have two classes scheduled at the same time.

University College Cork has some specific structural requirements:

- Lectures begin on the hour from 09:00 to 17:00, Monday to Friday, with no slack time between them.
- Departments do not control their own rooms and lecture venues must be requested from Room Booking.

Additionally, for the Computer Science Department at UCC has further requirements:

- Some courses include modules provided by other departments. Their scheduling is fixed from the Computer Science department's perspective.
- Lecturers should teach modules for which they have expressed a preference, with priority given to higher expressed preferences.

2.4 Constraint-Based Programming

Computer programs in traditional, imperative, computer languages such as COBOL and Java consist of data structures and instructions. The instructions tell the computer what to do and the data structures provide space to hold the information that the program processes.

For example, here is a Java program to work out all even numbers between one and ten:

```
for (int i = 1; i < 11; i++) //Assign variable i the values from 1 to 10
  if (i % 2 == 0) //If i is evenly divisible by 2
    System.out.println(i); //Write out value in i
```

The approach of Constraint-Based programming is very different.

A set of variables is defined and restrictions are placed on the values which they can assume. The computer then tries to determine values which are allowed by with these restrictions.

Here is an equivalent example of finding even numbers, this time in OPL:

```
range ValidRange 1..10;
var ValidRange i; //Declare variable
solve { //Find all valid values of i
i mod 2 = 0; //The contents of i is evenly divisible by 2
};
display i; //Write out the values in i
```

The constraint-based programming approach involves specifying a problem in terms of ranges of values which are allowed or are forbidden in valid solutions. The OPL Studio product then searches using general or specific algorithms for solutions which meet these constraints.

Note The term *variable* means different things in imperative and constraint-based programming. In imperative programming a variable is simply a data-structure in the computer's memory where a program can store data. In constraint-based programming the variables are those data-structures whose values are being searched for. Constraint-based programs will also contain data-structures of the more traditional kind.

2.4.1 Modelling problem using Constraints

Competition between Courses for shared resources makes it necessary that all the timetables for the department be calculated together. A single OPL model is used to try to find values in the following variables for each Class:

- Start time
- Room
- Lecturer

The values assigned to these variables make up the timetable solution.

These variables, together with the constraints and supporting data make up the OPL model which is solved by OPL Studio 3.7 to create a timetable for the department.

The constraints include:

- Preventing the scheduling of two Classes in the same room at the same time
- Ensure that the assigned Lecturer is qualified to teach the Module which the Class is a part of.
- Ensure that the same Lecturer teaches all the Classes for a single Module.

Chapter 3

Details

3.1 Design

3.1.1 Current Process

Departmental timetables are drawn up manually. This takes a lot of skill and a lot of time.

The problem is over-constrained. There are more competing requirements than there are hours available in the timetables to satisfy them all.

An element of intuition is required to finalise each year's timetables.

3.1.1.1 Annual Steps in drawing up Timetables

Feb-Mar Start collecting information for modules to be taught in the following year. E.g. new courses to be added to existing structure; Who is available to teach them; Likely class sizes (needed for room booking).

Apr Availability of rooms determines the timetable. Rooms booked for academic year; Bookings rolled forward to following year.

End of July Room Bookings "finalised" (though rooms can still be taken away).

Jul-Aug Final spreadsheet of room allocations issued. Timetable now more or less stable.

3.1.1.2 Complicating Factors

- Some classes, which are provided by Computer Science to students of other departments, have their timetables set at the faculty level (e.g. Classes for Mathematical Science or Chemical Science).
- Joint Honours courses are on the timetables of two departments (e.g. CS have joint honours courses with Business and Finance streams. Timetables of each department need to avoid clashes with the other.
- Practicals have to be timetabled around lectures. Sometimes the Rooms available are not big enough for entire classes so the is class split into several groups.

- Practicals may be run by Students from other years. This introduces inter-year dependencies.

3.1.2 Simplifying Assumptions

This project is undertaken as part of a taught MSc course. There is only a very limited amount of time available. In order to produce a working tool, various simplifying assumptions are made:

- Departments at UCC do not control their own rooms. For the purposes of this project it will be assumed that a known set of rooms is available to the Computer Science Department for the entire academic year.
- It is assumed that complete consistent data can be provided to allow OPL Studio 3.7 to calculate valid solutions, if any, to the timetabling questions posed. For instance, there must be a Lecturer available for every Module or rather at least one Lecturer must have expressed a preference for each Module.
- The timetabling of practicals is excluded from the model.

One of the goals of this project is to produce a system which can act as a foundation for the development of more sophisticated tools and models in future years, which can take these complications into account.

3.1.3 Model

It seems natural to express the requirements of a University Timetable as constraints.

DEFINITIONS

Let C be the set of classes making up the timetables of the department.

Let R be the set of rooms which are available to the department in which classes can be held.

Let L be the set of lecturers available to teach classes.

Each scheduled class $c \in C$ will have a timetable entry consisting of:

- Start time $c.s \in \{0 \dots 44\}$, an hour in the working week.
- Room where the class is held $c.r \in R$, and
- Lecturer teaching the class $c.l \in L$.

These are the variables for which valid combinations of entries are searched.

The structure of the courses taught is captured in the model by grouping classes together into Modules, M, and CourseGroups, $\mathbb{C}$. A CourseGroup is a set of Modules or Classes which must coexist on a single timetable without overlaps.

CONSTRAINTS

- No two Classes in a CourseGroup can begin at the same time: $\forall c, c' \in \mathbb{C}, s \in \{0 \dots 44\} : c.s \neq c'.s$.

- The same Lecturer teaches all the Classes in a Module: $\forall c, c' \in m, m \in M, l \in L : c.l = c'.l$.

The model must contain information about the numbers of students in each Class, $ClassSize_c \in \mathbb{Z}^+$, and the capacities of the Rooms, $RoomSize_r \in \mathbb{Z}^+$. Both of these amounts are positive integers. Then an additional constraint may be imposed:

- No Class can be held in too small a Room: $\forall c \in C : ClassSize_c \leq RoomSize_{c.r}$.

The model also needs to contain lists of Modules which Lecturers are able to teach, $CanTeach_l \subset M$.

- Class must be taught by a suitable Lecturer: $\forall c \in m, m \in M : m \in CanTeach_{c.l}$.

The physical constraints of space and time are modelled:

- If two Classes start at the same time they must be in different Rooms, $\forall c, c' \in C : c.s = c'.s \Rightarrow c.r \neq c'.r$.
- If two Classes start at the same time they must have different Lecturers, $\forall c, c' \in C : c.s = c'.s \Rightarrow c.l \neq c'.l$.

Additional constraints are needed to ensure that any FixedClasses, F, keep their position in the timetable. Each element in F consists of a Class c, its Start Time $c.s$, Room $c.r$ and Lecturer $c.l$.

- Timetable entries provided for FixedClasses must be used, $\forall c \in C, f \in F : c = f.c \Rightarrow c.s = f.c.s, c.r = f.c.r, c.l = f.c.l$.

Finally, as many KnownClass entries, K, as possible should be used. Each element in K consists of a Class c, its Start Time $c.s$, Room $c.r$ and Lecturer $c.l$.

- Maximise the number of KnownClass entries used in the timetable, $maximize \Sigma_{k \in K, c \in C}(k.c.s = c.s \wedge k.c.r = c.r \wedge k.c.l = c.l)$.

3.1.4 Architecture

OPL Studio 3.7 is a stand-alone product which allows problems to be expressed in terms of constraints. This is done by writing a model of the problem in the OPL language. The model is solved and any solutions found are output.

A model file can be combined with a data file (creating an OPL Studio 3.7 *Project*). The same model can be reused with different sets of data.

OPL Studio 3.7 also includes a scripting language which is used to pass data to a model and collect and format the results obtained. This is used in the project to pass data into and get results out of the model.

OPL and OPL Studio 3.7 are not particularly user-friendly or well-documented. Particularly to one with no background in constraint-based programming. They should be hidden from the user by an interface designed for usability.

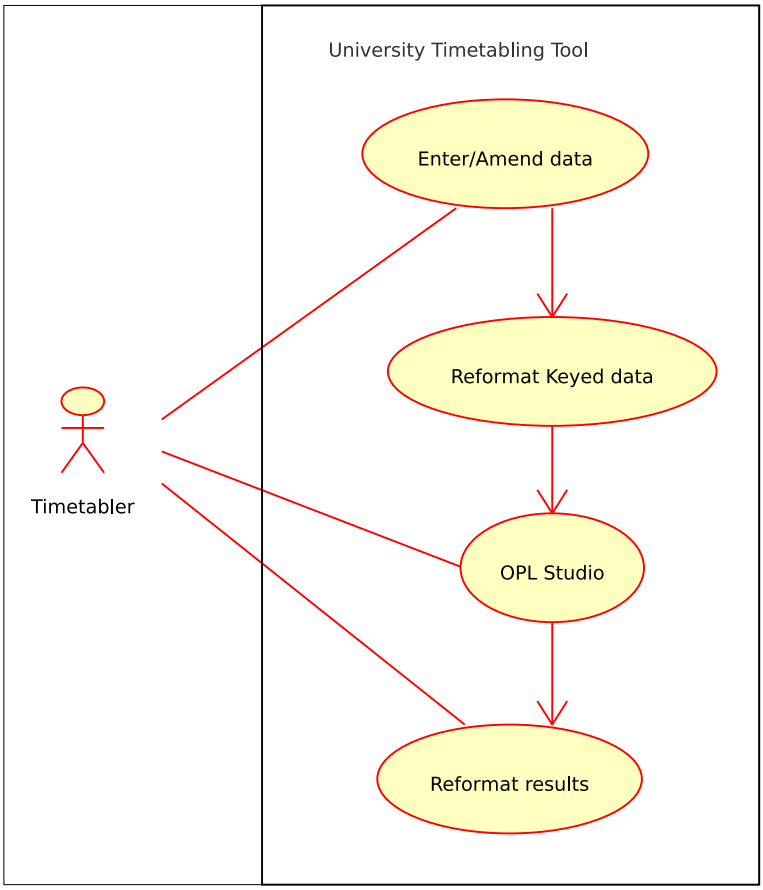

Figure 3.1: Use Case Diagram

Unfortunately, although there is a version of OPL Studio 3.7 which can be run in batch-mode, as a back-end solver to another application, the licence for this is prohibitively expensive. So the interactive version is used by this project.

In order to minimise the user's exposure to OPL and OPL Studio 3.7, two Java applications are used. One formats the data provided by the user into an OPL Studio 3.7 data file; The second formats the timetable results output by OPL Studio 3.7 to make them more human-readable.

3.1.5 Data

There are three types of data in the tool:

1. Data keyed by the user.
2. Data derived from the keyed data by a Java Application.
3. Solution data calculated by OPL Studio 3.7.

3.1.5.1 Keyed Data

Data is keyed by the user into a structure which was designed to:

- Minimise the amount of user keying.
- Use a simple and widely compatible data format.
- Include all of the information required to capture the timetabling requirements.

To meet these objectives, the user-entered data is stored in a comma-delimited format file. This is supported by spreadsheet products (including Microsoft Excel and OpenOffice Calc), which can be used to enter the data.

Each item of data is entered only once. So, for example, the Modules which each Lecturer can teach is keyed. The Lecturers who can teach each Module is derived from the keyed data.

The full data structure is detailed in the appendix to this document.

There are two main types of data - Unitary and Relational.

UNITARY ITEMS

- Room names and their capacities.
- Module attributes such as the (expected) number of Students.
- Fixed Classes, timetable entries which cannot be changed.
- Known Classes, timetable entries which can be amended if necessary to find a solution - e.g. entries on the previous year's timetable.

RELATIONAL ITEMS

- Courses which make up each CourseGroup.
- Modules which make up each Course.
- Modules, in order of preference, which each Lecturer is available to teach.

3.1.5.2 Derived Data

The user-supplied data is reformatted in order to be compatible with OPL Studio.

Additional data is derived from that keyed. For instance the names of all of the Lecturers can be generated from the list of Lecturer preferences. Each entry on the latter consists of the name of a Lecturer and the names of the Modules which that person can teach.

Extra Lecturer and Room names are captured from the FixedClass entries. These need to be included in the derived data as the model uses the same variable structure for all entries on the timetable.

3.1.5.3 Variable Data

OPL Studio 3.7 runs a script which combines the reformatted derived and keyed data with the model to calculate valid values for the variable fields: Class Times, Rooms and Lecturers.

The results are written out by the script to another comma-delimited format file.

3.1.5.4 Results

A second Java application is run to display the results. Each CourseGroup timetable is displayed in a separate window. The Classes in a CourseGroup do not overlap in time.

3.2 Implementation

3.2.1 Prototype

A prototype based on the Architecture described above was build.

It contains simple versions of the Java applications and of the OPL model.

The prototype generates timetables which meet the criteria specified in the test data and the constraints. This proves that the proposed design is valid and can produce solutions.

3.3 Detailed Requirements

Real departmental data, where available, is used in the final version of the tool.

Lecturer preference information was obtained, with the names of the members of staff removed. The missing names were replaced with those of the early roman emperors, to provide more readable results.

The structure of the Courses is obtained from the timetables manually created for the 2006-07 academic year. This gives the following CourseGroups:

- Year One (Computer Science)
- Year Two (Joint Hons CS + Economics - Financial Stream)
- Year Three (Joint Hons CS + Economics)
- Year Four (Joint Hons CS + Economics - Financial Stream)
- Year Four (Joint Hons CS + Economics - Business Stream)
- Year Five (MSc)

Room names are taken from this year's timetable. They are processed by the tool as if they are exclusively for the use of this department.

The names of lecturers from other deparments are not available on the timetables available from the department. They are entered in the keyed data merely as unknown.

Errors in the data reveal that not all Modules are covered by Lecturer preferences. It was assumed in the design that the data would be complete and consistent.

Extra roman emperors are manually included to provide cover for the orphan Modules. It is likely that the data provided was merely incomplete.

The process of deriving data could be changed to add any Modules for which no Lecturer has expressed a preference to the end of the list of preferences of every Lecturer.

3.3.1 Application Development

The Java application which formats and derives additional data from the user keyed data was completely reimplemented, from the version in the prototype, to use an Object Oriented design.

This is more in keeping with the style of the Java language and will make it easier to amend the program to add more input data or generate new derived data.

All of the data used by the OPL model is provided in the form required by the constraints. This simplifies the model as it no longer has to preprocess data before using it to solve the problem.

OPL does not lend itself to data manipulation, so it makes sense to move this processing into the Java application.

3.4 Evaluation

The usefulness of the timetables generated by the tool is a function of the quality of the data provided and the sophistication of the model used.

All of the room and course structure information used is manually entered from paper copies of the 2006-07 timetables, so there is scope for error there.

The Lecturer preference data seems to be incomplete.

Timetables are generated which meet the criteria specified in the model and data. To that extent the project is a success.

The development of a data structure and applications to pass data in to and out of OPL Studio 3.7 could provide a foundation for further work in this area.

Chapter 4

Conclusion

It is possible to draw up timetables for the department with OPL Studio 3.7. It is just not practical to do so.

4.1 Summary of Project

OPL Studio 3.7 facilitates the development of constraint-based solutions to many classes of problems. Sadly, university timetabling, at least in the case of a department at UCC, is not one of them.

In the beginning, this project was mainly about modelling the departmental timetables in OPL. As the work progressed, however, it mutated into more of a software development project.

This suits me as I am far more likely to be using Java than OPL in the future.

The change in emphasis in the project was driven by the difficulty of feeding large amounts of mutable data into OPL Studio 3.7.

The structure of the data is related to the structure of the model and a means was needed of modifying both in parallel as the design was refined.

This led to the development of a data structure which captured all of the information about the timetables while minimising the amount of keying on the part of the user. The latter was done both to increase usability and reduce the incidence of keying errors.

Someone with more time and more knowledge of constraint-based programming, or Operations Research, than I could build a more sophisticated model.

However, the problem of timetabling for the department is over-constrained. There are no solutions which meet all of the constraints.

The timetabling requirements of the department are inconsistent and too ill-defined to make this a good candidate for automated timetabling using constraint-based programming. Apart from the physical constraints of time and space, almost every restriction is relaxed from time to time.

Further, in some cases, the position and contents of the majority of the entries on one of the departmental timetables is driven by external factors. In other words there is no decision to be made as the required answer is provided.

Drawing up timetables is more of an art than a science. Competing requirements must be reconciled. I can see no optimum way of doing this automatically.

These decisions are ultimately political.

If the OPL approach is persisted with, instead of a timetabling expert, the department would have to retain the services of an expert in OPL. Since the timetabling model would have to be adjusted each time the constraints changed or new requirements were discovered.

4.1.1 OPL Studio Shortcomings

The manufacturer of OPL Studio is a company called ILOG. It seems to me that ILOG to gives insufficient emphasis to usability and customer service. Examples of this encountered on the project:

1. Their website structure is confusing and difficult to navigate.
2. The implementation of their licencing for their products is highly restrictive.
3. Between OPL Studio version 3.7 and version 5.0 they released six mutually incompatible versions of their product. Each new version changed the syntax of the OPL "Language".

The Optimisation Programming Language is no more a language than is the Structured Query Language.

While small problems can be modelled directly, systems as complex as timetabling require that more thought be given to usability. A front-end, separate from the solver, is needed to handle the interface to users who are not experts in OPL.

Should efforts to find a practical Constraint-Based approach to this problem continue, then I recommend that OPL Studio should not be used until it has matured.

4.2 Avenues for future work

The software developed in the course of this project is available for as a basis for future work. The data structure and model can be extended to more closely approximate the stated requirements of the department.

A more practical approach would be to develop tools which can analyse existing timetables to find which constraints are most tight.

These tools could present alternative configurations of timetable entries when circumstances change.

In order to even begin the process of drawing up a timetable for a single department, several simplifying assumptions are needed. Also an entire new data structure is needed to take account of external impacts. Timetabling at the level of the institution could avoid some of these issues.

University-level timetabling is a larger problem but is, in at least some respects, a simpler one.

Syllabus Plus was used a few years ago to generate timetables within UCC. Apparently, the solutions it generated were politically unacceptable.

Chapter 5

Appendices

5.1 Data Structures

5.1.1 Data Structure

5.1.1.1 Keyed Data

This data is entered before the timetabling process can begin.

Structure Name	Field Name	Data Type	Description
WeekLength	WeekLength	Numeric	Number of working hours in the week (45)
DayLength	DayLength	Numeric	Number of working hours in the day (9)
DayName	1-5 in array	String	Names of the days of the week
Preferred Times	1-9 in array for each day of week	Numeric	Weightings for hours during each working day when lectures should be scheduled
CoursesIn Group	Name	String	Name of CourseGroup
	CoursesIn Group	Set	Set of Names of Courses
ModulesIn Course	Name	String	Name of Course
	ModulesIn Course	Set	Set of Names of Modules
Module	Name	String	Name of Module
	NrLectures	Numeric	Number of Lectures per week
	NrTerms	Numeric	Number of Terms this Module is in
	NrStudents	Numeric	Number of Students taking this Module
Modules Taught	Name	String	Name of Lecturer
	Modules Taught	Set	Set of Names of Modules Lecturer is available to teach

Structure Name	Field Name	Data Type	Description
Room	Name	String	Name of Room
	NrStudents	Numeric	Number of Students this Room can accommodate
	Projector	Boolean	Does this room contain a Projector
	PCs	Boolean	Does this Room contain PCs (i.e. is it a lab)
FixedClass	Name	String	Name of Class
	Start Day	DayName	Weekday the Class starts on
	Start Time	HourName	Hour of the day Class starts on
	Term	String	Terms this Module is in (First, Second or Both)
	Room	String	Name of Room Class is in
	Lecturer	String	Name of Lecturer teaching Class
KnownClass	Name	String	Name of Class
	Start Day	DayName	Weekday the Class starts on
	Start Time	Numeric	Hour of the day Class starts on
	Term	String	Terms this Module is in (First, Second or Both)
	Room	String	Name of Room Class is in
	Lecturer	String	Name of Lecturer teaching Class

FixedClass items represent timetable entries which are beyond the control of the department (e.g. Classes provided by other departments) and thus cannot be amended. KnownClass items represent entries on an existing timetable which may be amended in the search for a valid solution, but which should be retained, if possible.

5.1.1.2 Data Derived from Keyed Data

More data is derived from the Keyed Data before OPL Studio 3.7 is run.

Generation of Class Names: Each Class occupies a single slot in a timetable. Class Names are generated from the Module Name by adding a single alphabetic character (a-i in turn). The NrLectures of the Modules determines the number of Class Names to generate.

Generate additional data:

- Derive additional Module names from the FixedClass data (NrStudents defaults to zero).
- Derive additional Lecturer names from the FixedClass data.
- Derive additional Room names from the FixedClass data (NrStudents defaults to zero).

Remove redundant data:

- Remove Modules from the ModulesTaught data if they are not present on the timetable.

Structure of the derived data:

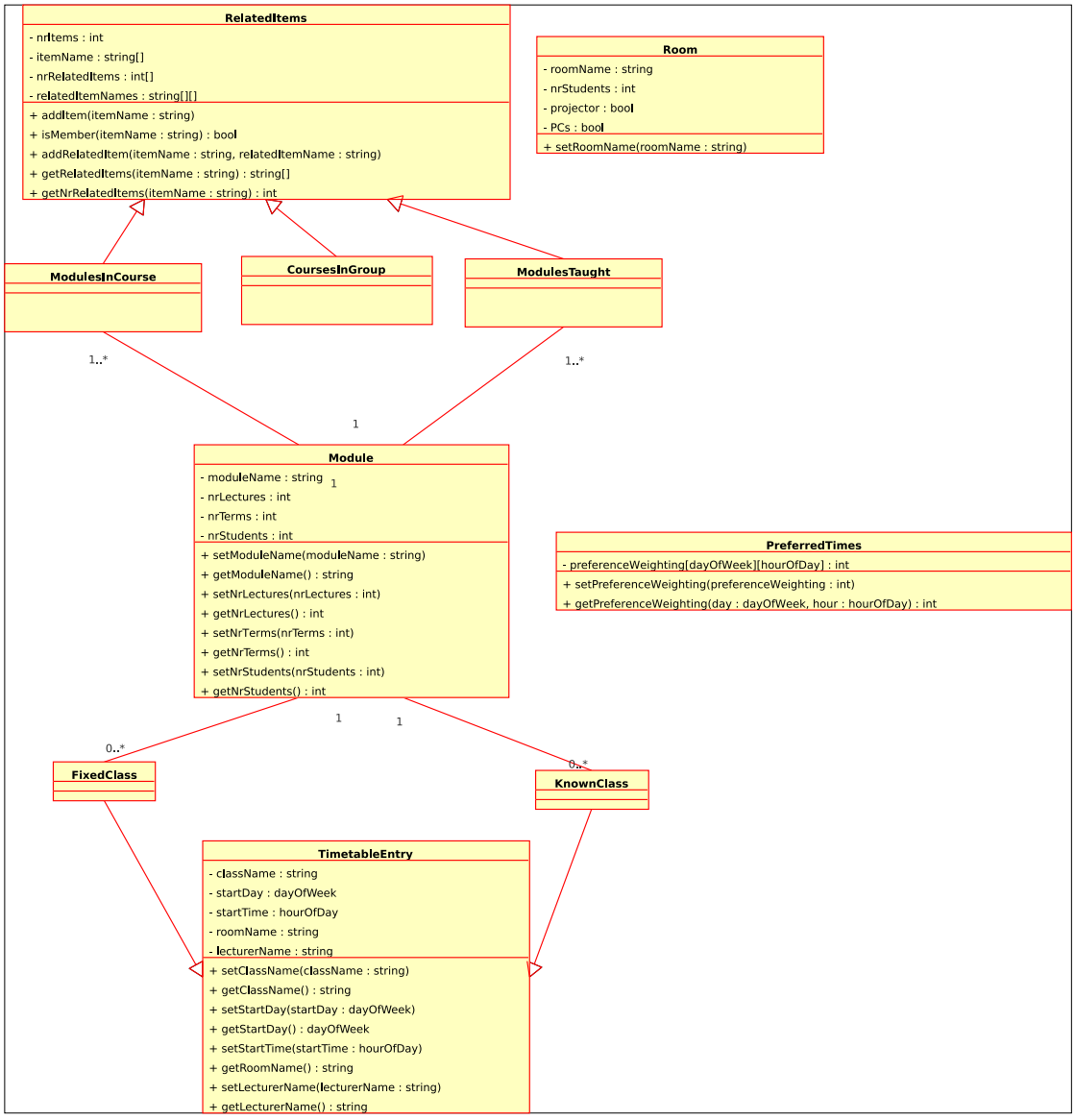

Figure 5.1: Derived Data Structure

Structure Name	Field Name	Data Type	Description
Class	Class	String	Name of Class
Lecturer	Lecturer	String	Name of Lecturer
Room	Room	String	Name of Room
Class	Class	String	Name of Class
	NrStudents	Numeric	Number of Students taking this Class
	CanTeachClass	Set	Set of Lecturers who can teach the Class
CoursesOf Module	Name	String	Name of Module
	CoursesOf Module	Set	Set of Courses that contain the Module
CourseGroups Of-Course	Name	String	Name of Course
	CourseGroups OfCourse	Set	Set of CourseGroups that contain the Course
ClassesIn Module	Name	String	Name of Module
	ClassesIn Module	Set	Set of Names of Classes contained in the Module
ClassesIn CourseGroup	Name	String	Name of CourseGroup
	ClassesIn CourseGroup	Set	Set of Classes contained in the CourseGroup
LecturersFor Module	Name	String	Name of Module
	CanTeach Module	Set	Set of Lecturers who can teach the Module
ModuleSize	Name	String	Name of Module
	NrStudents	Numeric	Number of Students taking this Module
RoomSize	Name	String	Name of Room
	NrStudents	Numeric	Number of Students this Room can accommodate

5.1.1.3 Variable Data

These are the variables whose values are determined by OPL Studio 3.7.

Structure Name	Field Name	Data Type	Description
Class	Term	String	Term the Class is in
	StartDay	DayName	Weekday the Class starts on
	StartTime	Numeric	Hour of the day Class starts on
	Room	String	Name of Room Class is in
	Lecturer	String	Name of Lecturer teaching Class

5.1.1.4 Keyed data validation:

- Ensure that each Module appears at least in the set of preferences of at least one lecturer. If not then the reformatting proceeds as normal, but an error message is issued.
- Ensure that Module details are found, or derived, for all details in each ModulesInCourse entry. If not a message is displayed.

5.2 Code and Test Data

The most recent version of the code and test data is available for download from www.ahernp.com/docs/CS5001.zip. Instructions on how to run the tool are included.

Steps to generate a set of timetables:

1. Edit keyedData.csv to include any changed timetable requirements
2. Reformat the keyed data into derived.dat for OPL Studio:
   java -jar reformatKeyedData.jar keyedData.csv derived.dat
3. Run OPL Studio and execute script timetable.osc to create timetable.csv
4. Display the timetables in a more human-readable form:
   java -jar displayTimetable.jar timetable.csv

5.3 OPL Script

```
ofile resultFile("timetable.csv");
resultFile << "TimeTable,Slot,Day,HourOfDay,Module Lecturer Room" <<endl;
Model m("timetable.mod",
"derived.dat");
if m.nextSolution() then
forall(cg in m.CourseGroups)
forall(c in m.ClassesInGroup[cg]) {
resultFile << cg << ","
<< m.StartTime[c] << ","
<< m.dayNames[m.StartTime[c]/m.dayLength] << ","
<< (m.StartTime[c] mod m.dayLength) + 9 << ":00,"
<< c << " "
<< m.Lecturer[c] << " "
<< m.Location[c]
<<endl;
}
resultFile.close();
```

5.4 OPL Model

```
int dayLength = ...; //Working hours in each day
int weekLength = ...; //Working hours in each week
```

```
string dayNames[0..4] = ...;
range hourOfDay 1..dayLength; //Working hours in each day
enum dayOfWeek = ...; //Working days of the week
enum terms = ...; //Terms in the academic year
range StartTimes 0..weekLength-1;
//Weighting of hours of the day when Classes should be scheduled
int preferredTimes[StartTimes] = ...;
//Basic Types
enum CourseGroups = ...; //Group of courses whose classes must not overlap
//enum Courses = ...;
enum Modules = ...;
enum Classes = ...;
enum Lecturers = ...;
enum Rooms = ...;
//Relations
setof(Classes) ClassesInGroup[CourseGroups] = ...; //Classes in CourseGroup
setof(Modules) ModulesTaught[Lecturers] = ...; //Modules which can be taught by each
setof(Classes) ClassesInModule[Modules] = ...; //Classes in Modules
setof(Lecturers) LecturersForModule[Modules] = ...; //Lecturers available to teach mo
//Attributes of Basic Types
int ClassSize[Classes] = ...;
int RoomSize[Rooms] = ...;
//Known and fixed entries on timetable
//Known entries are suggested solutions and may be changed; Fixed cannot be amended
int nrFixedClasses = ...; //Number of Classes whose position in timetable is fixed
int nrKnownClasses = ...; //Number of Classes whose possible position in timetable is
range fixedClasses 1..nrFixedClasses;
range knownClasses 1..nrKnownClasses;
range Bool 0..1;
struct KeyedClasses {
  Classes class;
  StartTimes st;
  terms term;
  Rooms room;
  Lecturers lecturer;
};
KeyedClasses FixedClass[fixedClasses] = ...;
KeyedClasses KnownClass[knownClasses] = ...;
//Variables to solve for
var StartTimes StartTime[Classes];
var Rooms Location[Classes];
var Lecturers Lecturer[Classes];
var Bool softConstraints[knownClasses];
//maximize sum(i in knownClasses) softConstraints[i] //use knownClasses
maximize sum(c in Classes) preferredTimes[StartTime[c]]
subject to {
  forall (k in knownClasses) {
    softConstraints[k] = (
      (KnownClass[k].st = StartTime[KnownClass[k].class])); //StartTime known
    };
```

```
  sum(i in knownClasses) softConstraints[i] = nrKnownClasses; //Use all KnownClasses
  // forall (k in knownClasses) {
  // softConstraints[k] = (
  // (KnownClass[k].lecturer = Lecturer[KnownClass[k].class]) //Lecturer known
  // & (KnownClass[k].room = Location[KnownClass[k].class]) //Room known
  // & (KnownClass[k].st = StartTime[KnownClass[k].class])); //StartTime known
  // };
  forall (f in fixedClasses) {
    (FixedClass[f].lecturer = Lecturer[FixedClass[f].class]); //Lecturer fixed
    (FixedClass[f].room = Location[FixedClass[f].class]); //Room fixed
    (FixedClass[f].st = StartTime[FixedClass[f].class]); //StartTime fixed
  };
  forall (m in Modules)
    forall (c in ClassesInModule[m])
      (Lecturer[c] in LecturersForModule[m]); //Lecturer can teach Module
  forall (c in Classes)
    (RoomSize[Location[c]] >= ClassSize[c]); //Room must be big enough
  forall (m in Modules)
    forall (ordered c1, c2 in ClassesInModule[m]) {
      (Lecturer[c1] = Lecturer[c2]); //Only one lecturer for each module
    };
  forall (CG in CourseGroups) {
    alldifferent(all (c in ClassesInGroup[CG])
      StartTime[c]); //Avoid Clashes amoung classes in same CourseGroup
    // 7 > sum(c in ClassesInGroup[CG]) (StartTime[c] < 9); //Max classes on Mon
    // 7 > sum(c in ClassesInGroup[CG]) ((StartTime[c] > 8)&(StartTime[c] < 18)); //M
    // 7 > sum(c in ClassesInGroup[CG]) ((StartTime[c] > 17)&(StartTime[c] < 27)); //
    // 7 > sum(c in ClassesInGroup[CG]) ((StartTime[c] > 26)&(StartTime[c] < 36)); //
    // 7 > sum(c in ClassesInGroup[CG]) (StartTime[c] > 35); //Max classes on Fri
  };
  forall (ordered c1, c2 in Classes) {
    StartTime[c1] = StartTime[c2] =>
      Location[c1] <> Location[c2]; //Avoid Room clashes
    StartTime[c1] = StartTime[c2] =>
      Lecturer[c1] <> Lecturer[c2]; //Avoid Lecturer clashes
  };
  // forall (l in Lecturers)
  // 19 > sum(c in Classes) (Lecturer[c] = l); //Max number of lectures given per wee
};
search {
  forall(c in Classes)
    once {
      tryall(i in StartTimes //Choose most preferred hours first
        ordered by decreasing preferredTimes[i])
          StartTime[c] = i;
      tryall(r in Rooms //Choose smallest adequate rooms first
        ordered by increasing RoomSize[r])
          Location[c] = r;
      tryall(l in Lecturers //Choose least assigned lecturers first
        ordered by increasing sum(c in Classes: bound(Lecturer[c])) (Lecturer[c] = l)
```

```
            Lecturer[c] = 1;
      };
    };
  display(c in Classes) <"Module",c,StartTime[c],
    "Day",dayNames[(StartTime[c]/dayLength)],
    "Hour",(StartTime[c] mod dayLength) + 9,
    "Room",Location[c],
    "Lecturer",Lecturer[c]>;
```

5.5 Keyed Data

Keyed Data for Second Term:

```
"//Data for CS5001 University Timetabling Model in OPL Studio",,,,,,,,,,
"//Constant values",,,,,,,,,,
"WeekLength",45,"//Number of working hours in each week",,,,,,,,
"DayLength",9,"//Number of working Hours in a day",,,,,,,,
"DayName","Mon","Tue","Wed","Thu","Fri","//Labels for Days in result",,,,,
"//Preference weighting for each working hour in the week","Day",900,1000,1100,1200,1
"PreferredTimes","Mon",2,10,9,6,1,8,7,5,5,
"PreferredTimes","Tue",3,10,9,6,1,8,7,5,5,
"PreferredTimes","Wed",3,10,9,6,1,8,7,5,5,
"PreferredTimes","Thur",3,10,9,6,1,8,7,5,5,
"PreferredTimes","Fri",3,10,9,6,1,8,7,5,5,
"//Groupings of Courses with their own timetables","Name","Courses:",,,,,,,,
"CoursesInGroup","CSYear1","CSUGrad1",,,,,,,,
"CoursesInGroup","CSYear2","CSUGrad2Eco",,,,,,,,
"CoursesInGroup","CSYear3","CSUGrad3Fin",,,,,,,,
"CoursesInGroup","CSYear4","CSUGrad4Fin","CSUGrad4Bus",,,,,,,
"CoursesInGroup","CSYear5","MScMob","MScNet","MScInt",,,,,,
"//Modules that make up each Course","Name","Modules on Course:",,,,,,,,
"ModulesInCourse","CSUGrad1","CS1100","CS1101","CS1102","CS1061","CS1063","CS1064","M
"ModulesInCourse","CSUGrad2Eco","CS2200","CS2202","CS2203","EC1204","EC2202","EC2207"
"ModulesInCourse","CSUGrad3Fin","CS3300","EC3406",,,,,,,
"ModulesInCourse","CSUGrad4Fin","CS4000","CS4405","CS4406","CS4408","CS4409","EC2206"
"ModulesInCourse","CSUGrad4Bus","CS4000","CS4405","CS4406","CS4408","CS4409","EC3101"
"ModulesInCourse","MScMob","CS563","CS5012","CS5013",,,,,,
"ModulesInCourse","MScNet","CS555","CS560","CS563","CS567",,,,,
"ModulesInCourse","MScInt","CS560","CS567","CS5201","CS5203","CS5205",,,,
"//Module details","Name","Number Lecturers in Week","nrTerms","nrStudents",,,,,,
"Module","CS1100",3,2,30,,,,,,
"Module","CS1101",2,2,30,,,,,,
"Module","CS1102",2,2,30,,,,,,
"Module","CS1061",2,2,30,,,,,,
"Module","CS1063",2,2,30,,,,,,
"Module","CS1064",2,2,30,,,,,,
"Module","CS2200",2,2,30,,,,,,
"Module","CS2202",2,2,30,,,,,,
```

```
"Module","CS2203",2,2,30,,,,,,
"Module","CS3300",1,2,30,,,,,,
"Module","CS4000",2,2,25,,,,,,
"Module","CS4405",1,2,25,,,,,,
"Module","CS4406",1,2,25,,,,,,
"Module","CS4408",2,2,25,,,,,,
"Module","CS4409",1,2,25,,,,,,
"Module","CS555",2,2,18,,,,,,
"Module","CS560",2,2,18,,,,,,
"Module","CS567",2,2,18,,,,,,
"Module","CS5201",2,2,6,,,,,,
"Module","CS563",2,1,12,,,,,,
"Module","CS5012",2,1,12,,,,,,
"Module","CS5013",2,1,12,,,,,,
"Module","CS5203",2,1,6,,,,,,
"Module","CS5205",2,1,6,,,,,,
"//Lecturers and what Modules they can teach","Name","Taught Courses:",,,,,,,,
"ModulesTaught","Augustus","CS3316","CS6002","CS6001","CS4405","CS4408","CS1064","CS1
"ModulesTaught","Tiberius","CS3314","CS4408","CS7200","CS1102",,,,,,
"ModulesTaught","Caligula","CS4409","CS3315","CS2205","CS5204","CS5205","CS5201","CS5
"ModulesTaught","Claudius","CS2200","CS5205","CS4150","CS5204","CS5202","CS3315",,,,
"ModulesTaught","Nero","CS4402","CS2021","CS6004","CS2203","CS1061","CS1063","CS6007"
"ModulesTaught","Galba","CS5013","CS5012","CS2204","CS3311","CS4031","CS4402",,,,
"ModulesTaught","Otho","CS6003","CS3310","CS3306","CS1063","CS2021","CS2024",,,,
"ModulesTaught","Vitellius","CS560","CS4000","CS6002","CS6007","CS3316","CS564","CS55
"ModulesTaught","Vespesian","CS2201","CS3313","CS1064","CS4150","CS1100","CS1102",,,,
"ModulesTaught","Titus","CS5201","CS5204","CS4409","CS3313","CS3315","CS2205",,,,
"ModulesTaught","Domitian","CS1100","CS4404","CS4001","CS2201","CS4150","CS3313",,,,
"ModulesTaught","Nerva","CS1102","CS1061","CS1063","CS4408","CS6006","CS3306","CS5201
"ModulesTaught","Trajan","CS1063","CS1061","CS3305","CS2021","CS2203","CS3316","CS403
"ModulesTaught","Hadrian","CS4054","CS7400","CS4404",,,,,,
"ModulesTaught","Antoninus","CS7200","CS7800","CS4405","CS7000","CS3312","CS3314",,,,
"ModulesTaught","Aurelius","CS565","CS6003","CS1061","CS5201","CS4000","CS3310","CS60
"ModulesTaught","Verus","CS1101","CS5202","CS5203","CS4001","CS3316","CS3315",,,,
"ModulesTaught","Cassius","CS4406","CS3314","CS1102","CS3312","CS4408","CS1101","CS44
"ModulesTaught","Commodus","CS5203","CS2024","CS4405","CS1061","CS4001","CS3315",,,,
"ModulesTaught","Pertinax","CS3315","CS5205","CS5201","CS5202","CS5203","CS6007",,,,
"ModulesTaught","Julianus","CS563",,,,,,,,
"ModulesTaught","Severus","CS1001","CS3312","CS3011","CS3012","CS6007","CS7000",,,,
"ModulesTaught","Caracalla","CS3313",,,,,,,,
"ModulesTaught","Geta","CS4040","CS2202","CS3012","CS1102","CS3314","CS2024",,,,
"ModulesTaught","Macrinus","CS563","CS5011","CS2204","CS6008","CS2203","CS5012","CS50
"ModulesTaught","Diadumenian","CS564","CS4403","CS3316","CS4031","CS2021","CS1101",,,,
"ModulesTaught","Elagabalus","CS5014","CS4402","CS3313","CS7100","CS4405","CS4404",,,,
"ModulesTaught","Alexander","CS2203","CS5202","CS6007","CS4150","CS6004","CS6001",,,,
"ModulesTaught","Thrax","CS3012","CS2021","CS555","CS2024","CS3011",,,,,
"ModulesTaught","Caesar","CS567","CS3300",,,,,,,
"//Venues for Lectures","Name","Size (Capacity)","Projector","PCs",,,,,,
"Room","PF1",9,0,0,,,,,,
"Room","G8",50,0,0,,,,,,
```

```
"Room","E1",20,1,1,,,,,,
"Room","GeoLT",40,1,0,,,,,,
"Room","W5",50,1,0,,,,,,
"Room","W6",40,1,0,,,,,,
"Room","AL8",20,0,0,,,,,,
"Room","CeG10",25,0,0,,,,,,
"Room","EeL2",30,1,0,,,,,,
"Room","G18",50,1,0,,,,,,
"Room","G19",35,1,0,,,,,,
"//Classes whose times are fixed (and cannot be changed)","Name","StartDay","StartTim
"FixedClass","EC1401a","Thur","eleven","Both","Eco201","unknown",,,,,
"FixedClass","EC1401b","Thur","twelve","Both","Eco201","unknown",,,,,
"FixedClass","MA1003a","Tue","eleven","Both","BooleLL2","unknown",,,,,
"FixedClass","MA1003b","Wed","eleven","Both","KaneG1","unknown",,,,,
"FixedClass","MA1015a","Mon","four","Both","BooleLL1","unknown",,,,,
"FixedClass","MA1015b","Wed","three","Second","KaneG19","unknown",,,,,
"FixedClass","MA1054a","Tue","nine","Both","FSBaL1","unknown",,,,,
"FixedClass","MA1054b","Wed","two","Both","W9","unknown",,,,,
"FixedClass","MA1054c","Thur","nine","Both","W5","unknown",,,,,
"FixedClass","EC2202a","Tue","ten","Both","AL10","unknown",,,,,
"FixedClass","EC2202b","Wed","three","Both","EeL2","unknown",,,,,
"FixedClass","EC2207a","Mon","four","Both","W6","unknown",,,,,
"FixedClass","EC2207b","Thur","twelve","Both","G2","unknown",,,,,
"FixedClass","EC1204a","Wed","four","Both","AL8","unknown",,,,,
"FixedClass","EC3406a","Thur","three","Second","BVG01","unknown",,,,,
"FixedClass","EC3406b","Thur","four","Second","BVG01","unknown",,,,,
"FixedClass","EC3406c","Thur","five","Second","BVG01","unknown",,,,,
"FixedClass","EC3406d","Fri","three","Second","BV11","unknown",,,,,
"FixedClass","EC3406e","Fri","four","Second","BV11","unknown",,,,,
"FixedClass","EC3406f","Fri","five","Second","BV11","unknown",,,,,
"FixedClass","EC4222a","Mon","eleven","Both","LL3","unknown",,,,,
"FixedClass","EC4222b","Mon","five","Both","FsA1","unknown",,,,,
"FixedClass","EC2206a","Thur","one","Both","ConS5","unknown",,,,,
"FixedClass","EC2206b","Thur","four","Both","W5","unknown",,,,,
"FixedClass","EC4403a","Thur","two","Both","W5","unknown",,,,,
"FixedClass","EC4403b","Thur","three","Both","W5","unknown",,,,,
"FixedClass","EC3104b","Tue","five","Second","LL3","unknown",,,,,
"FixedClass","EC3101a","Thur","one","Both","Ce110","unknown",,,,,
"FixedClass","EC3101b","Thur","five","Both","W9","unknown",,,,,
"//Classes whose times are known (but can be changed)","Name","StartDay","StartTime",
"KnownClass","CS567a","Tue","nine","Both","PF1","Caesar",,,,,
"KnownClass","CS567b","Wed","nine","Both","PF1","Caesar",,,,,
"KnownClass","CS5203a","Mon","eleven","Second","PF1","Commodus",,,,,
"KnownClass","CS5203b","Thur","one","Second","PF1","Commodus",,,,,
"KnownClass","CS5201a","Tue","twelve","Both","Ce6","Caligula",,,,,
"KnownClass","CS5201b","Thur","twelve","Both","Ce6","Caligula",,,,,
"KnownClass","CS5205a","Wed","one","Second","PF1","Pertinax",,,,,
"KnownClass","CS5205b","Thur","eleven","Second","PF1","Pertinax",,,,,
"KnownClass","CS5014a","Fri","eleven","Second","PF1","Elagabalus",,,,,
"KnownClass","CS5014b","Fri","twelve","Second","PF1","Elagabalus",,,,,
```

5.6 Derived Data

Derived Data for Second Term:

```
dayLength = 9;
weekLength = 45;
terms = {First, Second, Both};
nrFixedClasses = 29;
nrKnownClasses = 10;
dayNames = ["Mon","Tue","Wed","Thur","Fri"];
dayOfWeek = {
Mon,
Tue,
Wed,
Thur,
Fri
};
preferredTimes = [
2,10,9,6,1,8,7,5,5,
3,10,9,6,1,8,7,5,5,
3,10,9,6,1,8,7,5,5,
3,10,9,6,1,8,7,5,5,
3,10,9,6,1,8,7,5,5
];
CourseGroups = {
CSYear1,
CSYear2,
CSYear3,
CSYear4Fin,
CSYear4Bus,
CSYear5
};
Modules = {
CS1100,
CS1101,
CS1102,
CS1061,
CS1063,
CS1064,
CS2200,
CS2202,
CS2203,
CS3300,
CS4000,
CS4405,
CS4406,
CS4408,
CS4409,
CS555,
CS560,
```

```
CS567,
CS5201,
CS563,
CS5012,
CS5013,
CS5014,
CS5203,
CS5205,
EC1401,
MA1003,
MA1015,
MA1054,
EC2202,
EC2207,
EC1204,
EC3406,
EC4222,
EC2206,
EC4403,
EC3104,
EC3101
};
Classes = {
CS1100a,
CS1100b,
CS1100c,
CS1101a,
CS1101b,
CS1102a,
CS1102b,
CS1061a,
CS1061b,
CS1063a,
CS1063b,
CS1064a,
CS1064b,
MA1054a,
MA1054b,
MA1054c,
MA1003a,
MA1003b,
EC1401a,
EC1401b,
MA1015a,
MA1015b,
CS2200a,
CS2200b,
CS2202a,
CS2202b,
CS2203a,
```

```
CS2203b,
EC1204a,
EC2202a,
EC2202b,
EC2207a,
EC2207b,
CS3300a,
EC3406a,
EC3406b,
EC3406c,
EC3406d,
EC3406e,
EC3406f,
CS4000a,
CS4000b,
CS4405a,
CS4406a,
CS4408a,
CS4408b,
CS4409a,
EC2206a,
EC2206b,
EC4222a,
EC4222b,
EC4403a,
EC4403b,
EC3101a,
EC3101b,
EC3104a,
CS563a,
CS563b,
CS5012a,
CS5012b,
CS5013a,
CS5013b,
CS555a,
CS555b,
CS560a,
CS560b,
CS567a,
CS567b,
CS5201a,
CS5201b,
CS5203a,
CS5203b,
CS5205a,
CS5205b,
CS5014a,
CS5014b
};
```

```
Lecturers = {
Augustus,
Tiberius,
Caligula,
Claudius,
Nero,
Galba,
Otho,
Vitellius,
Vespesian,
Titus,
Domitian,
Nerva,
Trajan,
Antoninus,
Aurelius,
Verus,
Cassius,
Commodus,
Pertinax,
Julianus,
Geta,
Macrinus,
Diadumenian,
Elagabalus,
Alexander,
Thrax,
Caesar,
unknown1,
unknown2,
unknown3,
unknown4,
unknown5,
unknown6,
unknown7,
unknown8,
unknown9,
unknown10,
unknown11,
unknown12,
unknown13
};
Rooms = {
PF1,
G8,
E1,
GeoLT,
W5,
W6,
AL8,
```

```
CeG10,
EeL2,
G18,
G19,
Eco201,
BooleLL2,
KaneG1,
BooleLL1,
KaneG19,
FSBaL1,
W9,
AL10,
G2,
BVG01,
BV11,
LL3,
FsA1,
ConS5,
Ce110
};
ClassesInModule = #[
CS1100: {CS1100a,CS1100b,CS1100c},
CS1101: {CS1101a,CS1101b},
CS1102: {CS1102a,CS1102b},
CS1061: {CS1061a,CS1061b},
CS1063: {CS1063a,CS1063b},
CS1064: {CS1064a,CS1064b},
MA1054: {MA1054a,MA1054b,MA1054c},
MA1003: {MA1003a,MA1003b},
EC1401: {EC1401a,EC1401b},
MA1015: {MA1015a,MA1015b},
CS2200: {CS2200a,CS2200b},
CS2202: {CS2202a,CS2202b},
CS2203: {CS2203a,CS2203b},
EC1204: {EC1204a},
EC2202: {EC2202a,EC2202b},
EC2207: {EC2207a,EC2207b},
CS3300: {CS3300a},
EC3406: {EC3406a,EC3406b,EC3406c,EC3406d,EC3406e,EC3406f},
CS4000: {CS4000a,CS4000b},
CS4405: {CS4405a},
CS4406: {CS4406a},
CS4408: {CS4408a,CS4408b},
CS4409: {CS4409a},
EC2206: {EC2206a,EC2206b},
EC4222: {EC4222a,EC4222b},
EC4403: {EC4403a,EC4403b},
EC3101: {EC3101a,EC3101b},
EC3104: {EC3104a},
CS563: {CS563a,CS563b},
```

```
CS5012: {CS5012a,CS5012b},
CS5013: {CS5013a,CS5013b},
CS555: {CS555a,CS555b},
CS560: {CS560a,CS560b},
CS567: {CS567a,CS567b},
CS5201: {CS5201a,CS5201b},
CS5203: {CS5203a,CS5203b},
CS5205: {CS5205a,CS5205b},
CS5014: {CS5014a,CS5014b}
]#;
ClassesInGroup = #[
CSYear1: {CS1100a,CS1100b,CS1100c,CS1101a,CS1101b,CS1102a,CS1102b,CS1061a,CS1061b,CS1
CSYear2: {CS2200a,CS2200b,CS2202a,CS2202b,CS2203a,CS2203b,EC1204a,EC2202a,EC2202b,EC2
CSYear3: {CS3300a,EC3406a,EC3406b,EC3406c,EC3406d,EC3406e,EC3406f},
CSYear4Fin: {CS4000a,CS4000b,CS4405a,CS4406a,CS4408a,CS4408b,CS4409a,EC2206a,EC2206b,
CSYear4Bus: {CS4000a,CS4000b,CS4405a,CS4406a,CS4408a,CS4408b,CS4409a,EC3101a,EC3101b,
CSYear5: {CS563a,CS563b,CS5012a,CS5012b,CS5013a,CS5013b,CS555a,CS555b,CS560a,CS560b,C
]#;
ModulesTaught = #[
Augustus: {CS4405,CS4408,CS1064,CS1061},
Tiberius: {CS4408,CS1102},
Caligula: {CS4409,CS5205,CS5201,CS5203},
Claudius: {CS2200,CS5205},
Nero: {CS2203,CS1061,CS1063},
Galba: {CS5013,CS5012},
Otho: {CS1063},
Vitellius: {CS560,CS4000,CS555},
Vespesian: {CS1064,CS1100,CS1102},
Titus: {CS5201,CS4409},
Domitian: {CS1100},
Nerva: {CS1102,CS1061,CS1063,CS4408,CS5201},
Trajan: {CS1063,CS1061,CS2203},
Antoninus: {CS4405},
Aurelius: {CS1061,CS5201,CS4000},
Verus: {CS1101,CS5203},
Cassius: {CS4406,CS1102,CS4408,CS1101,CS4405},
Commodus: {CS5203,CS4405,CS1061},
Pertinax: {CS5205,CS5201,CS5203},
Julianus: {CS563},
Geta: {CS2202,CS1102},
Macrinus: {CS563,CS2203,CS5012,CS5013},
Diadumenian: {CS1101},
Elagabalus: {CS5014,CS4405},
Alexander: {CS2203},
Thrax: {CS555},
Caesar: {CS567,CS3300},
unknown1: {EC1401},
unknown2: {MA1003},
unknown3: {MA1015},
unknown4: {MA1054},
```

```
unknown5: {EC2202},
unknown6: {EC2207},
unknown7: {EC1204},
unknown8: {EC3406},
unknown9: {EC4222},
unknown10: {EC2206},
unknown11: {EC4403},
unknown12: {EC3104},
unknown13: {EC3101}
]#;
LecturersForModule = #[
CS4405: {Augustus,Antoninus,Cassius,Commodus,Elagabalus},
CS4408: {Augustus,Tiberius,Nerva,Cassius},
CS1064: {Augustus,Vespesian},
CS1061: {Augustus,Nero,Nerva,Trajan,Aurelius,Commodus},
CS1102: {Tiberius,Vespesian,Nerva,Cassius,Geta},
CS4409: {Caligula,Titus},
CS5205: {Caligula,Claudius,Pertinax},
CS5201: {Caligula,Titus,Nerva,Aurelius,Pertinax},
CS5203: {Caligula,Verus,Commodus,Pertinax},
CS2200: {Claudius},
CS2203: {Nero,Trajan,Macrinus,Alexander},
CS1063: {Nero,Otho,Nerva,Trajan},
CS5013: {Galba,Macrinus},
CS5012: {Galba,Macrinus},
CS560: {Vitellius},
CS4000: {Vitellius,Aurelius},
CS555: {Vitellius,Thrax},
CS1100: {Vespesian,Domitian},
CS1101: {Verus,Cassius,Diadumenian},
CS4406: {Cassius},
CS563: {Julianus,Macrinus},
CS2202: {Geta},
CS5014: {Elagabalus},
CS567: {Caesar},
CS3300: {Caesar},
EC1401: {unknown1},
MA1003: {unknown2},
MA1015: {unknown3},
MA1054: {unknown4},
EC2202: {unknown5},
EC2207: {unknown6},
EC1204: {unknown7},
EC3406: {unknown8},
EC4222: {unknown9},
EC2206: {unknown10},
EC4403: {unknown11},
EC3104: {unknown12},
EC3101: {unknown13}
]#;
```

```
RoomSize = #[
PF1: 20,
G8: 50,
E1: 20,
GeoLT: 40,
W5: 50,
W6: 40,
AL8: 20,
CeG10: 25,
EeL2: 30,
G18: 50,
G19: 35,
Eco201: 0,
BooleLL2: 0,
KaneG1: 0,
BooleLL1: 0,
KaneG19: 0,
FSBaL1: 0,
W9: 0,
AL10: 0,
G2: 0,
BVG01: 0,
BV11: 0,
LL3: 0,
FsA1: 0,
ConS5: 0,
Ce110: 0
]#;
ClassSize = #[
CS1100a: 30,
CS1100b: 30,
CS1100c: 30,
CS1101a: 30,
CS1101b: 30,
CS1102a: 30,
CS1102b: 30,
CS1061a: 30,
CS1061b: 30,
CS1063a: 30,
CS1063b: 30,
CS1064a: 30,
CS1064b: 30,
MA1054a: 0,
MA1054b: 0,
MA1054c: 0,
MA1003a: 0,
MA1003b: 0,
EC1401a: 0,
EC1401b: 0,
MA1015a: 0,
```

```
MA1015b: 0,
CS2200a: 30,
CS2200b: 30,
CS2202a: 30,
CS2202b: 30,
CS2203a: 30,
CS2203b: 30,
EC1204a: 0,
EC2202a: 0,
EC2202b: 0,
EC2207a: 0,
EC2207b: 0,
CS3300a: 30,
EC3406a: 0,
EC3406b: 0,
EC3406c: 0,
EC3406d: 0,
EC3406e: 0,
EC3406f: 0,
CS4000a: 25,
CS4000b: 25,
CS4405a: 25,
CS4406a: 25,
CS4408a: 25,
CS4408b: 25,
CS4409a: 25,
EC2206a: 0,
EC2206b: 0,
EC4222a: 0,
EC4222b: 0,
EC4403a: 0,
EC4403b: 0,
EC3101a: 0,
EC3101b: 0,
EC3104a: 0,
CS563a: 12,
CS563b: 12,
CS5012a: 12,
CS5012b: 12,
CS5013a: 12,
CS5013b: 12,
CS555a: 18,
CS555b: 18,
CS560a: 18,
CS560b: 18,
CS567a: 18,
CS567b: 18,
CS5201a: 6,
CS5201b: 6,
CS5203a: 6,
```

```
CS5203b: 6,
CS5205a: 6,
CS5205b: 6,
CS5014a: 18,
CS5014b: 18
]#;
FixedClass = [
< EC1401a,29,Both,Eco201,unknown1>,
< EC1401b,30,Both,Eco201,unknown1>,
< MA1003a,11,Both,BooleLL2,unknown2>,
< MA1003b,20,Both,KaneG1,unknown2>,
< MA1015a,7,Both,BooleLL1,unknown3>,
< MA1015b,24,Second,KaneG19,unknown3>,
< MA1054a,9,Both,FSBaL1,unknown4>,
< MA1054b,23,Both,W9,unknown4>,
< MA1054c,27,Both,W5,unknown4>,
< EC2202a,10,Both,AL10,unknown5>,
< EC2202b,24,Both,EeL2,unknown5>,
< EC2207a,7,Both,W6,unknown6>,
< EC2207b,30,Both,G2,unknown6>,
< EC1204a,25,Both,AL8,unknown7>,
< EC3406a,33,Second,BVG01,unknown8>,
< EC3406b,34,Second,BVG01,unknown8>,
< EC3406c,35,Second,BVG01,unknown8>,
< EC3406d,42,Second,BV11,unknown8>,
< EC3406e,43,Second,BV11,unknown8>,
< EC3406f,44,Second,BV11,unknown8>,
< EC4222a,2,Both,LL3,unknown9>,
< EC4222b,8,Both,FsA1,unknown9>,
< EC2206a,31,Both,ConS5,unknown10>,
< EC2206b,34,Both,W5,unknown10>,
< EC4403a,32,Both,W5,unknown11>,
< EC4403b,33,Both,W5,unknown11>,
< EC3104a,17,Second,LL3,unknown12>,
< EC3101a,31,Both,Ce110,unknown13>,
< EC3101b,35,Both,W9,unknown13>
];
KnownClass = [
< CS567a,9,Both,PF1,Caesar>,
< CS567b,18,Both,PF1,Caesar>,
< CS5203a,2,Second,PF1,Commodus>,
< CS5203b,31,Second,PF1,Commodus>,
< CS5201a,12,Both,CeG10,Caligula>,
< CS5201b,30,Both,CeG10,Caligula>,
< CS5205a,22,Second,PF1,Pertinax>,
< CS5205b,29,Second,PF1,Pertinax>,
< CS5014a,38,Second,PF1,Elagabalus>,
< CS5014b,39,Second,PF1,Elagabalus>
];
```

5.7 Results

Results for Second Term:

```
TimeTable,Slot,Day,HourOfDay,Module Lecturer Room
CSYear1,1,Mon,10:00,CS1100a Vespesian EeL2
CSYear1,10,Tue,10:00,CS1100b Vespesian EeL2
CSYear1,19,Wed,10:00,CS1100c Vespesian EeL2
CSYear1,28,Thur,10:00,CS1101a Verus EeL2
CSYear1,37,Fri,10:00,CS1101b Verus EeL2
CSYear1,2,Mon,11:00,CS1102a Tiberius EeL2
CSYear1,38,Fri,11:00,CS1102b Tiberius EeL2
CSYear1,5,Mon,14:00,CS1061a Augustus EeL2
CSYear1,14,Tue,14:00,CS1061b Augustus EeL2
CSYear1,32,Thur,14:00,CS1063a Nero EeL2
CSYear1,41,Fri,14:00,CS1063b Nero EeL2
CSYear1,6,Mon,15:00,CS1064a Augustus EeL2
CSYear1,15,Tue,15:00,CS1064b Augustus EeL2
CSYear1,9,Tue,9:00,MA1054a unknown4 FSBaL1
CSYear1,23,Wed,14:00,MA1054b unknown4 W9
CSYear1,27,Thur,9:00,MA1054c unknown4 W5
CSYear1,11,Tue,11:00,MA1003a unknown2 BooleLL2
CSYear1,20,Wed,11:00,MA1003b unknown2 KaneG1
CSYear1,29,Thur,11:00,EC1401a unknown1 Eco201
CSYear1,30,Thur,12:00,EC1401b unknown1 Eco201
CSYear1,7,Mon,16:00,MA1015a unknown3 BooleLL1
CSYear1,24,Wed,15:00,MA1015b unknown3 KaneG19
CSYear2,1,Mon,10:00,CS2200a Claudius G19
CSYear2,19,Wed,10:00,CS2200b Claudius G19
CSYear2,28,Thur,10:00,CS2202a Geta G19
CSYear2,37,Fri,10:00,CS2202b Geta G19
CSYear2,2,Mon,11:00,CS2203a Trajan G19
CSYear2,11,Tue,11:00,CS2203b Trajan EeL2
CSYear2,25,Wed,16:00,EC1204a unknown7 AL8
CSYear2,10,Tue,10:00,EC2202a unknown5 AL10
CSYear2,24,Wed,15:00,EC2202b unknown5 EeL2
CSYear2,7,Mon,16:00,EC2207a unknown6 W6
CSYear2,30,Thur,12:00,EC2207b unknown6 G2
CSYear3,1,Mon,10:00,CS3300a Caesar GeoLT
CSYear3,33,Thur,15:00,EC3406a unknown8 BVG01
CSYear3,34,Thur,16:00,EC3406b unknown8 BVG01
CSYear3,35,Thur,17:00,EC3406c unknown8 BVG01
CSYear3,42,Fri,15:00,EC3406d unknown8 BV11
CSYear3,43,Fri,16:00,EC3406e unknown8 BV11
CSYear3,44,Fri,17:00,EC3406f unknown8 BV11
CSYear4Fin,1,Mon,10:00,CS4000a Aurelius CeG10
CSYear4Fin,10,Tue,10:00,CS4000b Aurelius CeG10
CSYear4Fin,19,Wed,10:00,CS4405a Antoninus CeG10
CSYear4Fin,28,Thur,10:00,CS4406a Cassius CeG10
CSYear4Fin,37,Fri,10:00,CS4408a Nerva CeG10
```

```
CSYear4Fin,11,Tue,11:00,CS4408b Nerva CeG10
CSYear4Fin,20,Wed,11:00,CS4409a Titus CeG10
CSYear4Fin,31,Thur,13:00,EC2206a unknown10 ConS5
CSYear4Fin,34,Thur,16:00,EC2206b unknown10 W5
CSYear4Fin,2,Mon,11:00,EC4222a unknown9 LL3
CSYear4Fin,8,Mon,17:00,EC4222b unknown9 FsA1
CSYear4Fin,32,Thur,14:00,EC4403a unknown11 W5
CSYear4Fin,33,Thur,15:00,EC4403b unknown11 W5
CSYear4Bus,1,Mon,10:00,CS4000a Aurelius CeG10
CSYear4Bus,10,Tue,10:00,CS4000b Aurelius CeG10
CSYear4Bus,19,Wed,10:00,CS4405a Antoninus CeG10
CSYear4Bus,28,Thur,10:00,CS4406a Cassius CeG10
CSYear4Bus,37,Fri,10:00,CS4408a Nerva CeG10
CSYear4Bus,11,Tue,11:00,CS4408b Nerva CeG10
CSYear4Bus,20,Wed,11:00,CS4409a Titus CeG10
CSYear4Bus,31,Thur,13:00,EC3101a unknown13 Ce110
CSYear4Bus,35,Thur,17:00,EC3101b unknown13 W9
CSYear4Bus,17,Tue,17:00,EC3104a unknown12 LL3
CSYear4Bus,32,Thur,14:00,EC4403a unknown11 W5
CSYear4Bus,33,Thur,15:00,EC4403b unknown11 W5
CSYear5,1,Mon,10:00,CS563a Julianus PF1
CSYear5,10,Tue,10:00,CS563b Julianus PF1
CSYear5,19,Wed,10:00,CS5012a Galba PF1
CSYear5,28,Thur,10:00,CS5012b Galba PF1
CSYear5,37,Fri,10:00,CS5013a Macrinus PF1
CSYear5,11,Tue,11:00,CS5013b Macrinus PF1
CSYear5,20,Wed,11:00,CS555a Thrax PF1
CSYear5,5,Mon,14:00,CS555b Thrax PF1
CSYear5,14,Tue,14:00,CS560a Vitellius PF1
CSYear5,23,Wed,14:00,CS560b Vitellius PF1
CSYear5,9,Tue,9:00,CS567a Caesar PF1
CSYear5,18,Wed,9:00,CS567b Caesar PF1
CSYear5,12,Tue,12:00,CS5201a Caligula CeG10
CSYear5,30,Thur,12:00,CS5201b Caligula CeG10
CSYear5,2,Mon,11:00,CS5203a Commodus PF1
CSYear5,31,Thur,13:00,CS5203b Commodus PF1
CSYear5,22,Wed,13:00,CS5205a Pertinax PF1
CSYear5,29,Thur,11:00,CS5205b Pertinax PF1
CSYear5,38,Fri,11:00,CS5014a Elagabalus PF1
CSYear5,39,Fri,12:00,CS5014b Elagabalus PF1
```

5.8 Glossary

Class Timetable entry. A meeting of a group of students and a lecturer in a room at a specific time.

COBOL COmmon Business-Oriented Language. Third generation computer language. One of the oldest computer languages.

Constraint Something that serves to constrain; Restrictive condition.

Course Set of Modules attended by a group of students.

CourseGroup Set of courses which are available to students enrolled for a specific year.

DB2 Relational Database management system.

Eclipse Open Source IDE. Used in this project as an IDE for Java.

FixedClass Timetable entry which cannot be changed (e.g. Class supplied by another department).

IDE Integrated Development Environment. Set of tools which provide a programmer with all of the functionality needed to develop computer systems. Usually includes an Editor, Compiler and Debugger.

KnownClass Timetable entry which can be changed, if necessary to find an overall solution.

Lecturer Teacher of Classes.

Mainframe Very large computer.

Module A short course of study of a technical subject that together with other such completed courses can count towards a particular qualification.
Set of Classes about a subject.

Nomenclature Terminology used in a particular activity.

OPL Optimisation Programming Language. Modelling language for specifying optimisation problems.

OPL Studio IDE for OPL language.

Room Venue for Classes.

Bibliography

[Azevedo94] Francisco Azevedo and Pedro Barahona, Timetabling in Constraint Logic Programming (1994).

[Collins06] Collins English Dictionary, Complete and Unabridged (2006).

[Gueret96] Christelle Guèret, Narendra Jussien, Patrice Boizmault and Christian Prins, Building University Timetables using Constraint Logic Programming (1996).

[Gavanelli06] Marco Gavanelli, University Timetabling in ECL^iPS^e (2006).

[Hentenryck99] Pascal Van Hentenryck, The OPL Optimization Programming Language (1999).