CS5216 Projects

Projects are in groups of 3. The deliverables for the project are:

- Source code with comments
- Documentation of Predicates in ECLiPSe HTML format
- Overview report (maximum 10 pages) in MSWord or PDF

The above must be submitted electronically (in a zip file – you may add a readme.txt file to guide thought the zip content-) by email to steph@nus.edu.sg by March 31st, 17h00 latest. There will be no deadline extension.

Project presentations take place on Monday 14th April. Each group gives a 10 minutes presentation. The presentation consists of a short overview of the project outcome and of the answer to one question which will be emailed in advanced to the group by the lecturer marking the report between March 31st and April 14th.

The total mark for the project is 20. Each project topic consists of two parts: a core and possible extensions. The realization of the core yields 10 marks. Each group can then choose or define its own extension for the next 10 marks.

Project Topic 1: Regular Expressions
References:
“Introduction to Automata Theory, Languages, and Computation” by Hopcroft, Motwani, Ullman

Core:
Compile regular expressions into equivalent Definite Clause Grammars (DCG) that recognize the corresponding regular language.

For instance the regular expression “a*b*” can be compiled into:

```
l --> as, bs.
as --> [].
as --> [a], as.
bs --> [].
bs --> [b], bs.
```

However, there exists a direct mapping of Nondeterministic Finite State Automata (FSA) to DCG. Use this mapping and the mapping of regular expressions Nondeterministic FSA to compile regular expressions. The management of the dynamic code should be convenient. Package as a library.

Possible extensions:
Compilation into Deterministic FSA, emptiness, equivalence and minimization, Operations on regular languages and algebraic laws for regular expressions, Perl regular expressions, regular expressions constraints.

Project Topic 2: Minimum Spanning Tree
References:
“Introduction to Algorithms” by Cormen, Leiserson, Rivest, and Stein

Core:
Implement the two minimum spanning tree algorithms for undirected weighted graphs: Prim, and Kruskal. Consider alternative tree representations (the tree is a term; the tree is stored in the database). Package as a library (see existing graph libraries).

Possible extensions:
Reverse-delete and Boruvska, recent improvements, randomized algorithms, algorithms with online updates, application to clustering

Project Topic 3: Longest Common Subsequence
References:
“Introduction to Algorithms” by Cormen, Leiserson, Rivest, and Stein

Core:
Implement the Longest Common Subsequence algorithms: length of LCS, print LCS, print differences (unix Diff). Consider using a ‘C’/3 predicates as in DCG to make your code generic. Package as a library.

Possible extensions:
Edit distance, string alignment, string matching and approximate string matching.

Project Topic 4: Indexing
References:
“Database Management Systems” by Ramakrishnan and Gehrke

Core:
Implement a B+tree indexed database (insert, delete, update and retrieve). The tree and the data reside on disk in files (of fixed, parameterizable maximum size)

Possible extensions:
Use constraints in the B+tree, R-trees, O-tree (Constraint-Based Index Structure)

Project Topic 5: XML DOM in Prolog
References:
“An Introduction to XML and Web Technologies” by Moeller and Schwartzbach

Core:
Design and implement a Document Object Model for XML in Prolog. The API should include parsing (with error messages), testing well-formed-ness, basic navigation and serialization.

Possible extensions:
Advanced navigation and manipulation of trees, DTD validation, XPath evaluation, streaming and SAX.

Project Topic 6: Datalog
References:
“Logic Programming and Databases” by Ceri, Gottlob and Tanca

Core:
If we a database of facts involving only atoms (or ground terms), any logic program (not involving functors), called a Datalog program, can be evaluated.
For instance an ancestor program like the following never terminates in Prolog but can be evaluated using a strategy like Query/Subquery.

```
ancestor(X, Y) :- ancestor(X, Z), ancestor(Z, Y).
ancestor(X, Y) :- parent(X, Y).
```

Implement a Query/Subquery evaluation for Datalog queries in Prolog.

Possible extensions:
Magic sets, connection to a DBMS, integrity constraints in Datalog and their maintenance.

Project Topic 7: Streams
References:
“The Art of Computer Programming: Semi-numerical Algorithms” by Knuth

Core:
Implement the reservoir sampling algorithm for the random sampling of streams. Use ECLiPSe stream model. Package as a library.

Possible extensions:
Variants of reservoir sampling and various random sampling algorithms. Other stream processing. Combining streams (using constraints), other stream processing

Project Topic 8: Resource Constrained Shortest Path
References:

Core: The RCSP problem is just like the classic shortest path (SP) problem. However, each path is now associated with a cost (in addition to distance) and a valid path is one whose cost does not exceed a given bound.

While SP can be solved efficiently using dynamic programming, RCSP is NP-complete. This project is to implement RCSP using at least one optimization technique.

Possible extensions:

Project Topic 9: Task Scheduling
References:

Core: We are given a collection of tasks, each of which must be performed on one of a selection of machines. There is a list of precedences between the tasks which specify which tasks must be completed before a particular task can begin. The aim is to find a schedule, that is, a start time for each task so that the precedences are respected and no more than one task is being performed on any machine at any given moment.

Possible extensions: An extended version of this problem involves having machines of different types, and each task is limited to being performed on only certain types of machines.

Project Topic 10: Natural Language Processing
References:

Core:
Propose and write a NLP tool (text-to-phoneme, segmenter, morphological analyzer, part-of-speech tagger, parser) for the natural language of your choice. Suggested: (Thai, text-to-phoneme), (Malay/French/German,morphology),

Possible extensions:

Project Topic 11: Tree Layout
References:

Core: The problem is to display a binary tree, whose labels are short strings, with two criteria: the layout is aesthetically pleasing, and the space utilized is minimized.
More specifically: nodes on the same tree level should be aligned nodes on different levels should be spaced proportionately there is a minimum gap between adjacent nodes on the same level a parent node is spaced proportionately between its children the total space used for the layout is minimized.

The essential idea is to traverse the tree and construct the relevant constraints on the coordinates for each node.
Solving these constraints eventally produces values for the coordinates.

For example, consider the following tree described as a term:

```
node(node(node(node(nil, kangaroo, nil)
       marsupial,
       node(nil, monotreme, nil))),
   animal,
   node(node(node(nil, cockatoo, nil),
          parrot,
          node(nil, lorikeet, nil)),
       bird,
       node(nil,
          raptor,
          node(nil, eagle, nil))))
```

A vertical layout for this tree could be:

```
40                          animal
.
.
30              mammal                      bird
.
.
20         marsupial     mononteme       parrot         raptor
.
.
10    kangaroo    koala    platypus  cockatoo   lorikeet   eagle

01234567890123456789012345678901234567890123456789012345678901234567890123456789

         10        20        30        40        50        60
```

Your program should input a root coordinate, and be able to layout both horizontally and vertically.

Possible extensions: Consider also user preferences such as having certain nodes aligned vertically.

Project Topic 12: Examination Time Table
References:

Core: The problem consists of the following:

1. An examination session is made of a number of periods over a specified length of time i.e. examination session. Period lengths are provided.
2. A set of exams that are to be scheduled into a periods. Exam codes are not provided. As with all entities, competitors should assume sequential numbering beginning with 0.
3. A set of students enrolled on individual exams. Each student is enrolled on a number of exams. Students enrolled on an exam are considered to "take" that examination. For each exam, student numbers are provided.
4. A set of rooms with individual capacities are provided.
5. A number of Hard Constraints which must be satisfied
6. A number of Soft Constraints which contribute to a penalty if they are violated.
7. Details including a "weighting" of particular soft constraints.

Feasibility
A feasible timetable is one in which all examinations have been assigned to a period and room so that the following hard constraints are satisfied: Hard Constraints

* No student sits more than one examination at the same time;
* The capacity of individual rooms is not exceeded at any time
* throughout the examination session;
* Period Lengths are not violated;
* Satisfaction of period related hard constraints e.g. Exam_A After Exam_B;
* Satisfaction of room related hard constraints e.g. Exam_A must
* use Room 101.

Soft Constraints
A candidate timetable is penalised for each occurrence of the following soft constraints:

* Two exams in a row;
* Two exams in a day;
* Specified spread of examinations;
* Mixed duration of examinations within individual periods;
* Larger examinations appearing later in the timetable;
* Period related soft constraints;
* Room related soft constraints;

These can effectively be split into two groups i.e. those which are resource specific and those which can have a global setting.

Resource Specific Settings
The following constraints can be set for each period and each room:

* period related soft constraints;
* room related soft constraints;

This allows control of how resources would be used in constructing a solution. Values for these can be found after the introduction of periods and rooms in the datasets.

Global Settings

The following constraints can be set relative to each other:

* Two exams in a row;
* Two exams in a day;
* Specified spread of examinations;
* Mixed duration of examinations within individual periods;
* larger examinations appearing later in the timetable;

Possible extensions:

Project Topic 13: Post Enrolment Course Timetabling
References:

Core: The problem consists of the following: a set of events that are to be scheduled into 45 timeslots (5 days of 9 hours each); a set of rooms in which events take place; a set of students who attend the events; and a set of features satisfied by rooms and required by events. Each student attends a number of events and each room has a size.

In addition to this, some events can only be assigned to certain timeslots. Also, some events will be required to occur before other events in the timetable.

A feasible timetable is therefore one in where events have been assigned a timeslot and a room so that the following hard constraints are satisfied:

1. no student attends more than one event at the same time;
2. the room is big enough for all the attending students and satisfies all the features required by the event;
3. only one event is put into each room in any timeslot;
4. events are only assigned to timeslots that are pre-defined as available for those events;
5. where specified, events are scheduled to occur in the correct order in the week

You may not always be able to schedule all of the events into the timetable in the given time without breaking some of the hard constraints. It will therefore be necessary for entrants to not place some events in the timetable (or remove them later) in order to ensure that no hard constraints are being violated. These events will then be considered unplaced.

In addition, a candidate timetable is penalised equally for each occurrence of the following soft constraint violations:

* a student has a class in the last slot of the day;
* a student has more than two classes consecutively;
* a student has a single class on a day.

Possible extensions:

Project Topic 14: Curriculum Based Course Timetabling
References:

Core: The Curriculum-based timetabling problem consists in the weekly scheduling of the lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula published by the University and not based on enrolment data.

The problem consists of the following entities:
Days, Timeslots, and Periods
We are given a number of teaching days in the week (typically 5 or 6). Each day is split in a fixed number of timeslots, which is equal for all days. A period is a pair composed by a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.
Courses and Teachers
Each course consists of a fixed number of lectures to be scheduled in distinct periods, it is attended by given number of students, and is taught by a teacher. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.
Rooms
Each room has a capacity, expressed in terms of number of available seats. All rooms are equally suitable for all courses (if large enough).
Curricula
A curriculum is a group of courses such that any pair of courses in the group have students in common. Based on curricula, we have the conflicts between courses and other soft constraints.

The solution of the problem is an assignment of a period (day and timeslot) and a room to all lectures of each course.

The hard constraints are: all lectures of a course must be scheduled, and they must be assigned to distinct periods; two lectures cannot take place in the same room in the same period; lectures of courses in the same curriculum or taught by the same teacher must be all scheduled in different periods; if the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period.

Soft constraints are: for each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures; the lectures of each course must be spread into a minimum number of days; lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day; all lectures of a course should be given in the same room.

Possible extensions: