5DV037 Fundamentals of Computer Science Fall 2010

1 Course Staff

Instructor:	Name:	Stephen J. Hegner
	Office:	C444 MIT-huset
	Telephone:	090 786 5760
	E-mail:	`hegner@cs.umu.se`
	URL:	`http://www.cs.umu.se/~hegner`
	Office hours:	1015-1100 on days with a scheduled lecture
Assistant:	Name:	Tor Sterner-Johansson
	Office:	C420 MIT-huset
	E-mail:	`tors@cs.umu.se`
	Office hours:	Wednesday and Friday 1015-1200

- Stephen J. Hegner is available to answer questions about the lectures and the course material in general, but cannot answer detailed questions about what is or is not acceptable as an answer to an obligatory exercise.
- Questions regarding the obligatory exercises should be directed to Tor Sterner-Johansson.

2 Course Language

All lectures will be given in English. However, questions may be asked during the lectures in either English or Swedish, and written work may be submitted in either English or Swedish. The questions on final examination will be written in English; answers may be in either English or Swedish. For the final examination, it will be permitted to use an XX-English / English-XX dictionary, where XX is a natural language of the student's choice.

3 Course Literature

The official textbook for this offering of the course is the following.

- Peter Linz, *An Introduction to Formal Languages and Automata*, Fourth Edition, Jones and Bartlett, 2006; ISBN-10: 0-7637-3798-4; ISBN-13: 978–07637-3798-6.

In addition to the course text, there will be relatively detailed overhead slides. These materials will be available for download on the course Web page.

Version: February 04, 2011

4 Course Content and Outline

The official kursplan is available on this link. A more offering-specific outline is shown below.

The numbers shown in rectangular brackets (i.e., [..]) identify chapters and sections in the textbook. The items in rectangular brackets which begin with "X" indicate special topics not covered in the textbook. These topics are described in Section 5.2.

The numbers in angle brackets ⟨..⟩ indicate the approximate number of 45-minute lecture periods which will be devoted to the topic. This number is approximate, and in particular is rounded to the nearest integer. Adjustments will be made as the course progresses, and so the table below should not be used a definitive guide to which topics will be covered on which days.

I Introduction [1] ⟨2⟩

FA Finite Models of Computation

- FA.1 Finite Automata [2] ⟨2⟩
- FA.2 Regular Languages and Grammars [3] ⟨2⟩
- FA.3 Closure and Membership for Regular Languages [4] ⟨2⟩
- FA.4 Tools and Applications [XFA1-XFA3] ⟨1⟩

CF Stack-Based Models of Computation

- CF.1 Context-Free Languages and Grammars [5] ⟨2⟩
- CF.2 Simplification and Normalization [6] ⟨2⟩
- CF.3 Acceptors for Context-Free Languages [7] ⟨2⟩
- CF.4 Closure and Membership for Context-Free Languages [8] ⟨2⟩
- CF.5 Tools and Applications [XCF1-XCF4] ⟨2⟩

GC General Models of Computation

- GC.1 Turing Machines and Turing's Thesis [9,10.3] ⟨2⟩
- GC.2 The Programming Model of Computation [XCT1] ⟨1⟩
- GC.3 The Universal Computer [10.4] ⟨1⟩
- GC.4 The Classical Language Hierarchies [11] ⟨1⟩
- GC.5 The Limits of Algorithmic Computation [12,XCT2] ⟨2⟩

CC Computational Complexity [14] ⟨2⟩

R Review ⟨2⟩

5 Course Materials Outline

5.1 Textbook Topics Outline

The following is a list of those chapters and sections which will be covered in the course. For each chapter or section, a symbol is given which indicates the nature of coverage in the course. The meaning of these symbols is provided in the table below.

✔	Material will be covered in the course.
✘	Material will not be covered in the course.
✙	Review material, prior knowledge is expected.
✣	Material will be covered partially or selectively.

- In general, omitted items will not be covered. However, the possibility that some covered material may appear in an omitted chapter or section remains. In all cases, the lectures and course notes should be taken to be the definitive statement for the course material.

1 Introduction to the Theory of Computation

- 1.1 Mathematical Preliminaries and Notation ✙
- 1.2 Three Basic Concepts ✔
- 1.3 Some Applications ✣

2 Finite Automata

- 2.1 Deterministic Finite Acceptors ✔
- 2.2 Nondeterministic Finite Acceptors ✔
- 2.3 Equivalence of Deterministic and Nondeterministic Finite Acceptors ✔
- 2.4 Reduction of Number of States in Finite Automata ✘

3 Regular Languages and Regular Grammars

- 3.1 Regular Expressions ✔
- 3.2 Connection Between Regular Languages and Regular Expressions ✔
- 3.3 Regular Grammars ✔

4 Properties of Regular Languages

- 4.1 Closure Properties of Regular Languages ✔
- 4.2 Elementary Questions about Regular Languages ✔

- 4.3 Identifying Nonregular Languages ✔

5 Context-Free Languages

- 5.1 Context-Free Grammars ✔
- 5.2 Parsing and Ambiguity ✔
- 5.3 Context-Free Grammars and Programming Languages ✔

6 Simplification of Context-Free Languages and Normal Forms

- 6.1 Methods for Transforming Grammars ✔
- 6.2 Two Important Normal Forms ✔
- 6.3 A Membership Algorithm for Context-Free Grammars ✣

7 Pushdown Automata

- 7.1 Nondeterministic Pushdown Automata ✔
- 7.2 Pushdown Automata and Context-Free Languages ✣
- 7.3 Deterministic Pushdown Automata and Deterministic Context-Free Languages ✣
- 7.4 Grammars for Deterministic Context-Free Languages ✘

8 Properties of Context-Free Languages

- 8.1 Two Pumping Lemmas ✣
- 8.2 Closure Properties and Decision Algorithms for Context-Free Languages ✣

9 Turing Machines

- 9.1 The Standard Turing Machine ✔
- 9.2 Combining Turing Machines for Complicated Tasks ✣
- 9.3 Turing's Thesis ✔

10 Other Models of Turing Machines

- 10.1 Minor Variations on the Turing Machine Theme ✣
- 10.2 Turing Machines with More Complex Storage ✣
- 10.3 Nondeterministic Turing Machines ✔
- 10.4 A Universal Turing Machine ✔
- 10.5 Linear Bounded Automata ✘

11 A Hierarchy of Formal Languages and Automata

- 11.1 Recursive and Recursively Enumerable Languages ✔
- 11.2 Unrestricted Grammars ✣
- 11.3 Context-Sensitive Grammars and Languages ✣
- 11.4 The Chomsky Hierarchy ✔

12 Limits of Algorithmic Computation

- 12.1 Some Problems That Cannot Be Solved by Turing Machines ✔
- 12.2 Undecidable Problems for Recursively Enumerable Languages ✔
- 12.3 The Post Correspondence Problems ✣
- 12.4 Undecidable Problems for Context-Free Languages ✣
- 12.5 A Question of Efficiency ✣

13 Other Models of Computation

- 13.1 Recursive Functions ✘
- 13.2 Post Systems ✘
- 13.3 Rewriting Systems ✘

14 An Overview of Computational Complexity

- 14.1 Efficiency of Computation ✔
- 14.2 Turing Machine Models and Complexity ✔
- 14.3 Language Families and Complexity Classes ✔
- 14.4 The Complexity Classes P and NP ✔
- 14.5 Some NP Problems ✔
- 14.6 Polynomial-Time Reduction ✔
- 14.7 NP-Completeness and an Open Question ✔

5.2 Other Topics to be Covered

There are a number of topics which are not covered in the textbook but are related to the course material and are important for computer scientists to know. central to the course. These topics cover in particular, but not exclusively, important applications of the theory. They will be covered in the lectures and slides, and are listed below.

XFA1 Extended regular expressions in programming and systems: The regular expressions which are studied in the course are are proper subset of the "regexps" which are used in many programming environments. A brief comparison of the features of each will be made.

XFA2 Tools for parsing and using regular expressions: Regular languages and expressions are widely used, particular in the description and construction of scanners for programming languages. A brief overview of tools which have been developed for this purpose, such as *Lex* and *Flex* will be given. Unfortunately, due to a lack of any prerequisite for a knowledge of *C* and systems programming, no obligatory assignments using these tools will be possible.

XFA3 Modelling in computer systems using finite automata: The textbook considers finite automata only as acceptors of languages. However, such automata are widely used in more general ways to model aspects of computer systems. A brief overview of such modelling techniques will be given.

XCF1 Faithfulness to semantics – weak vs. strong representation: In many cases, the processing of languages involves assigning a meaning (semantics) to a string, and this requires that the underlying grammar be faithful to that semantics. An overview of what this entails will be given.

XCF2 Overcoming the limits of context-free languages: Although context-free languages are widely used in the modelling of both computer and natural languages, it is impossible to recapture such languages precisely using the context-free formalism. The ways in which this limitation is overcome will be discussed briefly.

XCF3 Parsing and understanding natural language: The representation and processing of natural (*i.e.*, human) languages poses problems not found with computer languages. A brief overview of how these issues are addressed will be given.

XCF4 Tools for parsing and using context-free languages: As context-free grammars are widely used for representing computer languages, tools for building parsers have been developed, *Yacc* and *Bison* being amongst the best known. A brief overview of what these tools can do will be given. Unfortunately, due to a lack of any prerequisite for a knowledge of *C* and systems programming, no obligatory assignments using these tools will be possible.

XCT1 The programming-language alternative to Turing Machines: While the classical model of Turing is still the most widely used representation for computability, it is not the most intuitive. For many purposes, a simple model based upon imperative *while* programs is more natural. This model will be presented briefly.

XCT2 Rice's Theorem for recursive languages: There is a simple and very general characterization of undecidable properties of languages which is unfortunately not covered in the textbook. Due to its simplicity and importance, it will be developed in the course notes and lectures.

5.3 Online Materials

There web site for the course is located at

http://www.cs.umu.se/kurser/5DV037/H10/index.html.

The following materials may be found on these pages.

1. This syllabus, in both PDF and HTML.
2. The lecture slides for the course.
3. Descriptions of the obligatory exercises.
4. Miscellaneous links to computation-related things.
5. Some official documents required by the Department of Computing Science.

6 Laboratory Schedule and Computer Resources

There is no official laboratory booking for the course, nor any in-laboratory instruction. In general, when not reserved by a course, the computer laboratories of the department are open for use by students for their coursework.

7 Course Schedule

The table below identifies the course meeting times and places, together with the nature of the meeting. The key "L" denotes a lecture, while "E" denotes an examination booking.

For each lecture, the topics to be covered are identified via the outline header number of Section 4 of this syllabus. So, for example, on September 09 the topics of FA.3 (closure and membership for regular languages) will be covered. This is only an approximate assignment of meeting times to topics, and it may be altered as the course progresses.

Rooms whose identifiers begin with the letter *M* are located in MIT-huset, while rooms whose identifiers begin with an *N* are located in Naturvetarhuset. The examination rooms (skrivsalar) are in the building known as Östra paviljongen.

TBA = to be announced; not yet known.

Week	Type	Date	Time	Room	Topics
35	L	Aug 30	0815-1000	MC313	I
35	L	Sep 02	0815-1000	MC313	FA.1
36	L	Sep 06	0815-1000	MC313	FA.2
36	L	Sep 09	0815-1000	MC313	FA.3
37	L	Sep 13	0815-1000	MC313	FA.4, CF.1
37	L	Sep 16	0815-1000	MC313	CF.1, CF.2
39	L	Sep 27	0815-1000	MC313	CF.2, CF.3
39	L	Sep 30	0815-1000	MC313	CF.3, CF.4
40	L	Oct 04	0815-1000	MC313	CF.4, CF.5
40	L	Oct 07	0815-1000	MC313	CF.5, GC.1
41	L	Oct 11	0815-1000	MC413	GC.1, GC.3
41	L	Oct 14	0815-1000	MC313	GC.5
42	L	Oct 18	0815-1000	MC313	GC.4 GC.2
42	L	Oct 21	0815-1000	MC313	CC
43	L	Oct 25	0815-1000	MC313	R
44	E	Nov 02	0900-1300	Skrivsal 3	Final examination
01	E	Jan 04	0900-1300	Skrivsal 7	Final examination
17	E	Apr 27	0900-1300	Skrivsal 1	Final examination

8 Prerequisites

The formal requirements are listed in the course plan, which may be found at the following link. They include the following.

1. A knowledge of discrete mathematics. This requirement is met by the course *Diskret matematik* (Discrete Mathematics, 5MA008). Formally, it is also met by the course *Algebra* (MATA96), although students who have read only the latter course are advised to study the introductory sections of the textbook particularly carefully to make sure that they grasp this material at the appropriate level.

2. An introductory course in programming at the university level.

This requirements should be met by students who are following a normal path of study in one of the programs of the Department of Computing Science. Students from other disciplines who are considering this course should understand that the level of sophistication required in these topics is relatively high, and the material is specifically oriented towards those who are pursuing extensive studies in computer science.

9 Grading System

This course has two parts (*moment* in Swedish), a conceptual part (*teoridelen* in Swedish and an exercise part (*laborationsmoment* in Swedish). However, unlike most other courses offered by the Department of Computing Science, separate grades are not reported to the administration for each of these parts. Rather, there is only one combined grade for these two parts, and both must be completed for the same offering of the course. Thus, it is not possible to complete the two parts of the course in different years. To obtain a passing grade for the course, the student must earn the grade of at least 3 on the conceptual part and the grade of S (satisfactory) on the exercise part.

The only possible grades for the exercise part are S (Satisfactory; G=*Godkänd* in Swedish) and U (Unsatisfactory, *Underkänd* in Swedish). The grade on this part will be determined entirely by five obligatory exercises. Each will be graded as S (Satisfactory) or U (Unsatisfactory). To obtain the grade of S for the exercise part of the course, it is both necessary and sufficient to obtain the grade of S on all five exercises. In addition, for each exercise it will be possible to earn a maximum of 40 quality points. These points will be assigned based upon the overall quality and correctness of the work. Thus, a maximum of 200 points may be earned on the five exercises. Quality points will be assigned only in the case that solutions are clearly expressed and in large part correct. They will not represent a detailed assessment of partial correctness.

The examination will have a total of 1000 points.

The final point total F for the course is computed as

$$F = \max(E, \ \ 0.8 \times E + L)$$

with E the number points earned on the examination, L the number of points earned on the obligatory exercises. The final grade on the conceptual part of the course is computed as follows.

Number of points	Grade
$F \geq 800$	5 (med beröm godkänd – excellent)
$650 \leq F < 800$	4 (icke utan beröm godkänd – very good)
$500 \leq F < 650$	3 (godkänd – satisfactory)
$F < 500$	U (underkänd – unsatisfactory)

In addition, to pass the course, a minimum of 500 points on the examination is necessary, regardless of how many points are earned on the exercises. Thus, exercise points can only be used to increase the grade from 3 to 4, or from 4 to 5. They cannot be used to rescue a performance of less than 50% on the examination.

10 Obligatory Work

10.1 General Remarks on the Obligatory Projects

- The exercises may be completed in groups, and collaboration is permitted on the software exercises, roughly as described in the documents *Riktlinjer vid labgenomförande (Policy for Obligatory Exercises)* and *Hederskodex (Honor Code)*. More details will be provided later, when the descriptions of these exercises are distributed.

- The exercises may be submitted individually, or two or three persons may submit one solution. However, once a solution is submitted, only those named on the submission will receive credit for it. Partners in solution may not be added after the initial submission.

- Each exercise will have a submission deadline. For late submissions, the number of points awarded will be $(1 - 0.2 \times p)$, where p is the number of working days which the submission is late. Thus, a submission which is more than one week late cannot receive any points at all.

- All obligatory exercises must be submitted for grading on or before the third and final examination (during April 2010). No exercises will be accepted after that date. After that date, the student must re-register for the next offering of the course and complete the exercise part according to the rules for that part.

10.2 Obligatory Work Completed in Previous Years

- Credit for individual exercises may not be carried over from previous years, nor may a grade of S on the exercises be carried forward from a previous year.