0.4pt0pt 0pt0.4pt
Department for Software and Computer Systems, KTH/ICT/SCS

Master level undergraduate course ID2213
LOGIC PROGRAMMING
Course Information 2017/18 (Period 1)

ID2213 *Logic Programming* is a 7,5-credit (hp) elective course on the advanced level for civil ingenjör programs. mainly CINTE (IT) and CDATE (D) (year 3), it is also in the curriculum for master programs (advanced level) as an elective course. It is offered by the department for Software and Computer Systems (SCS) within the ICT school of KTH.

Goal

The goal of this course is to acquaint you with the logic programming paradigm through the example of Prolog, an established and popular language for sequential programming based on the logic of Horn clauses, and some modifications and extensions of Prolog, but also with logic programming beyond Prolog, in order to widen your understanding of the potential and limitations of different logic programming paradigms.

According to the course plan, after the course the students will master the logic programming approach to developing software.

This goal is realized as follows:

- To formulate a problem as logical statements, facts and rules.
- To express algorithms as rules and give a formal logical semantics to logic programs.
- To design and/or choose appropriate data representations in a logic program
- To independently and creatively deal with different issues of translations between data representations.
- To use systematic refinement in software development.
- To be able to communicate ideas about logic program design in writing and orally.

After an approved course, the student is expected to be able to:

- explain the basic concepts in logic programming (program syntax for LP including DCG-notation for grammars, the equality theory for terms, the unification procedure
- implement algorithms over trees and lists in logic programming
- explain and use the logical and the most common non-logical control structures in Prolog
- discuss the stronger and weaker sides of logic programming and Prolog
- explain briefly whether and why a certain task is suitable (or not suitable) to be implemented using logic programming techniques

- implement smaller program prototypes as logic programs
- succesfully perform a smaller logic programming project in a group or as an individual within given time frames orally and in writing explain the meaning (the operational and the declarative semantics) of logic programs.

It is also desirable that the student can:

- explain the operational and declarative semantics for logic programs
- construct adequate representations of abstract data types, such as sets, sparse matrices, hash tables in logic programs
- use and explain difference structures (d-lists)
- use metaprogramming techniques in logic programming
- judge suggested improvements of the language design including its operational semantics.

Syllabus

- Horn clause logic and its usage in knowledge representation and reasoning.
- The pure Prolog programming language and its semantics.
- Real Prolog: facilities for arithmetic, structure inspection, metalogic; cut and negation.
- Programming techniques in Prolog: database programming, recursive programming, non-deterministic programming, incomplete datastructures, parsing with DCGs.
- Application examples: parsing, compilation, formula manipulation, expert systems.
- Modifications and extensions of Prolog.
- Logic programming "beyond" Prolog, systematic logic based structuring techniques (program specialisation, partial evaluation, abstract data types)

Prerequisites

Knowledge and experience of programming on the level of the mandatory courses of the programmes. It is desired that you have knowledge of logic on the level of either of the undergraduate courses
DD1350 *Logic for Computer Science*
or ID1015 *Logic for Computer Science.*
Formally there is no requirement of neither logic or mathematics, but in practice it is strongly advised that you have passed courses in programming, algebra and discrete mathematics such as
ID1018 *Programming I*
ID1020 *Algorithms and Data Structures*
SF1624 or IX1303 *Algebra and Geometry*
SF1610 or IX1500 *Discrete Mathematics*
A database course is also helpful as is some elementary knowledge of functional programming.

Follow-up or related courses

Undergraduate courses:
ID2202 *Compilers and Execution Environments*
ID2204 *Constraint Programming*
DD2454 *Semantics for Programming Languages*
DD2469 *Database Theory*

Tuition

The classroom tuition consists of classroom lectures. The number of meetings is adapted to the size of the group. After initial lectures of approximately one month the students start a project which is performed individually or in self assembled groups containing at most four individuals. Supervision is provided upon request, normally by email, or by personal meeting with the course leader, for instance in conjunction to the remaining lectures given simultaneously and where the more advanced material is presented together with some examples.

Requirements and course layout

The requirements are an approved examination on the reading course (TEN1; 4,5 hp) and one approved project assignment (LAB1; 3 hp).

For an approved grade on the exam for the course part (4,5hp) the student is expected to be able to:

- explain the basic concepts in logic programming (program syntax for LP including DCG-notation for grammars, the equality theory for terms, the unification procedure)
- implement algorithms over trees and lists in logic programming
- explain and use the logical and the most common non-logical control structures in Prolog

For higher grades the student is also expected to be able to:

- explain the operational and declarative semantics for logic programs including negation
- construct adequate representations of abstract data types, such as sets, sparse matrices, hash tables in logic programs
- use and explain difference structures (d-lists)
- use metaprogramming techniques in logic programming
- discuss the stronger and weaker sides of logic programming and Prolog (there are no clear "truths" here, it is the ability to reason clearly that is valued)

In the project course the student is expected to:

- formulate a problem oriented project that is suitable to illustrate the use of logic programming techniques
- execute the project using logic programming terminology (implementation of a program or analysis of a program/system)
- perform the project in a group or individually, within the time frame
- report through a short written report and an oral presentation in a "mini workshop".

Examination

The examination involves two parts: one (oral) exam for the reading course (TEN1, 4,5 hp) and a project work (LAB1, 3 hp).

The material necessary for passing the examination is covered in the lectures and practically backed up in the tutorials.

The examination consists of tiny programming exercises in Prolog and theoretical questions concerning Prolog, its various modifications and extensions.

The project course (3hp) is evaluated as Pass/Fail. If you submit a project report in time, this may give bonus points on the exam. You can normally get at most 16 bonus points added to those given for the exam. Note: the bonus points are valid only for the current academic year, and they are normally only counted if you have reached 25 points, i.e. a passed grade.

The tasks of the exam including bonus for the project work are worth 50 points.

The grades for the entire course are defined by total points being the sum of the exam points and the bonus points given on the assignments. At least 25 total points are required to pass the exam.

The exam for the reading course is evaluated with the grades A-F.

The grades for the number of total points n are as follows:

- n greater or equal to 45: A
- n from 40 until 44 : B
- n from 35 until 39 : C
- n from 30 until 34 : D
- n from 25 until 29 : E
- n from 15 until 24 : Fx
- below 15 : F

In case of the grade Fx, completing examination is possible within three month after the original exam. In that case, the course leader will on demand offer an extra home assignment to be solved by the student within one week. Please note that completing is only allowed if the student reached the limit for Fx without counting bonus points. If the student achieves less than 15 without bonus points, the bonus points are not counted, but the degree F is given.

This is to ensure that students do not take advantage of work done by other group members in order to pass the course.

The "bonus points" approved on the examination based on the evaluation of the project task, are transferrable to the extra exams.

The project work consists of a practical programming task or possibly a report on some semantical aspects of logic programming carried out by students alone or in teams of at most four students. The student(s) should propose a task covering about two full time weeks of work. The task should highlight some aspect of logic programming of interest to the student(s). A set of older project assignments are provided on the course web to serve as inspiration. The project work is graded on the scale Fail-Pass.

Ambitious project work reported *before* the examination will lead to "bonus" points on the exam as indicated above. Oral reporting of projects is done through a common seminar where students present their projects to each other, including a demonstration and an oral presentation of the project results. If deemed appropriate by the course leader reporting can also be done by appointment or other arrangement.

In order to get a favorable grading on the project report it must be submitted in conjunction to the scheduled oral presentation before the ordinary examination. Checking and grading of project reports is guaranteed only in connection to some exam, that is within three weeks of the examination period (before or after).

During the oral presentation, all participants in a team should be able to answer questions concerning the whole project.

If you have a problem finding out a good project task, I suggest that you pick a topic from the textbook, and then solve some of the more advanced exercises for that topic (mostly those not

listed here). With a nice and concise report and presentation this makes a perfect task for the project part of the course.

Completion of the course yields 7,5 hp. The grade for the course as a whole is determined by the grade on the examination for the reading course taking also bonus points into consideration.

Literature

Compulsory literature

- Leon Sterling and Ehud Y. Shapiro. *The Art of Prolog: Advanced Programming Techniques*. 2nd ed. The MIT Press, Cambridge, MA, 1994. Available from Teknologbutiken, floor 5.

  The course book. A Prolog programming textbook with lots of carefully commented program examples. Theory is somewhat tersely covered (but enough for this course). Some think the book is a bit verbose. Others appreciate the level of detailed explanation.

Alternative, Complementary non-compulsory literature

- I. Bratko. *Logic Programming and Artificial Intelligence*. 3rd ed. Addison Wesley, 1999, A 4th edition 2011 is available, ISBN 9780321417466.

  A course book also used in courses on Artificial Intelligence. The prolog part is well thought out with a lot of useful examples. This first part of the book is what is covered in this course.

- Ulf Nilsson and Jan Małuszynski. *Logic, Programming and Prolog*. 2nd ed. J. Wiley and Sons, Chichester, 1995.

  Alternatively students can use this earlier course book, a textbook in standard (sequential) Prolog that also comprises a short introduction into modifications and extensions of Prolog such as AND- and OR-parallel modifications of Prolog and extensions of Prolog with equation and constraint solving. The book is available for cost free download on the web and should for some areas be complemented with examples from other sources.

- William F. Clocksin. *Clause and Effect - Practical Programming for the Working Programmer*. Springer-Verlag, Berlin, 1997.

  A book containing many useful logic programming techniques introduced through examples. Two of the presented case studies are taken from this book.

- *SICStus Prolog User's Manual*. The most recent version is provided on line on the web page `http://www.sics.se/sicstus/`.

- Paul Brna. *Prolog Programming - A First Course*. A compendium introducing Prolog.

- Reports and some code from some older project assignments are available on the web or through the course leader.

- Earlier exams and solutions are also available on the web.

Extra literature on Prolog for those especially interested

- John W. Lloyd. *Foundations of Logic Programming*. 2nd ed. Springer-Verlag, Berlin, 1987.

  An accessible presentation of the logical foundations of Prolog.

- Krzysztof R. Apt. Logic Programming. Ch. 10 in J. van Leeuwen, ed., vol. B of *Handbook of Theoretical Computer Science*, pp. 493–574. Elsevier Science Publishers, Amsterdam, 1990.

  A handbook exposition of the logical foundations of Prolog.

- J. Alan Robinson. Logic and Logic Programming. *Commun. ACM*, 1992, 35(2):43–51, 56–66.

  Some history of computational logic and a nice popular exposition of various calculi and strategies of resolution, written by the father of the resolution method.

Useful Online Resources

The KTH software distribution `https://intra.kth.se/it/programvara/download/sicstus`.

SICStus Prolog homepage: `http://www.sics.se/sicstus/`.

The course home page : `https://www.kth.se/social/course/ID2213/`.

Schedule

Classroom Tuition

Lectures: Weekly two-hour lectures.

Project presentations: One session in class where students present projects.

A tentative plan of the lectures (Case studies will be presented in class if time permits):

Content	Ch. [Brna][Cl]	Ch. [Ste&Sha]	Ch. [Ma&Ni]	Ch. [Bratko]	SICStus Manual
L1 Aug 28th 13-15					
Theory of logic programming. Unification. Database programs with relations over constants (Datalog). Least Herbrand models. Proof trees and search trees. Operational semantics: SLD-resolution.	1, 2, 3	1, 2, 4, 5 (not 2.4, 2.5, 4.3, 5.2, 5.3, 5.5, 5.6)	1, 2, 3, 6	1.1, 1.2, 1.3, 1.4, 1.5	4.1, 4.6
L2 Sep 4th 13-15					
Recursive programming style and techniques.	4, (5), 6, 10	3, 4, 5, 6, 7, 8, 13 (not 3.6, 4.3, 5.2, 5.3, 5.5, 5.6, 6.3, 7.3, 7.6, 13.3, 13.4)	1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 2.7, 4.1, 4.2, 4.3, 4.4, 4.6	1.3, 3.1, 3.2, 3.3, 3.4, 4.1, 4.2, 9.1, 9.2, 9.3	8.2, 13.1 (parts), 13.6, 16
L3 Sep 11th 13-15					
Accumulating parameters and differential structures. Abstract data types in LP.	12	7, 8, 13.3, (not 13.4), 15	7.3	(10.1, 10.2 not taught)	14.6.1, 13.6.2, 13.9.3, 17, 19, 22, 23
L4 Sep 18th 13-15					
Search and control. Cut and negation in Prolog. Concurrent and parallel logic programming models. State space programming. Puzzle-solving, game-playing.	7, 9	5.5, 11, 14, 20, 21 (not 11.7, 14.4)	4, 5, 11, 12, A.3	2.5, 2.6.1, 4.3, 4.4, 5.1, 5.2, 5.3, 5.4, 9.5, 22	4.5, 8.4, 13.1, 13.2, 13.3
L5 Sep 29th 13-15					
Grammars. Parsing with DCGs. Translating DGCs into definite clauses.	8, 11	19, 24	10	21	13.9.8, 41
L6 Oct 6th 13-15					
Program transformation techniques. Partial evaluation. Metaprogramming. Structure inspection, metalogical primitives. Expert systems.	12	10, 13, 16, 17 (not 17.5), 18, 22	8, 9	7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 15, 16, 23.1, 23.2	8.7, 8.9, 8.12, 13.7, 13.8, 13.9.5, 13.9.6, 13.9.7
L7 Oct 10th 10-12					
Case study: Compiler. Case study: Reasoning about electronic circuits. Examples from older exams	C9 C7, C8	24			
Sem Oct 13th 13-17					
Presentation of Projects.					
Ten Oct 24th 14-19					
Written Examination					

Tips on reading Sterling & Shapiro's book

This is not a course where you are expected to learn the content by heart but rather you show your understanding of the material through your ability to solve problems.

Recommended exercises from Sterling and Shapiro "The Art of Prolog 2 ed" are listed below organised according to the lecture structure (more advanced exercises not listed, or as (—) can be used for projects).

The textbook contains exercises for most chapters. Below you find a selection of them that can be useful. The other exercises are not irrelevant, but in most cases I consider them to be more complicated.

L1 - Lecture 1

For L1 I suggest reading chapters 1,2,4,5 You can safely skip 2.4, 2.5, 4.3, 5.2, 5.3, 5.5 and 5.6

These excercises are relevant:
ch 1 - What is the essence of logic programming?
(computation, meaning of a program)
2.1.1 (i) (ii) (iii)
2.2.1 (i) (ii) (iii)
2.3.1 (i) (ii)
4.1.1 (i) (ii)
4.2.1 (i) (ii) (iii)
5.4.1 (i) (ii)

L2 - Lecture 2

For L2 I suggest reading chapters 3,4,5,6,7,8 and 13 You can skip 3.6, 4.3, 5.2, 5.3, 5.5, 5.6, 6.3, 7.3, 7.6, 13.3 and 13.4

These excercises are relevant:
3.1.1 (i) (iv) (v) (viii)
3.2.1 (ii) (iii) (v)
3.3.1 (i) (ii) (iii) (iv) (v)
3.4.1 (i) (ii) (iii) (iv) (v)
3.5.1 (i) (iii) (iv) (v)
6.1.1 (i) (ii) (iii)
7.1.1 (i) (ii)
7.2.1 (i) (ii)
7.3.1 (i) (ii)
7.5.1 (i)
8.2.1 (i) (ii) (iii)
8.3.1 (i) (ii) (iii) (iv) (v) (vi) (vii) (viii)

L3 - Lecture 3

For L3 I suggest reading chapters 7,8 and 15. You should also read 13.3 here, but 13.4 can be skipped The textbook uses the infix operator '́to indicate difference lists. My examples use '-'. As you know the choice of operator is irrelevant. A good idea is to use an operator that has no other meaning in your data. Otherwise programs might be quite unreadable.

These excercises are relevant
13.3 (i) (ii)
15.1 (i) (ii) (iii)
15.2 (i) (ii) (iii) (iv)

L4 - Lecture 4

For L4 I suggest reading chapters 5.5, 11, 14, 20 and 21 You can skip 11.7 and 14.4

These excercises are relevant:
11.1 (i) (ii) (iii)
11.3 (—)
11.4 (ii)
14.1 (i) (ii) (v) (vi)
14.2 (ii)
20.1 (ii) (iii)

L5 - Lecture 5

For L5 I suggest reading chapters 19 and 24.

These excercises are relevant:
ch 19 (—)
ch 24 (—)

L6 - Lecture 6

For L6 I consider a few examples taken from other textbooks. There is somewhat corresponding text in S&S in in chap 17.1. For meta-programming I suggest reading chapters 9,10,13,16,17, 18 and 22. You can skip 17.5 ch 10 covers basic predefined predicates for metaprogramming. ch 17.2 -17.4 are related to programming techniques with meta-interpreters. Chap 22 describes an expert system witten in Prolog. The techniques of extending a meta-interpreter for "vanilla Prolog" to form an expert system are found in the older textbook N&M. (copies are on the course web).

These excercises are relevant:
16.1 (i)
16.2 (—)
16.3 (—)
17.2 (i) (ii) (iii) (iv)
17.4 (—)
18.1 18.2 18.6 (—)
ch22 (—)

L7 - Lecture 7

For L7 I suggest reading chapter 24 and of course the separately distributed copies from Clocksin's book.

For lecture 7 tasks from Clocksin are relevant.

Registration for the course edition and the exams

In order to be allowed to participate in the tuition, take an exam, defend project work in ID2213, you must have elected the course via the study councellor (Studievägledningen) of your programme. You must also register yourself with the education administration for the autumn semester. We ask you to do so even if you attended some earlier edition of the course, did not complete the examination and wish to continue this year. The registration on the course is done by sending an e-mail to the course leader.

In order to be admitted to any of the scheduled extra exams in ID2213, you have to register yourself by sending an e-mail to the course leader. The deadline is two weeks before the exam. If fewer than five students wish to take the extra exam, oral examination will be offered.

Exams

Exam (1st): October 2017 (always given, bonus points count).

Re-exam: December 2017 (mandatory registration before dec 1st, converted to oral exam if fewer than five register).

Computer work

The implementation of Prolog that you are normally supposed to work with is SICStus Prolog. This is a commercial product from SICS freely available for your educational use via the KTH software distibution system. If you have access to another prolog system you can feel free to use that instead.

The project assignments related files are located on the course web page.

Teacher

- Thomas Sjöland, lecturer — tutorials, project supervision, administration
  Phone: 08-790 4113

Important Addresses

Department for Software and Computer Systems (SCS)

Postal address:
Programvaruteknik och datorsystem, KTH/ICT/SCS, Kistagången 16, 164 40 KISTA.

Visiting address:
Electrum, plan 4, elevator A, KTH/ICT/SCS (Kistagången 16).

Contact:

Email for correspondance relating to the course: `mailto:sjoland@kth.se`.
Web page of this year's edition of the course: `https://www.kth.se/social/course/ID2213/`.

Studentexpedition in Electrum floor 3

Web page: `http://www.ict.kth.se/`