Teaching Formal Methods for Computational Linguistics at Uppsala University

Roussanka Loukanova
Computational Linguistics
Dept. of Linguistics and Philology, Uppsala University
P.O. Box 635, 751 26 Uppsala, Sweden
rloukano@stp.lingfil.uu.se

Abstract

In the first sections of this paper, I discuss some of the motivation for teaching Formal Methods for interdisciplinary programs, such as Computational Linguistics, that use computational apparatus. In the second section, I share my experience by teaching such courses in the program for Computational Linguistics and Language Technology at Uppsala University. In the last section, I will present existing gaps in education on Formal Methods in interdisciplinary and science subjects.

Keywords: computability, automata, logic, computational linguistics, formal grammar, computational grammar, computational semantics, interdisciplinary program

1. INTRODUCTION

Contemporary work in many, if not all sciences, inevitably requires interdisciplinary cooperation, especially by involving expertise on various branches of mathematics including pure numerical and symbolic computing and other formal methods.

Computational Linguistics and Natural Language Processing are among the interdisciplinary subjects that depend heavily on theoretical results from both quantitative and symbolic mathematics. On the side of the quantitative math, Statistics and Probability methods have a special status for applications in Natural Language Processing. Most of the other subfields of Computational Linguistics originate from Logic and Theory of Computability and use various formal methods. Major studies in Linguistics have evolved into modern computational subjects such as Computational Syntax, Computational Semantics, Computational Morphology, etc. that have various applications for software development and computerised Natural Language Processing.

The first decisive step for introducing formal methods in the study of syntax was done by Noam Chomsky. In the second half of the 50s, he published mathematical results that established¹ the study of so called *Chomsky's Hierarchy of Formal Grammars and Languages* in the now classical field *Theory of Formal Languages*. This field has been developed, by logicians and computer scientists, into a vast subject in theoretical computer science, which also has many applications in Natural Language Processing. Chomsky's book *Syntactic Structures*, Chomsky 1957, introduced formal syntax for English. That work initiated continuously emerging and developing of elaborate formalisms for syntax and computational grammar of human languages.

On the side of semantics of human languages, Richard Montague (see [10]) introduced a version of Higher Order Intensional Logic and gave a formal analysis of a small fragment of English. From then, it has been recognised that the semantics of human languages can be studied in a

¹Logicians had been working in the field of automata and algebraic structures of formal languages before Chomsky, but predominantly, their work was considered, by logicians and linguists, as inapplicable to the studies of human languages.

scientifically rigorous way with computational apparatus that is one or other variant of higher order logic.

The field of Automata Theory is essential for implementations of context-free parsing algorithms specialized for natural language. Finite State Automata have proved to be highly efficient formal technique for building automatic morphological analysers.

These and other research developments have already significant and very successful applications in mundane and advanced technologies involving natural language processing, from spell-checkers and grammar checking, to automatic speech recognition and machine translation.

The nature of the field of Computational Linguistics is innately interdisciplinary. The object of study is human language. The methods and techniques are amalgamations of methods and techniques developed in different fields of mathematics (both numerical or symbolic), research in theoretical linguistics, and by using different techniques of computer science, including programming. Simultaneously with Computational Linguistics, other interdisciplinary subjects have been emerging, e.g. Artificial Intelligence, Cognitive Science, Neuroscience of Language and Bioinformatics, which also rely on formal methods in addition to pure mathematics.

Advances in such interdisciplinary research, technology and industry open job market for specialists with appropriate education, including such in formal methods. To meet the needs of research and job markets for specialists, nowadays, many universities have established complete educational programs in Computational Linguistics or Cognitive Science. Other Universities offer individual courses, either in some specialized field of Computational Linguistics, or broad coverage courses in Natural Language Processing. By being interdisciplinary, these programs or individual courses can be located in Departments of Linguistics, Computer Science, or in Mathematics, and sometimes are joint educational programs. The major courses that students are required to take vary depending on the affiliation of the program to these departments.

2. FORMAL METHODS IN COMPUTATIONAL LINGUISTICS

2.1. Formal Methods in Computational Linguistics in Uppsala

In this section I will describe the courses on formal methods in the program for Computational Linguistics and Language Technology at Uppsala University. However, these courses are typical for many others such programs, including in Cognitive Science and Artificial Intelligence.

The number of the students in the program (as in general in Computer Science and Information Technology in Sweden) has dropped down significantly in the last years. This has lead to limited financial resources and the need to decrease the number of lectures per course. Typically, each course consists of 10 lecture units (2x45 min) and 5-6 computer labs or exercise units (2x45). Despite of this, the program has a good set of foundational courses in formal methods and specialized courses in computational methods, programming and software development for linguistics.

Students usually understand that for their future professional well-being they need specialized courses. On the contrary, it seems that there is a world-wide general tendency of misunderstanding, at least among majority of students, that formal methods are abstract theoretical matter which were not of use in real-life practice and professional skills. As a motivation of all foundational courses, I use the argument that the specialized courses in computational subjects, including quality programing, are heavily dependent on abilities to work with formal and rigorous specifications and symbolic manipulations. The methods for such work are provided by the foundational courses in formal methods.

Handling students with insufficient mathematical background is a significant problem, especially in the context of short course terms of 5 weeks. Typically, each class group consists of students with varying mathematical background. This puts challenges for the course level and pace in the presence of students with limited mathematics backgrounds, while in the same time maintaining

sufficient interest for the more advanced students. My general approach to students with lower math background than required for the course is to provide them with additional consultation during office time and with supplementary literature for filling up the missing knowledge. Another useful way is to encourage work on homework and lab assignments in small groups of two to three students, where they help each other in understanding and solving problems.

2.2. Some Courses on Formal Methods and Dependent Specialized Courses

In what follows, I will introduce the central goals of some of the courses in Formal Methods for Computational Linguistics, which I have experience with. Alongside, I will describe my techniques for teaching and point to software tools that I have been using in teaching logic, computational grammar, semantics and morphology. Most of the lectures are accompanied by computer labs for work with software that students use in assignments and project work.

Foundations of Computer Science I (Introductory Logic)

The course is an elementary introduction to the syntax and semantics of First Order Language. The emphasis is on understanding the relationship between syntax and semantics of a language, the similarity and distinction between a formal language and natural language. This is the very first introduction to finite descriptions of infinite languages via recursive definitions and Frege's Principle of Compositionality for the semantics of First Order Logic.

In my practice with major and non-major students, the syntax and semantics of first order language is easily understood by using the software package LPL by Barwise and Etchemendy 1999 (see [1]). Often, when a Computational Linguistics course is located in a humanity department, students come into the program with limited background in mathematics and difficulties in understanding and carrying on formal proofs without extra care. Having in mind such students, the textbook gives various concepts and proofs in two parallel modes: informal and formal, which greatly facilitates learning the formal definitions and proof techniques.

For more details, see the course website [6].

Foundations of Computer Science II (Formal Languages and Automata Theory)

The goals of the course includes introductory understanding of some fundamentals of formal models of computation, in particular, techniques for finite representation of infinite objects by recursive definitions, formal grammars and automata. In a term of 5 weeks, it is an introduction mainly to naive set theory, finite state automata, context-free grammars and push-down automata, brief introduction to context-sensitive languages and Turing Machines, and an emphasis on the intuitions behind the various limitations to what different types of automata can compute.

Typically, this course is rather challenging for most students since it is their first encounter with the methods of symbolic computations. This is why motivation is very important. I have been presenting arguments such as: it is very important one to know what problems can be solved algorithmically and what can not, and if a task is computable, what algorithm can be used, and how effective and efficient an algorithm can be.

The course provides background for other specialized courses such as computational syntax, algorithms for parsing, applied automata for morphological analysis, etc.

For more details, see the course website [7].

Applied Automata Theory

The course is a continuation of the sequence of the courses in *Foundations of Computer Science* and most of the specialized courses in Computational Linguistics. This is why it is scheduled in the last year of the Master's program.

The goals of the course are in two parallel directions: (1) acquiring knowledge on some of the mathematical theory of abstract automata and, in parallel, (2) practice with software for operations with formal languages and automata, and applications of automata to automatic processing of natural language.

The computer lab work is mainly on the applications of the Finite State technology, which includes practice with Finite-State software developments and implementations of algorithms for the Finite State classes of languages and automata: closure properties, determinization, minimization, decision properties, regular languages and expressions, algorithms for conversions between Finite State Automata and regular expressions, combinations of Finite State Automata with other systems. With lab assignments and course projects, students also experiment with processing natural language by using finite state tools: tokenization, morphological analyzers and generators, tagging, e.g. part of speech disambiguation, shallow parsing. For this purpose, in the computer labs, we use the Xerox Finite-State tools (see [2]).

Computational Grammar

The course introduces the concepts of formal and computational grammar of natural language. The formal description of grammar theory covering syntax, semantics and lexicon (see [11], uses the feature-value language and logic with the fundamental operator of unification. Linguistic information is represented from the perspective of

- linguistic generality and adequateness
- computational efficiency
- syntax-semantics relations
- computerized processing of natural language

In this course, students experience the direct link between formal methods and software development for large scale grammars. In the computer labs, we use the LKB grammar tool for writing small grammars of natural language, see [5]. Students practice advanced applications of formal methods for grammar software development, chart parsing with feature-value descriptions, syntactic and semantic ambiguity and syntax-semantics interface.

For more details, see the course website [9].

Computational Semantics I

The goal of the course is to introduce the central concepts of language, information and math logic, and the relations between them via acquiring knowledge on

- formal semantics: in particular, by learning some of the methods of model theoretic semantics
- computational semantics: by using fundamental techniques for computing semantic representations of natural language expressions

The technique of teaching is a combination of lectures on theoretical concepts of the formal apparatus of lambda calculus and practical work with logic software in computer labs, assignments and projects. In the combination of theoretical lectures and experimenting with computerized tools, students learn

- first order model building and checking
- first order theorem proving
- unification
- resolution systems
- the central concepts of quantifiers and scope ambiguities

Students acquire experience with software for automated calculations of truth values and meaning representations (see [1] and [3]).

For more details, see the course website [9].

Computational Semantics II

The goals of course is to introduce the advanced methods of higher order logic for more adequate coverage of semantics of natural language and computing semantic representations of natural language expressions. The lectures are introduction to higher order logic and modern approaches to computational semantics: Possible World Semantics, Situation Semantics, Minimal Recursion Semantics, Discourse Representation.

In computer labs, assignments and projects students practice on the meaning representations of quantifiers and scope ambiguities by using logic software [1] and [3]. Then we experiment on what they have learned in several courses by using grammar software handling semantic representations and syntax-semantics interface. For this purpose, we have used the LKB grammar tool, see [5].

For more details, see the course website [8].

2.3. Learning Experiences

From my experience, the difficulties in the above courses divide into two loci: one from the perspective of the students and the other, on the side of the lecturer. But both loci are grounded on one major problem: the limited time of the courses. When a course meets twice per week, it is typically given just in 5 weeks. For such a short period of time, students can not comprehend the nature of abstract theoretical concepts and have little time to practice on their computational aspects, computer implementations, programming, etc. On the side of the lecturer, the difficulties are on the choices of what and how much of the material to be presented, the level of introducing abstract concepts and how much to be left for self-study. Often, the course is limited to intuitive presentations of the concepts supplemented with examples, some formal definitions and little on major results, formal proofs, algorithms, methods and techniques for implementations. The limited time spent on major results and formal characteristics of abstract concepts compromises students' comprehension of methods and techniques for computational implementations.

On the positive side, despite of the above difficulties, by taking a set of courses which includes material on foundations and computability, students learn computational concepts, acquire sound senses for symbolic manipulations and algorithmic work. This reflects highly positively on their abilities to work on implementations by using or developing software, and after all, gives them a good start in developing professional skills.

3. GAPS IN EDUCATION ON FORMAL METHODS

By my experience (in mathematics, informatics (computer science), linguistics) teaching formal methods to non-major students can be very successful. However, currently there is a strong tendency for cutting down course hours or entire courses, and wide spread attitude to formal methods as highly abstract material with little applications.

For providing needed up-to date background, science students (e.g. in physics, chemistry, medicine, biology, etc.) are required to take relevant courses in pure math including material on Calculus, Differential Equations, Statistics and Probability, numerical methods, scientific computing, etc. This is needed in the professional life of specialists, either directly for their work, or indirectly for communicating and comprehending information including math. Then, the following questions arise: How many of the science researchers, say in medicine, biology, or even in bioinformatics, are aware about the advances in formal methods of logic and computability? How many logicians are involved in interdisciplinary work² in medical sciences and pharmaceutical industry? Even those medical graduates who have taken courses in Logic, or Discrete/Finite Math end up with vague memories of elementary set theory, boring truth tables of propositional calculus,

²As some of the talks given at the Formal Methods 2006 Symposium, demonstrate, there are initiations for such work.

and some elementary first order predicate logic. In most cases, natural science (medicine, biology, etc.) communities consider logic as a branch of philosophy without any practical use in their professional practice. This is often a view even of researchers working on top edge scientific advances on medical and biological models and simulations. How many of them know that logic and computability theory provide powerful methods for formal specification of models and verification?

Currently, there are two vast areas, namely, medical and pharmaceutical research, which have fallen behind in using the advances in formal methods of mathematical and computational logic. Trial studies in medicine, pharmacology and biology, which include tests with animals and humans, use sophisticated methods of various branches of pure mathematics such as statistics and probability, multivariable analysis, and other numerical methods. There are powerful statistical methods for testing medical and biological models. However, those models are usually specified in natural language, which despite of the specialized terminology and style has all features of human language, especially, underspecification and ambiguity. Something more, models described in human language are vulnerable to inconsistency and gaps in the chains of compound information and reasoning.

Educational programs in natural sciences, such as medicine and biology, should offer specialized introductory courses in formal methods. These courses would provide foundations for interdisciplinary and cooperative work on research and development of advanced industry technologies, building sound models and simulation, and automatic verification of complex systems.

REFERENCES

- [1] Barwise, Jon and John Etchemendy, et al., 1999. *Language, Proof, and Logic*. CSLI. Stanford. (<http://www-csli.stanford.edu/LPL/>)
- [2] Beesley, Kenneth R. and Lauri Karttunen. *Finite State Morphology*. CSLI Publications. 2003.
- [3] Blackburn, P. and Bos, J. 2005. *Representation and Inference for Natural Language*. CSLI Publications and University of Chicago Press.
- [4] Chomsky, Noam. 1957. *Syntactic Structures*. The Hague: Mouton.
- [5] Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications. <http://wiki.delph-in.net/moin/LkbTop>
- [6] Loukanova, R., VT 2003, Foundations of Computer Science I (Logic) <http://stp.ling.uu.se/~rloukano/classes/2003s/logik/index.html>
- [7] Loukanova, R., HT 2006, Foundations of Computer Science II (Theory of Formal Languages and Automata) http://stp.ling.uu.se/~rloukano/classes/06a/math_ling/syllabus.html
- [8] Loukanova, R., VT 2006, Computational Semantics II, http://stp.ling.uu.se/~rloukano/classes/06s/compsem_2/syllabus.html
- [9] Loukanova, R., HT 2006, Computational Grammar II, http://stp.ling.uu.se/~rloukano/classes/06a/compgr_2/syllabus.html
- [10] Montague, R. 1970. *English as a formal language*. In B. Visentini et al., (eds.) *Linguaggi nella Società e nella Tecnica*. Milan: Edizioni di Comunità. Reprinted in R. Thomason, editor, *Formal Philosophy*, 188-221. New Haven: Yale University Press.
- [11] Sag, Ivan A., Thomas Wasow, and Emily M. Bender. *Syntactic Theory - A Formal Introduction*. Stanford: CSLI Publications. 2003.