What is Computational Semantics?
What is Functional Programming?
What is the Connection?

Jan van Eijck, CWI Amsterdam and Uil-OTS Utrecht
Christina Unger, Uil-OTS Utrecht

LOT Summer School, June 15, 2009

Overview

- What is computational semantics?
- Why use functional programming for computational semantics?
- Today, as a first sample of computational semantics, we present a natural language engine for talking about classes.
- Material for this course is taken from Jan van Eijck and Christina Unger, Computational Semantics with Functional Programming, Cambridge University Press 2009 (to appear).
- `http://www.cwi.nl/~jve/cs/`
- But first: Getting Started with Haskell

What is computational semantics?
The art and science of computing or processing meanings

- ‘meaning’: informational content, as opposed to syntactic ‘form’
- ‘computing’: what computers do
- ‘processing’: what happens in the human brain

A Brief History of Formal Linguistics

A Brief History of Formal Linguistics

1916 Ferdinand de Saussure, Cours de linguistique générale published posthumously. Natural language may be analyzed as a formal system.

1957 Noam Chomsky, Syntactic Structures, proposes to define natural languages as sets of grammatical sentences, and to study their structure with formal (mathematical) means. Presents a formal grammar for a fragment of English.

1970 Richard Montague, English as a Formal Language, proposes to extend the Chomskyan program to semantics and pragmatics. Presents a formal grammar for a fragment of English, including semantics (rules for computing meanings). Links the study of natural language to the study of formal languages (languages from logic and computer science).

Richard Montague (1930-1971)

Developed higher-order typed intensional logic with a possible-worlds semantics and a formal pragmatics incorporating indexical pronouns and tenses.

Program in semantics (around 1970): universal grammar.

Towards a philosophically satisfactory and logically precise account of syntax, semantics, and pragmatics, covering both formal and natural languages.

“The Proper Treatment of Quantification was as profound for semantics as Chomsky’s Syntactic Structures was for syntax.” (Barbara Partee on Montague, in the Encyclopedia of Language and Linguistics.)

- Chomsky: English can be described as a formal system.
- Montague: English can be described as a formal system with a formal semantics, and with a formal pragmatics.

Montague’s program can be viewed as an extension of Chomsky’s program.

The Program of Montague Grammar

- Montague's thesis: there is no essential difference between the semantics of natural languages and that of formal languages (such as that of predicate logic, or programming languages).
- The method of fragments: UG [6], EFL [5], PTQ [4]
- The misleading form thesis (Russell, Quine)
- Proposed solution to the misleading form thesis
- Key challenges: quantification, anaphoric linking, tense, intensionality.

Misleading Form

Aristotle's theory of quantification has two logical defects:

1. Quantifier combinations are not treated; only one quantifier per sentence is allowed.
2. 'Non-standard quantifiers' such as most, half of, at least five, . . . are not covered.

Frege's theory of quantification removed the first defect.

The Fregean view of quantifiers in natural language: quantified Noun Phrases are systematically misleading expressions.

Their natural language syntax does not correspond to their logic:

"Nobody is on the road" $\rightsquigarrow \neg\exists x(\text{Person}(x) \wedge \text{OnTheRoad}(x))$

Solution to the Misleading Form Thesis

expression	translation	type
every	**every**	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$
princess	P	$(e \rightarrow t)$
every princess	**every** P	$(e \rightarrow t) \rightarrow t$
laughed	S	$(e \rightarrow t)$
every princess laughed	$($**every** $P)\ S$	t

where **every** is a name for the constant $\lambda P \lambda Q.\forall x(Px \rightarrow Qx)$.

From semantics to pragmatics

- Analysing communication as flow of knowledge.
- Logical tool: Dynamic Epistemic Logic
- Computational tool: Dynamic Epistemic Model Checking
- DEMO: Epistemic Model Checker [1]
- Will be discussed on the last day of the course.

A Brief History of Functional Programming

1932 Alonzo Church presents the lambda calculus

1937 Alan Turing proves that lambda calculus and Turing machines have the same computational power.

1958 John McCarthy starts to implement LISP.

1978-9 Robin Milner cs develop ML.

1987 Agreement on a common standard for lazy purely functional programming: Haskell.

Haskell λ

A Purely Functional Language

http://www.haskell.org

http://www.haskell.org/hugs/

Natural language analysis and functional programming

- Usefulness of typed lambda calculus for NL analysis.
- Linguist Barbara Partee: “Lambda’s have changed my life.”
- Computational linguistics: From Prolog to Haskell?
- Appeal of Prolog: Prolog-style unification [8], ‘Parsing as Deduction’ [7]
- But a new trend is emerging [2, 3]
- NLP Resources in Haskell: see

  http://www.haskell.org/haskellwiki/Applications_and_libraries/Linguistics

References

[1] Jan van Eijck. DEMO — a demo of epistemic modelling. In Johan van Benthem, Dov Gabbay, and Benedikt Löwe, editors, Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop, number 1 in Texts in Logic and Games, pages 305–363. Amsterdam University Press, 2007.

[2] R. Frost and J. Launchbury. Constructing natural language interpreters in a lazy functional language. The Computer Journal, 32(2):108–121, 1989.

[3] Richard A. Frost. Realization of natural language interfaces using lazy functional programming. ACM Comput. Surv., 38(4), 2006.

[4] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka, editor, Approaches to Natural Language, pages 221–242. Reidel, 1973.

[5] R. Montague. English as a formal language. In R.H. Thomason, editor, Formal Philosophy; Selected Papers of Richard Montague, pages 188–221. Yale University Press, New Haven and London, 1974.

[6] R. Montague. Universal grammar. In R.H. Thomason, editor, Formal Philosophy; Selected Papers of Richard Montague, pages 222–246. Yale University Press, New Haven and London, 1974.

[7] F.C.N. Pereira and H.D. Warren. Parsing as deduction. In Proceedings of the 21st Annual Meeting of the ACL, pages 137–111. MIT, Cambridge, Mass., 1983.

[8] S.M. Shieber. An Introduction to Unification Based Approaches to Grammar, volume 4 of CSLI Lecture Notes. CSLI, Stanford, 1986. Distributed by University of Chicago Press.