COMPUTATIONAL SEMANTICS WITH FUNCTIONAL PROGRAMMING

Computational semantics is the art and science of computing meaning in natural language. The meaning of a sentence is derived from the meanings of the individual words in it, and this process can be made so precise that it can be implemented on a computer. Designed for students of linguistics, computer science, logic and philosophy, this comprehensive text shows how to compute meaning using the functional programming language Haskell. It deals with both denotational meaning (where meaning comes from knowing the conditions of truth in situations), and operational meaning (where meaning is an instruction for performing cognitive action). Including a discussion of recent developments in logic, it will be invaluable to linguistics students wanting to apply logic to their studies, logic students wishing to learn how their subject can be applied to linguistics, and functional programmers interested in natural language processing as a new application area.

JAN VAN EIJCK is a Senior Researcher at CWI, the Centre for Mathematics and Computer Science in Amsterdam, and Professor of Computational Linguistics at Uil-OTS, the Research Institute for Language and Speech, Utrecht University.

CHRISTINA UNGER is based in the Semantic Computing Group in the Cognitive Interaction Technology Center of Excellence at the University of Bielefeld.

COMPUTATIONAL SEMANTICS WITH FUNCTIONAL PROGRAMMING

JAN VAN EIJCK AND CHRISTINA UNGER

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,
São Paulo, Delhi, Dubai, Tokyo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521757607

First published 2010

Printed in the United Kingdom at the University Press, Cambridge

A catalogue record for this publication is available from the British Library

ISBN 978-0-521-76030-0 Hardback
ISBN 978-0-521-75760-7 Paperback

Contents

Foreword

The view of computational semantics that informs and drives this book is one that sees the computation of linguistic meaning as that of computing logically transparent representations of meaning from "raw linguistic data", linguistic input as it reaches a recipient when he hears something or reads it, and to which he has to attach a sense. The book abstracts away from the differences between hearing and reading in that it assumes that the "raw data" have already been preprocessed as strings of words. Such computations of meaning are generally assumed to involve the computation of structure at several levels, minimally at a level of syntactic form and then, via the syntactic structure obtained at that level, of the semantic representation. This implies that in order to do computational semantics properly you need proficiency in at least three things: (i) proficiency in computation (you need to be proficient in the use of at least one suitable programming language), (ii) proficiency in syntax, and (iii) proficiency in semantics, in the more narrow sense in which semantics is understood by many linguists, but also in a broader sense.

The message this book drives home is that computing semantic representations from "raw data" isn't all there is to computational semantics. Computing semantic representations wouldn't be of much use to us if, once we have constructed them, there wouldn't be anything we could do with them. But of course there is. One thing we do with the semantic representations we construct from linguistic input is to employ them as premises, usually in conjunction with representations we already have, and that often come from other sources (e.g. from what we have seen with our own eyes). In this way the new representation may yield additional information, information that follows neither from the information we already had, nor from the new representation when taken by itself. Or the new representation may be instrumental in practical reasoning, help us to develop a better idea of how we should proceed in order to get what we want. It is because we use semantic representations in these inferential ways that they must be "logically transparent":

they must be in a form on which the inference mechanisms of formal logic must have a purchase.

There is also something else that we can do with the semantic representations we get from what we hear or read: we evaluate them against models of the world. These can be models that we have put together on the basis of, say, what we learn by seeing, hearing, or touching. "Model checking", i.e., checking whether what a semantic representation says about a given model or data structure obtained in some such way is true or false, is no less important than drawing inferences from it. Model checking is implicit in what speakers do when they choose sentences to describe the models or data structures about which they want to say something, and it is what the recipient of an utterance does when he has independent access to the model or data structure that he assumes the speaker is talking about, for instance in situations where the speaker is making a comment on something which they are both looking at.

The importance of model checking and of the inferential uses we make of language may be plain enough in any case. But how important they really are comes even more dramatically into focus in connection with developing language skills for robots, which can tell us about things that they can see but we cannot, or that must be able to communicate with us when we are jointly working on a common task. Such uses of semantic representations are what makes them worth having in the first place. A computational semantics that tells us how representations get built, but has nothing to say about what is done with them would hardly be worth having.

This book adds accounting for the uses of semantic representations to the Computational Semantics task list, thus making the agenda more ambitious than it would have been in any case by a good stretch. As far as I know no one has so far made an attempt to address this amplified agenda within the scope of a single text. This is one respect in which the present book breaks new ground, both conceptually, as a presentation of what computational semantics, defined by this agenda, is actually like, and pedagogically, by introducing the readers to the various different items on this agenda and showing them how each of those items can be tackled and what is needed for that.

The book's third novelty is that it starts without any presuppositions. Nothing is assumed beyond common sense: no programming skills, no knowledge of syntax, no knowledge of semantics, no knowledge of natural language processing of any sort. The book begins by introducing the basics of Haskell, the programming language that is used throughout. Haskell is a member of the family of functional programming languages and suitable for the theoretical purposes of this book because of its exceptional transparency. The use of Haskell in an introduction to computational semantics is a departure from the widespread use of Prolog in intro-

ductions to symbolic natural language processing. The authors motivate this choice by pointing out that much of symbolic computational linguistics consists, like so many other types of computational problems, in defining the right data structures and the right functions that map one data structure (the one that presents the problem) into another (the one that presents its solution).

Indeed, the first applications that are shown in the book are non-linguistic, and chosen solely for the purpose of giving the reader a good grasp of how Haskell works and what can be done with it. From these first applications there is a gradual progress, via applications that involve elementary forms of language processing (such as word counts in texts), to applications that belong to computational semantics proper. One important benefit of this way of easing into the subject is that it makes the reader aware of how much language processing has in common with problem solving and data processing tasks that do not involve language. On the face of it such an awareness might sit awkwardly with the cherished view of many linguists that our ability to acquire and use language is unique among the cognitive capacities we have; but I do not think that there is likely to be any real conflict here. As demonstrated amply by the diversities of the actual algorithms and programs presented in this book, what similarities between linguistic and non-linguistic processing there are still leaves plenty of room for the cognitive distinctness and uniqueness of those algorithms that characterise the computational aspects of the human language capacity.

Although the book starts from scratch, and presupposes nothing beyond common sense, it nevertheless manages to penetrate into some of the more advanced areas of computational semantics: the final chapters present continuation semantics, discourse representation theory, and a system of dynamic epistemic logic that serves as a first step in the direction of a full-fledged computational theory of verbal communication. And yet, this wide span notwithstanding, the book complies with current demands on texts in computational linguistics in that it presents for every bit of theory that it introduces a Haskell implementation, which the reader can run herself and play with, if she downloads the relevant, freely available supporting software. The same goes for the numerous implementation-related exercises. This means that the diligent student of this book will, by the time she gets to the end of it, have not only learned a good deal about syntax and semantics, but have also acquired much of that which distinguishes the computational from the theoretical linguist.

All this goes to make this book into the important and innovative contribution to the field of computational semantics that I think it is. Most important of all is the influence that I believe and hope it will have on coming generations of computational linguists, by instilling in them the elements of sophisticated programming expertise at the same time as introducing them to many aspects of theoretical lin-

guistics, and conveying some of that passion for the structural aspects of language that makes theoretical linguistics a must and (sometimes) a source of satisfaction for the true linguist. It has been pointed out again and again that the practical potential of computational linguistics, and of computational semantics as an essential part of it, is immense. Still, it is fair to say that the practical achievements of computational semantics have so far been quite limited. The reasons for that, I think, are two-fold. Automated symbolic processing of natural language is notoriously brittle: even where it is clear what the system should compute, it often lacks the necessary resources, in particular wide coverage lexicons with substantive semantic information and world knowledge in accessible form. But in many cases the problem goes deeper. We still haven't even properly understood yet what it is that should be computed. These are hard problems, which will be solved – to the extent that they can be solved at all – only by people who have the combination of skills that are presented and taught here as parts that naturally fit into a coherent whole.

HANS KAMP

Professor of Formal Logic
and Philosophy of Language,
University of Stuttgart

Preface

This book on applications of logic in the semantic analysis of language pays the reader the compliment of not assuming anything about what he or she knows (in particular, no specific logical knowledge or experience with programming is presupposed), while making very flattering assumptions about his or her intelligence and interest in the subject matter.

The method used throughout in the book is the pursuit of logical questions and implementation issues occasioned by concrete examples of formally defined language fragments. At first, no distinction is made between formal and natural language; in the first chapter it is explained why. At the end of the text the reader should have acquired enough knowledge and skills for the development of (at least the semantic part of) fairly serious Natural Language Processing applications. The reader who makes it to the end of the book will also find that he or she has acquired considerable programming skills, and will have learned how to put a wide variety of logical systems to use for natural language analysis.

Throughout the text, abstract concepts are linked to concrete representations in the functional programming language Haskell. Haskell is a language that is well suited for our purposes because it comes with a variety of very easy to use interpreters: Hugs, GHCi, and Helium. Haskell interpreters, compilers, and documentation are freely available from the Internet.† Everything one has to know about programming in Haskell to understand the programs in the book is explained as we go along, but we do not cover every aspect of the language. Further on we will mention various good introductions to Haskell. The only thing we do assume is that the reader is able to retrieve the appropriate software from the Internet, and that he or she is acquainted with the use of a text editor for the creation and modification of programming code.

The book is intended for linguists who want to know more about logic, including

† See `http://www.haskell.org`

recent developments in dynamic logic and epistemic logic, and its applicability to their subject matter, for logicians with a curiosity about applications of their subject to linguistics, and for functional programmers who are interested in a new application domain for their programming skills.

This text has quite a long prehistory. The prefinal version of this book grew out of a series of lectures the first author gave at UiL OTS in the fall of 2000, for a mixed audience of linguists and formal semanticists. This was extended with material from a tutorial at the LOLA7 Pecs Summer School of 2002, with the results of an implementation project for Theta Theory at UiL OTS, Utrecht, in the fall of 2002 (follow-up to a course on Theta Theory that the first author co-taught with Tanya Reinhart), and with examples from a natural language technology course developed in collaboration with Michael Moortgat.

This manuscript would have remained only an Internet resource if Helen Barton, CUP editor in charge of linguistics and humanities, had not made a Spring visit to UiL OTS in 2008, in enthousiastic pursuit of suitable textbook manuscripts. And it would not have turned into the book you are now reading if it weren't for an intellectual support group, the *reading group*, which met up with the authors once every two weeks, September through December 2008, for feedback. We really owe its members a lot. The rock solid core of the group consisted of Arno Bastenhof, Gianluca Giorgolo, Jeroen Goudsmit and Andres Löh, who were always there. They kept us on track with their support and with their keen eye for failings of the draft chapters under discussion. The members of the outer shell of the group, who also made important contributions, were Marta Castella, Anna Chernilovskaya, Stefan Holdermans, Matthijs Melissen, and Hanna de Vries. An early version of the manuscript also had two anonymous Cambridge University Press readers, who encouraged us to carry on with the project and gave valuable feedback.

There is a web page devoted to this book, at address

`http://www.computational-semantics.eu`,

where the full code given in the various chapters can be found. Suggestions and comments on the text can be sent to the first author, at email address `jve@cwi.nl`, and will be very much appreciated.

Acknowledgements Many thanks to the reading group; the names of the members were mentioned above. Helen Barton, commissioning editor at CUP, was supportive and encouraging throughout, and CUP production editor Rosina Di Marzo gave us generous help in the final stage of production. Thanks also to Herman Hendriks for acting as an oracle on references, to Michael Moortgat for pleasant cooperation with the first author in teaching various natural language technology courses, to Rick Nouwen for joint work with the first author on implementations of pronomi-

nal reference algorithms, to Albert Visser for illuminating discussions on dynamic semantics, life and everything, to Krzysztof Apt and Willemijn Vermaat for advice on LaTeX matters, to the course participants of the first author's UiL OTS courses, to the participants in the 2002 Pecs Summer School Tutorial, and to Nicole Gregoire and Rianne Schippers, members of the 2002 Theta Theory implementation group for pleasant discussions. Theo Janssen made a number of useful suggestions concerning presentation. Aarne Ranta gave valuable advice, as did Doaitse Swierstra. Harm Brouwer, Matteo Capelletti, Daniël de Kok, Nickolay Kolev, Greg Matheson, Stefan Minica, Patrick O'Neill, Jim Royer, and Ken Shirriff sent us corrections via email. Yoad Winter, Makoto Kanazawa, and Reinhard Muskens corresponded with us about issues of intensionalization. Shalom Lappin tried out a version of the book in a course, which resulted in numerous update requests, inspiring editorial suggestions and bug reports. Finally, we thank Hans Kamp for having graced the book with a foreword.

JAN VAN EIJCK, Amsterdam
CHRISTINA UNGER, Bielefeld