Anaphora and Intensionality[1] *in Classical Logic*

Jørgen Villadsen
Technical University of Danmark

Abstract

The dynamic perspective on natural language suggests that texts change the value of contextual parameters in much the way computer programs change the value of program variables. One phenomenon where dynamic aspects emerge is that of anaphora (introduction of referents for later anaphoric links), but intensionality and tense are other candidate phenomena. However, most logics used in computer science for the study of the semantics of program are often rather complicated and different from classical logic. We describe a representation directly in classical logic using a categorial grammar. In the logic a theory is defined (via axioms) to describe how in a text a pronoun manages to pick up a referent that was introduced by a determiner. Besides (nominal) anaphora we discuss how to handle intensionality present in (belief and knowledge) attitude reports.

Introduction

We present a theoretical study of central semantical problems of anaphora and intensionality in natural language texts (English). A basic knowledge of anaphora, intensionality and logic is assumed.

As texts we take sequences of declarative and disambiguous sentences; hence each text has a unique meaning and the objective is to define a semantical theory specifying, in some way and to some extent, the meaning of each text.

We think that in order to use the texts to communicate our information about facts (or assumptions) concerning "the universe around us" the theory must handle anaphora and intensionality as amply present in natural language. For instance, consider the following *Love Story* text:

A man kisses a woman.

She believes that he loves her.

The pronouns (**he, she,...**) imply anaphoric links across sentences and attitude reports (**believe that,...**) imply intensional properties for embedded sentences.[2]

We require that the theory must be a so-called correspondence theory of meaning – that a relation between the language and certain "things" independent of the language must be specified. We also

Proceedings of NODALIDA 1991, pages 165-176

require that the theory is computational – that a "calculus" for manipulations of meanings of texts must be specified.[3] Both requirements can be directly satisfied using the model theory and the proof theory, respectively, of a formal logical system. For instance, returning to the first sentence of the *Love Story* we might tentatively represent it as the following formula in classical predicate logic:

$$\exists x\,(man\,x \wedge \exists\, y\,(woman\,y \wedge kiss\,x\,y))$$

The model theory says that man and woman denote sets of objects, that kiss is a relation between two objects and that the variables x and y can be instantiated to certain objects satisfying the given constraints. The proof theory, on the other hand, lists axioms and inference rules for deriving or rewriting the formula as such.

The representation of the second sentence is less straightforward. Even ignoring the proper representation of the embedded sentence construction (symbolised by $\boxed{\ldots}$) it seems necessary to redo the already completed representation of the first sentence to accomodate the second sentence:

$$\exists x\,(man\,x \wedge \exists y\,(woman\,y \wedge kiss\,x\,y \wedge belief\,y\ [\,love\,x\,y]\,))$$

This seems to render impossible a systematical and compositional analysis where the meaning of an expression is systematically composed from the meanings of its parts.

Also, a seemingly innocent change (conjoining the two previous sentences to a single conditional sentence) implies a drastic change for the representation (where the previous representations cannot be reused in any straight-forward manner):

If a man kisses a woman then she believes that he loves her.

$$\forall x\,(man\,x \rightarrow \forall y\,(woman\,y \rightarrow \ldots))$$

Of course, the representation in the logical system above is rather ad hoc. Discourse representation theory (see [2,3,4] for references) deals with translations of texts in a systematical way, but, as the name suggests, uses a separate representation level and does not yield a direct compositional analysis in a logical system.

Our claim is that a compositional analysis in a logical system is possible, but that texts are complicated and are best understood if one moves from static semantics (as above) to dynamic semantics (to be fully explained later).

It must be emphasized that we can do dynamic semantics for natural language in a logical system where the logical language itself has a static semantics! This can be done by choosing suitable representations and axioms in the logical system. We briefly illustrate this point by considering a recent non-classical logical system, namely dynamic predicate logic [3], which is based on the same language as classical predicate logic, but where the following entailment holds (ϕ and ψ any formulas):

$$(\exists x\phi[x]) \rightarrow \psi\,[x] \models \forall x(\phi[x] \rightarrow \psi\,[x])$$

Notice that the quantifier binds the variable outside the scope of the quantifier. The quantifiers and operators have different behavior: existential quantification is externally dynamic, universal quantification is externally static, and implication is internally dynamic (between operands) but externally static.

The entailment above allows for a compact and elegant treatment of many anaphora problems using dynamic predicate logic, but compared to classical predicate logic it is less flexible and transparent.

For instance the following simple entailment does not hold:

$$\phi \models \phi$$

However, a restricted version holds (when no overlap between active quantifier variables and free variables in the formula). See [3] for details about dynamic predicate logic as well as references to the extended logical system called dynamic Montague grammar.

Overview

First the general idea of logical semantics is described, subsuming both static and dynamic semantics. Then static semantics is defined (based on the notion of truth values) and hereafter dynamic semantics (based on the notion of information state changes).

Information states can be structured in many ways – we propose to let an information state be a set of alternatives.[4] Even without saying more about what an alternative is we obtain interesting definitions of vacuous, absurd and definite information states. Moreover, reductions are possible if the information state changes are distributive and eliminative.

We argue that by a proper construction of the set of alternatives anaphora can be handled in an elegant way. We propose a representation in logical type theory using a categorial grammar. A tiny example is provided. Finally we discuss how to incorporate intensionality in this framework.

A few words on notation. In schemas we take $\Leftrightarrow$ to mean: if and only if (for all appropriate instantiations with expressions of the shown schema variables). We have two equality symbols: $\equiv$ means equality by definition and = means equality by calculation (hence $\equiv$ implies =).

Logical Semantics for Natural Language

The main purpose of logical semantics for natural language is, we think, to provide a precise description of the intuitive notion of a valid argument.[5]

Arguments can be built in several ways. We take an argument to consist of a sequence of texts $t_1, \ldots, t_n$ (the premises) and a single text t_* (the conclusion). We arrange the argument as in the following schema with the symbol $\therefore$ in front of the conclusion:

$$\begin{array}{l} t_1 \\ \vdots \\ t_n \\ \therefore t_* \end{array}$$

We translate each text t in natural language to a formula $t^\bullet$ in a suitable logical language – a logic.[6] By a logic we mean a formal language endowed with a notion of entailment $\models$, which we require must match the notion of a valid argument in the following sense:

$$\begin{array}{l} t_1 \\ \vdots \\ t_n \\ \therefore t_* \end{array} \quad \Longleftrightarrow \quad t_1^\bullet, \ldots, t_n^\bullet \models t_*^\bullet$$

A formal language is a language with a precise syntax and semantics.[7] The syntax defines the set of formulas (for instance via formation rules). The semantics defines for each formula its meaning (for instance a mathematical object). In general we write the meaning of the formula $t^\bullet$ as $[t]$. This hides the translation, but no confusion can arise. We shall also refer to $[t]$ as the meaning of the text t.

We require that although entailment is introduced as relation $\models$ between formulas (syntactical objects) it must correspond to a relation $\mathcal{R}$ between the meanings of the formulas (semantical objects):

$$t_1^\bullet, \ldots, t_n^\bullet \models t_*^\bullet \iff \mathcal{R}([t_1], \ldots, [t_n], [t_*])$$

Since we aim at using mathematical objects as the meanings, our goal is to find objects with enough structure to make the definition of such a relation $\mathcal{R}$ possible. In the static semantics the objects are rather coarse-grained, but in the dynamic semantics to be described later, the objects are more fine-grained.

Static Semantics

In static semantics the meaning of a text is a truth value (0/1 for false and true) with respect to a so-called model. It would actually be more appropriate to say that the meaning then is a function from models to truth values, but since it turns out that the model parameter is common to all meaning compositions (based on the formation rules) it makes sense to use the "with respect to" phrase.

We write $[t]_M$ for the meaning of the text t with respect to the model $M = \langle D, d \rangle$, where D is the domain (a set of objects) and d is the denotation (a function from constants to mathematical structures of objects). There can be many kinds of constants: individual constants (denoting objects), first order predicate constants (denoting relations overobjects), second order predicate constants (denoting relations over objects as well as over relations over objects), etc. The denotation of a predicate constant is often called its extension. The so-called order of a logic is not determined by the orders of the constants present, but by the orders of the variables of quantification.

We assume that the formula $t^\bullet$ is closed (no occurrences of free variables).[8] To account for free variables in sub-formulas the meaning must be given with respect to an assignment a (a function from variables to mathematical structures of objects) besides the model M, but this complication can be ignored here. As for constants there can be many kinds of variables. In first order logic, quantification is allowed over individual variables only, whereas in higher order logic quantification is allowed over predicate variables too.

Entailment is defined as:

$$\begin{array}{l} t_1 \\ \vdots \\ t_n \\ \therefore t_* \end{array} \iff \text{For all } M\text{: if } [t_1]_M = \ldots = [t_n]_M = 1 \text{ then also } [t_*]_M = 1$$

Dynamic Semantics

Dynamics imply the (continuous or discrete) change of something – in case of natural language semantics we suggest to consider the change from one information state to another information state of an agent (human or program) processing the texts.[9] We first present the general idea leaving the details of information states unspecified. We then consider different specific structures as information states.

We intuitively have the following distinguished information states:

Vacuous information state
The agent has no information at all.

Absurd information state
The agent has contradictory information.

Definite information state
The agent has maximal information.

Often there is only one vacuous and one absurd information state, written I_0 and I_∞, whereas there are numerous definite information states.[10] We often have a partial ordering of information states corresponding to getting "better" informed.

In dynamic semantics we let $[t]$ be a function from information states to information states. Observe that there is no need to see the meaning relative to a model as in the static semantics (although it is possible to do so). If the initial information state is I_0 we have the following information state changes:

$$I_0 \xrightarrow{[t_1]} I_1 \xrightarrow{[t_2]} I_2 \xrightarrow{[t_3]} \cdots \xrightarrow{[t_n]} I_n \xrightarrow{[t_*]} I_*$$

We define the relevant resulting information states as follows:[11]

$$I_n \equiv [t_n](\ldots[t_1](I_0)) \qquad I_* \equiv [t_*](I_n)$$

There are several different definitions of entailment possible. We here follow the slogan: entailment makes explicit implicit information. More precisely: The conclusion t_* is implicit in the premisses $t_1, \ldots, t_n$ if no (important) change in information state occurs when t_* is processed after $t_1, \ldots, t_n$ have been processed:

$$\begin{array}{l} t_1 \\ \vdots \\ t_n \\ \therefore t_* \end{array} \quad \Longleftrightarrow \quad I_n \approx I_*$$

Here $\approx$ is a suitable equivalence relation that compares information states for (important) changes; if all changes are important then we can use $=$ for $\approx$.

Information States as Sets of Alternatives

One way to specify information states is to let each information state be a set of alternatives that has to be taken into account. Growth of information then comes down to elimination of alternatives, up to the level of definite information where all but one of the alternatives have been eliminated.

Let $A \equiv \{a_1, a_2, a_3, \ldots\}$ be a set of not further specified alternatives and let an information state I be a set of alternatives ($I \subseteq A$). The definitions of the vacuous and absurd information states are:

$I_0 \equiv A \quad I_\infty \equiv \emptyset$

The definite information states are I = $\{a\}$, $a \in A$.
The information states have a partial ordering via the subset relation $\subseteq$. For example:

$$\{a_1\} \subseteq \{a_1, a_2\} \subseteq \{a_1, a_2, a_3\}$$

$$\{a_1, a_2\} \not\subseteq \{a_2, a_3\}$$

For a particular information state change $[t]$ there are some interesting properties it may have:

Distributive information state changes $[t](I) = \bigcup_{a \in I} [t](\{a\})$ for all I

Eliminative information state changes $[t](I) \subseteq I$ for all I

Distributive information state changes mean that the agent works "point-wise", i.e. based on the definite information states. This property does not hold for agents dealing with defaults etc. An illustrative example:

$t \equiv$ **John might lie.**

$$[t](I) = \begin{cases} I & \text{if there is an } a \in I \text{ such that "John lies".} \\ I_\infty & \text{otherwise.} \end{cases}$$

Note that if one has the information that "John does not lie" it is absurd to process t. It is easily seen from the above definition that the information state change is not distributive, since the test $a \subseteq I$ does not work point-wise. However, the information state change is eliminative, since $I_\infty \subseteq I$ for all I.

Eliminative information state changes means that the agent gets "better informed" towards the definite information states (with the exception that the only change from a definite information state is to the absurd information state). This property does not hold for agents dealing with revision etc.

When these two properties are "universal" – that is fulfilled for all texts t considered – optimisations are possible:

- Distributive information state changes:
  Instead of meaning [t] as a function from information states to information states, that is a function from sets of alternatives to sets of alternatives, we just specify a relation $\rightarrow_t$ between alternatives:

  $$a \rightarrow_t a' \iff a' \in [t](\{a\})$$

The meaning $[t]$ is then obtained as:

$$[t](I) = \bigcup_{a \in I} \{a' \mid a \rightarrow_t a'\}$$

The condition $a \rightarrow_t a'$ corresponds to the following idea in computer science:[12]

Non-deterministic execution of program t when in program state a yields program state a'.

- Distributive and eliminative information state changes:
  The relation $\rightarrow_t$ now only holds between identical alternatives; hence we just specify a set S_t of alternatives.

We then simply have:

$$a \rightarrow_t a' \iff a = a' \wedge a \in S_t$$

Observe that if the information state changes are eliminative, but not distributive, optimisations of the abovementioned kinds are not possible (although these information state changes are conceptually simplified).

Some final observations: Classical logic, which has a static semantics, can be viewed as having an optimised dynamic semantics, where sets of models play the role as information states and all information state changes are both distributive and eliminative. Also, and this is the line we shall follow here, if the structure of the alternatives is not too complicated, it ought to be possible to make denotations of classical logic function as information states. Since information states are sets of alternatives we need a logical system that makes sets (for instance as characteristic functions) available as first class citizens. Logical type theory is such a logic.

Logical Type Theory

A type theory asserts certain terms to be of certain types. A type theory can make other assertions too; for instance about the equality of terms or types. Terms and types belong to formal languages. These language will be distinct here (but are often the same in advanced type theories).

In the simple type theory a type τ is either basic (interpreted as a set of objects) or of the form $\tau \rightarrow \tau''$ *(interpreted as the set of functions from type* τ' to type τ'').[13] The terms are so-called λ-terms from the λ-calculus.[14]

In (simple) logical type theory the types are simple types with t as the basic type of propositional formulas. The set of object of type t are the set of (classical) truth values $\{0,1\}$. Axioms and inference rules must be given to ensure that the λ-calculus can be seen as a higher-order logic based on functions and predicates. The latter are characteristic functions (functions returning truth values). For our purposes it suffices to regard the terms as formulas in classical first-order predicate logic augmented with λ-terms. We write $\phi : \tau$ when the formula ϕ has type τ. See [5] for further details about logical type theory.

Anaphora

The strategy for anaphora handling is to let each alternative be a pair: world and environment, resembling the model and the assignment in case of static semantics. Hence a world is a function from constants to mathematical structures of objects and an environment is a function from stores to objects.[15]

We have a separate store for each (anaphoric) discourse markers indicated by a natural number index:

A_1 man kisses a_2 woman.

She_2 loves him_1.

This gives distributive information state changes, since the different environments can be seen point-wise. If we apply the abovementioned optimisation for distributive information state changes then we get non-eliminative information state changes, since the definite information states must also be used and not only the information states resulting from processing the texts.

Note: The chain of information states $I_0, I_1,...I_n$ has eliminative information state changes with respect to $t_1,...t_n$; hence for $1 \leq i \leq n$ we have:

$I_i \equiv [t_i](I_{i-1}) \subseteq I_{i-1}$

Representation in Logical Type Theory

Besides the basic type t of truth values we introduce the following basic types:

e Entities

i Indices

The variables *x,y,z* and *v,w* are of types *e* and *i* respectively.

The worlds are represented via models for logical type theory and the type of entities.[16] The environments are represented via the type of indices using a special constants for each store, that is for each (anaphoric) discourse marker:[17]

$m_1,...,m_k : i \rightarrow e$

The following definition (for $1 \leq i \leq k$) expresses the equality of two environments on the values of all stores except store *j*:

$$\sim_i \equiv \lambda v w(\bigwedge_{\substack{1 \leq j \leq k \\ j \neq i}} m_j v = m_j w)$$

The following axioms (for $1 \leq i \leq k$) expresses that we can change each value of each store in all environments independently:

$\forall v \forall x \exists w (v \sim_i w \wedge m_i w = x)$

Example using a Categorial Grammar

We use the following categorial grammar (here written as a phrase structure grammar, but see [1] for details and a more substantial fragment):

S	→ NP VP	Sentence
VP	→ walk | whistle | ...	Verb Phrase
NP	→ DET CN | he_1 | ... | he_k	Noun Phrase
DET	→ a_1 | ... | a_k	Determiner
CN	→ man | ...	Common Noun

Consider the following tiny example (morphology ignored):

$t \equiv$ **A_1 man walks.**

Translation of lexical entries (variables X,Y of type $e \to i \to i \to t$):

$$\begin{aligned}(a_i)^\bullet &\equiv \lambda X Y \lambda v w \exists v' v''(v \sim_i v' \wedge X\,(m_i\,v')\,v'\,v'' \wedge Y\,(m_i\,v')\,v''\,w)\\ (he_i)^\bullet &\equiv \lambda X \lambda v w (X\,(m_i\,v)\,v\,w)\\ (\rho)^\bullet &\equiv \lambda x \lambda v w (v = w \wedge \rho^\dagger\,x)\end{aligned}$$

where $\rho^\dagger \equiv$ walk | whistle | man | ... are constants of type $e \to t$ corresponding to $\rho =$ walk | whistle | man | ...

Based on the grammar, where string concatenation corresponds to function application (shown as juxtaposition as usual), we obtain the following meaning after λ-conversions:

$$\to_t = (a_i)^\bullet\,(\text{man})^\bullet\,(\text{walk})^\bullet = \lambda v w (v \sim_1 w \wedge \text{man}\,(m_1\,w) \wedge \text{walk}\,(m_1\,w))$$

Analogously:

$t' \equiv$ **He_1 whistles.** $\to_{t'} = \lambda v w (v{=}w \wedge \text{whistle}(m_1,w))$

Sentence conjunction uses the operator $\oplus \equiv \lambda p q \lambda v w \exists v' (pvv' \wedge qv'w)$ (variables p,q of type $i \to i \to t$) and using the above-mentioned axioms we have the desired result:

$$\to_{tt'} = \oplus\,t^\bullet\,t'^\bullet = \lambda v w (v \sim_1 w \wedge \text{man}\,(m_1\,w) \wedge \text{walk}\,(m_1\,w) \wedge \text{whistle}\,(m_1\,w))$$

All there remains is to spell out the details of a suitable relation $=$ and we have our entailment $\models$ using a systematical and compositional analysis in a classical logic.

Intensionality

A common approach to intensionality rests on the notion of "possible worlds" (see 1), usually manipulated by new operators like □ and ◊ Diamond. Such a "possible world" corresponds to the world component of our alternatives in an information state. However, in our representation in logical type theory we have kept a fixed world via models for the logical type theory and the type of entities (the type of indices is used for the environments together with the discourse markers), but this setup will not work any longer. One way out is to introduce a new basic type w of worlds and let $\rho^\dagger$ from above be constants of type $w \to e \to t$ instead of type $e \to t$ just. A special marker m_{world} of type $i \to w$ would then select the current world. We intend to spell out the details in a future paper.

Notes

1) Notice that "intention" means purpose, aim, etc. whereas "intension" here (as well as in logic and the philosophy of language) is the direct opposite of "extension" (cf. the distinction between sense and reference)!
2) Neither the anaphora resolution nor the intensionality characterisation are too important here. For the latter the following example might help: Even though the properties of being an unmarried man and being a bachelor are extensional equivalent (fulfilled by the same indivi-

duals) they are not (always) intensional equivalent; believing that a particular individual is an unmarried man without believing that the same individual is a bachelor is entirely plausible.

3) However, if meanings are specified as mathematical objects and structures then in principle it is always possible to use, say, an object-language axiomatisation of the number and set theory of the meta-language.
4) It is most convenient to think of an alternative as a so-called (possible) world – that is, a complete description of a "state of affairs" – but other choices can be made as to be shown.
5) We do not require that such a description must provide an explanation of how humans perform the judgement.
6) Remember that we assume the text to be disambiguous. If the text was ambiguous the proper translation would depend on, but not necessarily be determined by, the context.
7) A formal calculus has just a precise syntax – possibly with some syntactical operations defined. Occasionally the syntactical operations are regarded a having semantical content themselves.
8) Anaphora in texts is not to be handled by free variables in formulas.
9) An information state is also called a knowledge state or a belief state.
10) One sure way to obtain contradictory information is to gather all possible and impossible pieces of information – this explains the ∞ sign.
11) Better notation if argument before function, namely $I_n \equiv I_o[t_1]...[t_n]$ etc.
12) Program states and information states are different notions. Also in computer science we have a fixed interpretation in mind. See [5] for the application of dynamic logics to computer science.
13) There are other interpretations for the simple type theory than the element/set one given here, for example proof/proposition and program/specification (due to the similarity in the explanation of these pairs of notions).
14) The λ-calculus comes in many guises – also without types.
15) More precisely from store names to objects (likewise with constant and variable it symbols).
16) The type of entities plays much the same role in dynamic semantics as the (single) domain for static semantics. This changes when intensionality is added.
17) The environment/store matrix is transposed – instead of environments taking a store we have stores taking an environment (or rather an index).

References

Jørgen Villadsen: Combinatory Categorial Grammar for Intensional Fragment of Natural Language. In Scandinavian Conference on AI '91, 328-339, IOS Press, 1991.

Reinhard Muskens: Anaphora and the Logic of Change. In Logics in AI '90, 412-427, Springer Lecture Notes in Computer Science 478, 1991.

Jeroen Groenendijk and Martin Stokhof: Dynamic Predicate Logic. Linguistics and Philosophy, 14, 39-100, 1991.

Johan van Benthem: Semantic Parallels in Natural Language and Computation. In Logic Colloquium '87, 331-375, North-Holland, 1989.

Robert Goldblatt: Logics of Time and Computation. Lecture Notes 7, Center for the Study of Language and Information, Stanford University, 1987.

Peter Andrews: An Introduction to Mathematical Logic and Type Theory: To Truth through Proof, Academic Press, 1986.

Jørgen Villadsen
Department of Computer Science
Technical University of Denmark
ID/DTH, Building 344
DK-2800 Lyngby, Denmark

E-mail: jv@id.dth.dk