English as a Formal System

Reinhard Muskens

Tilburg University

Aims

- In this talk I will discuss some ideas about a natural logic that uses the abstract terms of abstract categorial grammars/lambda grammars as its main vehicle for representation.
- $(\texttt{some woman})\,\lambda x.(\texttt{every man})(\texttt{loves}\ x)$
- These abstract terms are also very close to the Logical Forms (LFs) considered in generative grammar.
- This is very much work in progress. There is still a lot to do.

Previous Work

- Since abstract terms can also be considered to be the proof terms of proofs in categorial grammar, the present ideas are related to previous work by Van Benthem; Sánchez-Valencia; Bernardi; Zamansky, Francez and Winter; and others.
- But the strategy is not to annotate proofs in categorial grammar but to work directly with a Gentzen calculus for the proof terms/LFs.
- We also obviously build on work in Generalised Quantifier Theory.

Providing Abstract Terms with a Logic

- The abstract terms that we will use are built up using non-logical constants, variables, application, and abstraction.
- In a sense these have no logic at all: $\beta\eta$ conversion and that is it.
- We may assume that there is some homomorphic translation tr into a standard semantics, using rules like $tr(\texttt{some}) = \lambda P' \lambda P \exists x (P' x \wedge P x)$.
- That induces a logic on our terms. Let us call this the underlying interpretation.

Cascaded Interpretation

- We will not directly use the underlying interpretation. We are interested in proof rules that operate on abstract terms.
- These rules should of course be sound with respect to the underlying interpretation.
- But, while it would be easy to find rules that are also complete with respect to this interpretation (provided we allow extra constants), there is no sense in working with such rules (in that case we could just as well use the underlying interpretation).
- The hope is that for the language-like abstract terms it will be possible to formulate rules that capture the fast and automatic part of reasoning, while the rules of the object semantic level also capture the slow and concious ones.

The Boolean Core

- It is by now quite obvious that natural language has a Boolean core that is all-important.
- We basically have coordination in all categories. To some extent also negation.
- Monotonicity is at the heart of much reasoning. Traditional Logic to a large degree is based on monotonicity rules (Sánchez-Valencia).
- Intimate connection between (downward) monotonicity and syntactic requirements: negative polarity.
- But there is more. Van der Wouden & Zwarts (1992) single out properties like anti-additivity and antimorphicity as syntactically relevant.

Some Properties of Functions in a Boolean Algebra

- Upward Monotonicity: $F(X \wedge Y) \leq F(X) \wedge F(Y)$
- Downward Monotonicity: $F(X \vee Y) \leq F(X) \wedge F(Y)$
- Anti-additivity: Downward Monotonicity + $F(X) \wedge F(Y) \leq F(X \vee Y)$
- Antimorphicity: Anti-additivity + $F(X \wedge Y) \leq F(X) \vee F(Y)$ $F(X) \vee F(Y) \leq F(X \wedge Y)$
- While e.g. few children is downward monotonic, without is also anti-additive (Hoeksema 1983, as cited in VdW&Zw), and the Dutch niet (not) and allerminst (not at all) are antimorphic (VdW&Zw).

Relational Type Logic

- We will be working with a relational type logic.
- Basic types + the rule that $\langle \alpha_1 \ldots \alpha_n \rangle$ is a type if $\alpha_1, \ldots, \alpha_n$ are types.
- Objects of type $\langle \alpha_1 \ldots \alpha_n \rangle$ as their extensions have relations that take objects of type α_i as their i-th argument.
- $\langle\rangle$ is the type of propositions (extensions: truth values), comparable with type t.
- In general $\langle \alpha_1 \ldots \alpha_n \rangle \approx \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \rightarrow t$.
- Relational type logics can be found in Orey (1959), Schütte (1960). In Muskens (1989) it is shown how lambdas and single-step application can be interpreted. Muskens (2007) gives a hyperintensional interpretation.

Format of Our Sequents

- For each type $\langle \alpha_1 \dots \alpha_n \rangle$ there will be sequents $\Gamma \Rightarrow_{\langle \alpha_1 \dots \alpha_n \rangle} \Delta$. We will often drop the subscript $\langle \alpha_1 \dots \alpha_n \rangle$.
- In $\Gamma \Rightarrow_{\langle \alpha_1 \dots \alpha_n \rangle} \Delta$ the elements of Γ and Δ may be
  - Terms of type $\langle\rangle$ (i.e. formulas).
  - Terms of type $\langle \alpha_1 \dots \alpha_n \rangle$.
- If in $\Gamma_1, \Gamma_2 \Rightarrow_{\langle \alpha_1 \dots \alpha_n \rangle} \Delta_1, \Delta_2$ the sets Γ_1 and Δ_1 are the terms of type $\langle\rangle$ while Γ_2 and Δ_2 are the terms of type $\langle \alpha_1 \dots \alpha_n \rangle$, we interpret this as saying that

$$\bigcap_{\gamma \in \Gamma_2} [\![\gamma]\!]_a^M \subseteq \bigcup_{\delta \in \Delta_2} [\![\delta]\!]_a^M$$

for any model M and assignment a such that all $\varphi \in \Gamma_1$ are true in M given a while all $\varphi \in \Delta_1$ are false in M given a.

Another Format

- Sometimes it is handy to write sequents $\Gamma \Rightarrow \Delta$ as sets of labeled terms $\{\mathsf{L} : \gamma \mid \gamma \in \Gamma\} \cup \{\mathsf{R} : \delta \mid \delta \in \Delta\}$.
- This then enables us to write rules that apply to both sides of a sequent, e.g.

$$\frac{\Gamma, X : QAB}{\Gamma, X : QA(A \texttt{ and } B)} \text{(conservativity)}$$

- Or

$$\frac{\Gamma, -X : \texttt{some}\, AB}{\Gamma, X : \texttt{every}\, A(\texttt{not}\, B)} \text{(duality rule)}$$

Analytic Calculi

- We are interested in Gentzen calculi that are analytic.
- The usual axiom rule applies:

$$\frac{}{\Gamma, A \Rightarrow A, \Delta}$$

- Other axioms may be purely lexical, e.g.

$$\frac{}{\Gamma, \texttt{horse} \Rightarrow \texttt{animal}, \Delta}$$

- Terms that are $\beta\eta$ equivalent are taken to be equal.

Rules Suggested by the Format

- $$\frac{\Gamma, A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m, \Delta}{\Gamma, A_1 C, \ldots, A_n C \Rightarrow B_1 C, \ldots, B_m C, \Delta}$$
- (We require all sequents to be well-formed. In the case above that means Γ and Δ must consist of formulae.)
- $$\frac{\Gamma, A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m, \Delta}{\Gamma, \lambda x.A_1, \ldots, \lambda x.A_n \Rightarrow \lambda x.B_1, \ldots, \lambda x.B_m, \Delta}$$

  provided x is not free in any formula in $\Gamma \cup \Delta$.
- The last rule is a generalisation of the abstraction rule of Zamansky, Francez & Winter.

Reasoning with `and`, `or`, and `not`

- We first concentrate on `and`, `or` and `not`. The first two will be in infix notation.
- We need rules that allow us to get rid of these operators as much as possible.

Rules for and, or, and not

$$\frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \texttt{ and } B, \Delta} \qquad \frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \texttt{ and } B \Rightarrow \Delta}$$

$$\frac{\Gamma \Rightarrow A, B, \Delta}{\Gamma \Rightarrow A \texttt{ or } B, \Delta} \qquad \frac{\Gamma, A \Rightarrow \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \texttt{ or } B \Rightarrow \Delta}$$

$$\frac{\Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \texttt{not } A, \Delta} \qquad \frac{\Gamma \Rightarrow A, \Delta}{\Gamma, \texttt{not } A \Rightarrow \Delta}$$

Percolation from Functors and through Abstraction

- $$\frac{\Gamma, X : AC \texttt{ and } BC}{\Gamma, X : (A \texttt{ and } B)C}$$
- $$\frac{\Gamma, X : (\lambda x.A) \texttt{ and } (\lambda x.B)}{\Gamma, X : (\lambda x.A \texttt{ and } B)}$$
- $+$ similar rules for `or` and `not`.
- What remains is percolation from arguments.

Percolation from Arguments

- No general rule, obviously. The possibility of percolation depends on properties of the functor.
- But many of the properties corresponding with syntactic phenomena seem related to Gentzen rules that allow percolation of Booleans.

Upward Monotonic Functors

- $$\frac{\Gamma, A \Rightarrow B, \Delta \qquad \Gamma, F \Rightarrow G, \Delta}{\Gamma, FA \Rightarrow GB, \Delta}$$ if F or G is mon$\uparrow$.
- Examples of mon$\uparrow$ functors: some, some N, every N, many N, most N.
- $$\frac{\texttt{horse} \Rightarrow \texttt{animal} \qquad \texttt{some} \Rightarrow \texttt{some}}{\texttt{some horse} \Rightarrow \texttt{some animal}}$$

Downward Monotonic Functors

- $$\frac{\Gamma, B \Rightarrow A, \Delta \qquad \Gamma, F \Rightarrow G, \Delta}{\Gamma, FA \Rightarrow GB, \Delta}$$ if F or G is mon$\downarrow$.
- Examples of mon$\downarrow$ functors: no, no N, every, few, few N.
- $$\frac{\dfrac{\texttt{sparrow} \Rightarrow \texttt{bird} \qquad \texttt{no} \Rightarrow \texttt{no}}{\texttt{no bird} \Rightarrow \texttt{no sparrow}} \qquad \texttt{flew} \Rightarrow \texttt{moved}}{\texttt{no bird moved} \Rightarrow \texttt{no sparrow flew}}$$

Derived Rules

- $$\frac{\Gamma, FA, FB \Rightarrow \Delta}{\Gamma, F(A \texttt{ and } B) \Rightarrow \Delta}$$ if F is mon$\uparrow$.
- $$\frac{\Gamma \Rightarrow FA, FB, \Delta}{\Gamma \Rightarrow F(A \texttt{ or } B), \Delta}$$ if F is mon$\uparrow$.
- $$\frac{\Gamma, FA, FB \Rightarrow \Delta}{\Gamma, F(A \texttt{ or } B) \Rightarrow \Delta}$$ if F is mon$\downarrow$.
- $$\frac{\Gamma \Rightarrow FA, FB, \Delta}{\Gamma \Rightarrow F(A \texttt{ and } B), \Delta}$$ if F is mon$\downarrow$.
- Allow us to get rid of `and` and `or` at least in some cases.

More Percolation Rules

- $$\frac{\Gamma \Rightarrow FA, \Delta \qquad \Gamma \Rightarrow FB, \Delta}{\Gamma \Rightarrow F(A \texttt{ or } B), \Delta}$$ if F is anti-additive.
- $$\frac{\Gamma \Rightarrow FA, \Delta \qquad \Gamma \Rightarrow FB, \Delta}{\Gamma \Rightarrow F(A \texttt{ and } B), \Delta}$$ if F has meet.

- The last property was taken from Van der Does (1992), in which a bunch of similar properties are discussed.
- no N and every N have meet.
- We can classify words in natural language according to which of these rules (and similar ones) may be applied to each of their arguments.

Rules for every and some

- To get things going we also need rules for the classical quantifiers. Here are some special ones.
- $$\frac{}{\Gamma, \texttt{every}\ AB, A \Rightarrow B, \Delta} \qquad \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow \texttt{every}\ AB, \Delta}$$
- $$\frac{}{\Gamma, A, B \Rightarrow \texttt{some}\ AB, \Delta} \qquad \frac{\Gamma \Rightarrow A, B, \Delta}{\Gamma, \texttt{some}\ AB \Rightarrow \Delta}$$
- In the right-hand rules Γ and Δ must be sets of sentences.

More Rules for every and some

- Further rules for these quantifiers are of the kind we have just met (monotonicity, having meet, splittingness—a dual of having meet),
- plus the conservativity rule shown on a previous slide,
- plus duality rules,
- plus Symmetry:

$$\frac{\Gamma, X : \texttt{some}\, B\, A}{\Gamma, X : \texttt{some}\, A\, B}$$

Conclusion

- In this talk I have reported on ongoing work with the aim of giving an analytic sequent calculus that is based on representations very close to natural language.
- Unlike the usual sequent calculi, the calculus necessarily must have many rules.
- Rules of the calculus are intimately connected with the distributional properties of certain items.
- Calculi like these lend themselves very easily to implementation.