Understanding Natural Language with Functional Discourse Grammar

Reinier Lamers

Universiteit Utrecht

February 13, 2009
Outline

1. Introduction
2. Relevance
3. Functional Discourse Grammar
4. The Algorithm
5. Demonstration
6. Conclusions
I am writing my thesis at the Intelligent Systems group.
They are designing a companion robot for the kitchen domain.
They’re using Philips’s iCat.
It has to understand language.
And I have to make the “understanding” part.
“Understanding” means here: “converting syntax to semantics”.

Syntax:

Semantics: $\forall x \exists y(\ldots)$

Why design it ourselves?
Relevance of Converting Syntax to Semantics

- **Speech**
- **Text**
- **Syntax**
- **Semantics**
- **Pragmatics**
- **Beliefs, Desires, Intentions**
A Short History of Functional Discourse Grammar

- FDG principles:
  - No transformations or filters
  - Discourse Act as basic unit
  - Single theory of language
  - Maximal Depth
  - Not a Theory of Discourse
The Levels of FDG

- FDG analyzes utterances at four *Levels*
- That has to do with the Single Theory Principle
- The four levels are the Phonological, Morphosyntactic, Representational and Interpersonal Levels
- Syntax to Semantics $\sim$ Morphosyntactic Level to Representational Level
Tree Notation for FDG Levels

- Levels consists of layers may contain other layers: *Tree Structure*
- How I draw them as trees:
Outline

A quick outline of the algorithm:
A quick outline of the algorithm:

1. Run through the Morphosyntactic Level tree and generate pieces of Representational Level tree
A quick outline of the algorithm:

1. Run through the Morphosyntactic Level tree and generate pieces of Representational Level tree
2. Try to glue the pieces together in a consistent way
Outline

A quick outline of the algorithm:

1. Run through the Morphosyntactic Level tree and generate pieces of Representational Level tree
2. Try to glue the pieces together in a consistent way
3. Fill in some references to other parts of the tree
A quick outline of the algorithm:

1. Run through the Morphosyntactic Level tree and generate pieces of Representational Level tree
2. Try to glue the pieces together in a consistent way
3. Fill in some references to other parts of the tree
4. Add Prolog’s backtracking to taste
The Algorithm

Treewalk I

- Run through tree in depth-first order
- For every node (excluding list or restrictor nodes), apply *node function* to subtree rooted in it
- Node function returns list of *tree chunks* - pieces of the Representational Level
- Result of treewalk procedure: ordered list of chunks
Lexicon:
- The node function often consults the *lexicon*
- The lexicon contains only truly lexical items
- It stores identifier, spelling, lexeme class, combination schemes and class-dependent properties

Why such a complex treewalk procedure?
- Not every utterance is as form-preserving as the previous example...
- Some features depend on co-occurrence (count/mass)
Composition

Those tree chunks are glued together during the *composition stage*.

Basic procedure:
1. Start with an empty tree
2. Add pieces to it as they fit, until there are no pieces any more
3. Is the tree complete? If yes, solution. Else, backtrack.

“Fitting” under point 2 above is defined by a mostly language-specific matching function. Universal bits for position, function, more?

Matching function compares criteria stored on the chunks. So node function and matching function cooperate.
Coreference resolution

- For resolving anaphora and vaguely similar things: the coreference resolution stage
- Some nodes marked as dangling references by node function, resolved now
- Only used for implicit subjects in implementation
Demonstration
Conclusion

- The computer program formalizes a linguistic theory that was not very rigorous.
- Using a three-step algorithm with backtracking, we can convert English syntax to semantics.
- It works!
Conclusion

- The computer program formalizes a linguistic theory that was not very rigorous
- Using a three-step algorithm with backtracking, we can convert English syntax to semantics
- It works!
- For a small subset of English at least
Discussion

- Will it scale to any linguistic phenomenon?
- Isn’t Functional Discourse Grammar too complex?
- How does this compare to the Categorial Grammar + Montague-like approach?
- Isn’t there an easier way to convert one tree to the other?