New Type-Theoretic Tools in Natural Language Semantics: Program Teaser

NASSLLI 2018

Carnegie Mellon University
Pittsburgh, Pennsylvania

June 27, 2018

Thanks to Our Guest Speakers!

Chris Barker (NYU)

Daisuke Bekki (Ochanomisu)

Dylan Bumford (UCLA)

Simon Charlow (Rutgers)

Stergios Chatzikyriakidis (Gothenburg)

Type-Theoretic Tools

1. Dependent Types
    - Tutorial by Zawadowski
    - Dynamic Semantics (Bekki, Grudzinska)
    - Common Nouns-as-Types, other applications (Chatzikyriakidis)
    - Computational Semantics (Chatzikyriakidis)
2. Monads and Comonads
    - Tutorial by Awodey
    - Monads
        - Dynamic Semantics,.... (Tutorial by Bumford and Charlow)
        - Quantifier Scope Ambiguities (Barker)
    - Comonads (Zwanziger)

Dependent Types: History

- Types which depend on terms of (other) types
- Dependent type theory developed by Per Martin-Löf, circa 1970's
- Applied to natural language by Sundholm (1986, 1989), Ranta (1994)
- Further work by Asher, Bekki, Chatzikyriakidis, Grudzinska et al., Luo, ... Now a community of researchers.

Dependent Types: Dynamic Semantics/Anaphora

Problem: Donkey Sentences

Consider the sentence: "Every farmer who owns a donkey beats it." What is its logical translation?

Pronouns are interpreted as variables, so "beats it" is interpreted as:

$$\lambda x(B(x,y))$$

Relative clauses are interpreted intersectionally, so "Every farmer who owns a donkey" is interpreted as:

$$\lambda P(\forall x(F(x) \wedge \exists y(D(y) \wedge O(x,y)) \rightarrow P(x)))$$

If we combine these, we get as the interpretation of the whole sentence:

$$\forall x(F(x) \wedge \exists y(D(y) \wedge O(x,y)) \rightarrow B(x,y)) \qquad (???)$$

Dependent Types: Dynamic Semantics/Anaphora, Continued

Problem: Donkey Sentences

Solution: Dependent Types!! (Sundholm 1986)

The advantages of dependent type theory in tracking anaphoric dependencies have been developed into a full-fledged approach to dynamic semantics.
Talks by Bekki and Grudzkinska on this topic!

Further Applications of Dependent Types

What are the other applications and particularities of the DTT framework?

- Common nouns-as-types: Like natural language, DTT has bounded quantification. But to exploit this, we must view common nouns as types, not predicates on a type of entities.
- Uses for adjectival and adverbial modification (Chatzikyriakidis and Luo 2014,...)
- Lexical Semantics (Asher 2011)
  Talk by Chatzikyriakidis on topics in dependently-typed semantics!

Dependent Types in Computational Semantics

- Dependent Types are implemented in interactive theorem provers (e.g. Coq, Agda, Lean)
- These provide ready-made tools for computational semantics Further talk by Chatzikyriakidis on using Coq for computational semantics!

Monads and Comonads

- Specific kinds of type operators (or endofunctors on a category)

  Arose in category theory (1958)
  Applied to type theory in 1990's (Moggi, Benton et al.,...)

- A monad (resp. comonad) T takes a type A to a type $T(A)$ of "enriched outputs" (resp. "inputs"). Using a monad or comonad allows us to enrich our semantics modularly on top of the compositional semantics.

Monads and Comonads: Background

- Monads applied to natural language in 2000's (from Barker 2000, Shan 2000)

  Dynamic Semantics
  Quantifier Scope ambiguities

- Comonads applied to natural language in 2010's (from Awodey et al. 2015)

  Intensionality (a la Montague)

Talks: Bumford and Charlow on monads in NL, Barker on current issues using monads to model scope ambiguities, and Zwanziger on intensionality!

Thanks!