Overall goal

Montagovian semantics for Computer Scientists, or Derivation calculators for Semanticists

Derivations and normalizations are boring, let the computer do it

Gains
  ▶ for NL researchers: a helpful tool
  ▶ for PL researchers: an interesting application to build tools for

Beginning of a beautiful friendship
(or, collaboration, or at least mutual comprehension)

http://okmij.org/ftp/gengo/NASSLLI10/
Grand goal

NL researchers will
▶ gain rational reconstruction of Montagovian tricks
▶ import developed CS ideas:
  side effects, continuations, regions, staging, dependent types

PL researchers will
▶ export developed CS ideas:
  side effects, continuations, regions, staging, dependent types
▶ build theories of programming language competence

All would benefit from connections with logic and probability theory
Plan

June 18

- Making (intuitive) sense of our metalanguage (Haskell)
- CFG: writing and (re-)interpreting derivations
  overall: how to embed (object) languages and represent
  (grammar/type) derivations

June 19 Denotations and truth conditions: LLF

- Propositional logic
- STLC (STT)
- Simplifying formulas:
  teaching computer simple logical inferences

June 20

- Simple language fragments and interpreters
- Quantifiers, in two ways
- Question: quantifiers and scope ambiguity
Plan, cont

June 21

▷ Pronouns. Donkey anaphora
▷ Dynamic semantics: sentence as an imperative program
▷ *Extending* previous language fragments, interpreters and STT to account for information “update”
▷ A compositional semantics of donkey anaphora

June 22

▷ Scope and inverse linking in continuation semantics
Main ideas

- *Calculemus*: yields, denotations
- *Many* fragments, languages, interpretations
- Growing fragments and languages
- Interactivity
- Montagovian tradition
The look of Haskell

- GHCI prompt
- Arithmetic, Logic, Strings
- Abstractions and applications
- Types, type annotations, type errors
- Definitions, parametrized definitions
Exercises 1

twice = \ f \rightarrow \ x \rightarrow f\ (f\ x)

▶ How else we can write this definition?
▶ Does this term reminds us something from lambda-calculus?
▶ How to quickly verify that?
Exercises 2

1. Write Church numeral for 0
2. Write increment incr. How to test it?
3. Write addition, multiplication, exponentiation, decrement
Further look at Haskell

Pairs (products)
introduction, elimination, pattern-matching in definitions

Sums (co-products)
introduction, elimination, defining by clauses

Why pairs are called products and why Either is called a sum or a co-product?

Polymorphic types
Exercises 3

Write functions of these types:

\[
\begin{align*}
((), a) & \rightarrow a \\
 a & \rightarrow ((), a) \\
\text{Either } a \ b & \rightarrow (a \rightarrow c) \rightarrow (b \rightarrow c) \rightarrow c \\
((a, b) & \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c) \\
(a \rightarrow b \rightarrow c) & \rightarrow ((a, b) \rightarrow c) \\
a & \rightarrow ((a \rightarrow f) \rightarrow f) \\
(((a \rightarrow f) \rightarrow f) & \rightarrow f) \rightarrow (a \rightarrow f) \\
(\text{Either } a \ b & \rightarrow f) \rightarrow (a \rightarrow f, \ b \rightarrow f) \\
((a, b) & \rightarrow f) \rightarrow (((\text{Either } (a \rightarrow f) (b \rightarrow f)) \rightarrow f) \rightarrow f)
\end{align*}
\]

- what do these functions do?
- What do these types remind you of?
- What do the terms your wrote signify?
Exercises 4

1. How polymorphic types relate to universals?
2. Why existentials in Haskell look the way they do?
Exercises 5

1. Define the data type of Pizzas
   The datatype describes which baked thing can be considered a pizza and which cannot.

2. Define a data type for burrito
Think about representing the derivation of, and computing yield and truth values of two sample sentences from the Semantics boot camp:

- Rick Perry is conservative
- Rick Perry is in Texas
Map

Symantics

Quantifier

Pronoun

Lambda

States

Dynamics

Sem

Sem

Sem

Sem

P

R

C

D

P

P