Towards a proof-theoretic natural language semantics for wide-coverage grammars

Koji Mineshima
Ochanomizu University

New Landscapes in Theoretical Computational Linguistics
Ohio State University
October 15, 2016

Collaborators

- Daisuke Bekki
- Pascual Martínez-Gómez
- Ribeka Tanaka
- Yusuke Miyao

- With funding from the JST CREST program, “Establishment of Knowledge-Intensive Structural Natural Language Processing and Construction of Knowledge Infrastructure”

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Two faces of this research

1. Applications to mainstream computational linguistics and NLP
2. New methodology for theoretical linguistics
   - Implementation of syntax-semantics interface
   - with statistical, corpus-based wide-coverage parsers
   - from truth-condition (Tarski) to proof-condition (Gentzen): "Proof-theoretic turn"
   - inferences-as-tests: empirical evaluation of semantic theories in terms of textual entailment (Bekki and Mineshima, 2016)
   - Importance of (automated) theorem proving: implementation of syntax-semantics-prover interface
   - Establishing a "benchmark" for formal semantics

Why “computational” theoretical linguistics?

- Computational modelling is needed to specify detailed syntactic and semantic architectures.

Why “computational” theoretical linguistics?

- Computational modelling is needed to specify detailed syntactic and semantic architectures.
- making it possible to compute the predictions of each syntactic/semantic framework quickly and precisely.

Why “computational” theoretical linguistics?

- Computational modelling is needed to specify detailed syntactic and semantic architectures.
- making it possible to compute the predictions of each syntactic/semantic framework quickly and precisely.
- A necessary step towards establishing a meaningful and systematic way to compare/evaluate each framework.

Why “computational” theoretical linguistics?

- Computational modelling is needed to specify detailed syntactic and semantic architectures.
- making it possible to compute the predictions of each syntactic/semantic framework quickly and precisely.
- A necessary step towards establishing a meaningful and systematic way to compare/evaluate each framework.
- ... although it can be very frustrating

> *I know from experience that in order to write a complete fragment, you normally have to include decisions about things for which you have no principled basis to choose, or for which you know that none of the available alternatives are really good. (Partee 2005)*

Implemented syntax-semantics-prover interface

Wide-coverage "formal semantics" parsers

- Bos et al. (2004): Boxer/Nutcraker
  CCG + DRT + FOL prover/model builder (English)
- Moot (2010):
  TLG + DRT (French)
- Butler and Yoshimoto (2012):
  SCT + Treebank Semantics (English and Japanese)
- Abzianidze (2015):
  CCG + Natural Logic Tableau prover (English)
- Mineshima et al. (2015)
  CCG + HOL prover (English and Japanese)

Benchmarking

- To test the capacity of formal semantics systems, we would also need a shared dataset ("benchmark").

Benchmarking

- To test the capacity of formal semantics systems, we would also need a shared dataset (“benchmark”).
- Cf. the TPTP library in Automated Theorem Proving:
  - a comprehensive and up-to-date list of problems
  - a guideline for adding new problems and for correcting errors in existing problems
  - competition for evaluating system performance

Benchmarking

- To test the capacity of formal semantics systems, we would also need a shared dataset ("benchmark").
- Cf. the TPTP library in Automated Theorem Proving:
    - a comprehensive and up-to-date list of problems
    - a guideline for adding new problems and for correcting errors in existing problems
    - competition for evaluating system performance
- Unshared Task at LENLS 13 : Theory and System analysis with FraCaS, MultiFraCaS and JSeM Test Suites (November 13, NINJAL, Japan)
  `http://www.compling.jp/fracas_task/index.html`

The framework

Our favorite theories:

- Syntax: CCG (wide-coverage parsers are available for English and Japanese)
- Semantics: DTS (dynamic/underspecification semantics smoothly combined with modern categorial grammar and prover)
- Prover: HOL (but it's almost first-order; it's better than first-order reification)

The framework

Our favorite theories:

- Syntax: CCG (wide-coverage parsers are available for English and Japanese)
- Semantics: DTS (dynamic/underspecification semantics smoothly combined with modern categorial grammar and prover)
- Prover: HOL (but it's almost first-order; it's better than first-order reification)

Three steps:

1. Build a flexible platform to implement the syntax-semantics-prover interface: ccg2lambda (CCG + HOL)
2. Adding a robust external knowledge base
3. Extend it with underspecification semantics (@-terms) to handle dynamics (anaphora/presupposition): ccg2lambda + DTS

Plan

1. Introduction
2. ccg2lambda
3. Experiments on RTE datasets
4. Dependent Type Semantics

ccg2lambda

Compositional semantics and higher-order inference system for wide-coverage CCG parsers

Natural Language Inferences (Textual Entailment)

- Does Premise P entail Hypothesis H?

P Smoking in restaurants is prohibited by law in most cities in Japan.
H Smoking in public spaces is not allowed in some cities.

Yes (Entailment)

Natural Language Inferences (Textual Entailment)

- Does Premise P entail Hypothesis H?

P Smoking in restaurants is prohibited by law in most cities in Japan.
H Smoking in public spaces is not allowed in some cities.

Yes (Entailment)

- *The best way of testing an NLP system's semantic capacity* (Cooper et al. 1996)
- Many applications (Question Answering, Machine Translation, etc.)

Natural Language Inferences (Textual Entailment)

- Does Premise P entail Hypothesis H?

P Smoking in restaurants is prohibited by law in most cities in Japan.
H Smoking in public spaces is not allowed in some cities.

Yes (Entailment)

- *The best way of testing an NLP system's semantic capacity* (Cooper et al. 1996)
- Many applications (Question Answering, Machine Translation, etc.)

- relevant factors:
  1. syntax
  2. logical words: *most*, *not*, *some*, *every*
  3. content words:
     restaurant $\rightarrow$ *public_space*
     prohibited $\rightarrow$ $\neg$ *allowed*

Formal Semantics

Logic-based approaches to entailment

Natural Logic	First-order logic (FOL)	Higher-order logic (HOL)
• formalizes inferences with surface form ▲ only allows single premise inferences (mononicity inference)	• efficient provers exist • dominate computational linguistics ▲ limited expressive power	• high expressive power • dominate formal semantic ▲ no general-purpose efficient prover exists
more efficient less expressive ⟷		⟷ less efficient more expressive
MacCartney (2009)	Boxer (Bos 2008)	Our approach

Logic-based approaches to entailment

Natural Logic	First-order logic (FOL)	Higher-order logic (HOL)
• formalizes inferences with surface form ▲ only allows single premise inferences (mononicity inference)	• efficient provers exist • dominate computational linguistics ▲ limited expressive power	• high expressive power • dominate formal semantic ▲ no general-purpose efficient prover exists
more efficient less expressive	⟷	less efficient more expressive
MacCartney (2009)	Boxer (Bos 2008)	Our approach

Logic-based approaches to entailment

Natural Logic	First-order logic (FOL)	Higher-order logic (HOL)
• formalizes inferences with surface form ▲ only allows single premise inferences (mononicity inference)	• efficient provers exist • dominate computational linguistics ▲ limited expressive power	• high expressive power • dominate formal semantic ▲ no general-purpose efficient prover exists
more efficient less expressive	⟷	less efficient more expressive
MacCartney (2009)	Boxer (Bos 2008)	Our approach

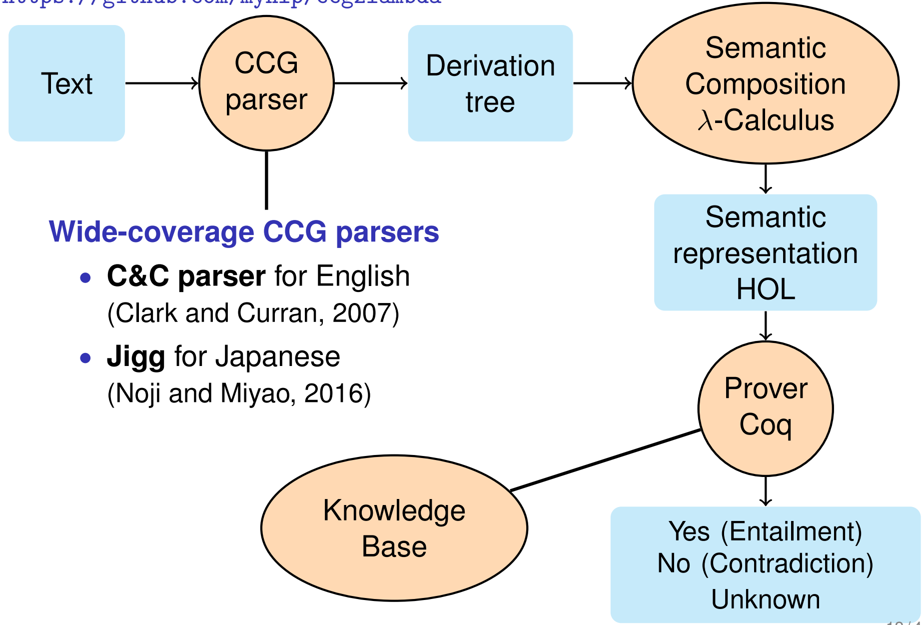
Logic-based approaches to entailment

Natural Logic	First-order logic (FOL)	Higher-order logic (HOL)
• formalizes inferences with surface form ▲ only allows single premise inferences (mononicity inference)	• efficient provers exist • dominate computational linguistics ▲ limited expressive power	• high expressive power • dominate formal semantic ▲ no general-purpose efficient prover exists
more efficient less expressive	⟷	less efficient more expressive
MacCartney (2009)	Boxer (Bos 2008)	Our approach

Goal: To develop a higher-order inference system specialized for natural language inferences, and combine it with a wide-coverage parser

Higher-order inference system: ccg2lambda

https://github.com/mynlp/ccg2lambda



Wide-coverage CCG parsers

- **C&C parser** for English (Clark and Curran, 2007)
- **Jigg** for Japanese (Noji and Miyao, 2016)

CCG-based Compositional Semantics

Basic category: S, NP, N
Function category: $X/Y, X\backslash Y$
Combinatory rules in CCG:

$$\frac{X/Y : f \quad Y : a}{X : fa} > \qquad \frac{Y : a \quad X\backslash Y : f}{X : fa} <$$

$$\begin{array}{c}
\dfrac{\dfrac{\texttt{John}}{NP :} \qquad \dfrac{\dfrac{\texttt{likes}}{(S\backslash NP)/NP :} \quad \dfrac{\texttt{Mary}}{NP :}}{S\backslash NP :}>}{S :}<
\end{array}$$

CCG-based Compositional Semantics

Basic category: S, NP, N
Function category: $X/Y, X\backslash Y$
Combinatory rules in CCG:

$$\dfrac{X/Y : f \quad Y : a}{X : fa}\, > \qquad \dfrac{Y : a \quad X\backslash Y : f}{X : fa}\, <$$

$$\dfrac{\dfrac{\texttt{John}}{NP : \mathrm{john}} \qquad \dfrac{\dfrac{\texttt{likes}}{(S\backslash NP)/NP : \lambda y.\lambda x.\textbf{like}\,(x, y)} \quad \dfrac{\texttt{Mary}}{NP : \mathrm{mary}}}{S\backslash NP :}\, >}{S :}\, <$$

CCG-based Compositional Semantics

Basic category: S, NP, N
Function category: $X/Y, X\backslash Y$
Combinatory rules in CCG:

$$\frac{X/Y : f \quad Y : a}{X : fa} > \qquad \frac{Y : a \quad X\backslash Y : f}{X : fa} <$$

$$\dfrac{\dfrac{\texttt{John}}{NP : \text{john}} \quad \dfrac{\dfrac{\texttt{likes}}{(S\backslash NP)/NP : \lambda y.\lambda x.\textbf{like}(x, y)} \quad \dfrac{\texttt{Mary}}{NP : \text{mary}}}{S\backslash NP : \lambda x.\textbf{like}(x, \text{mary})} >}{S :} <$$

CCG-based Compositional Semantics

Basic category: S, NP, N
Function category: $X/Y, X\backslash Y$
Combinatory rules in CCG:

$$\frac{X/Y : f \quad Y : a}{X : fa} > \qquad \frac{Y : a \quad X\backslash Y : f}{X : fa} <$$

$$\dfrac{\dfrac{\texttt{John}}{NP : \text{john}} \quad \dfrac{\dfrac{\texttt{likes}}{(S\backslash NP)/NP : \lambda y.\lambda x.\textbf{like}(x, y)} \quad \dfrac{\texttt{Mary}}{NP : \text{mary}}}{S\backslash NP : \lambda x.\textbf{like}(x, \text{mary})} >}{S : \textbf{like}(\text{john}, \text{mary})} <$$

CCG-based Compositional Semantics

Basic category: S, NP, N
Function category: $X/Y, X\backslash Y$
Combinatory rules in CCG:

$$\frac{X/Y : f \quad Y : a}{X : fa} > \qquad \frac{Y : a \quad X\backslash Y : f}{X : fa} <$$

$$\dfrac{\dfrac{\texttt{John}}{NP : \text{john}} \quad \dfrac{\dfrac{\texttt{likes}}{(S\backslash NP)/NP : \lambda y.\lambda x.\textbf{like}(x, y)} \quad \dfrac{\texttt{Mary}}{NP : \text{mary}}}{S\backslash NP : \lambda x.\textbf{like}(x, \text{mary})} >}{S : \textbf{like}(\text{john}, \text{mary})} <$$

Quantifier: Standard analysis (Steedman, 2000)

$$\dfrac{\dfrac{\dfrac{\texttt{every}}{NP/N : \lambda F \lambda G.\forall x(F(x) \to G(x))} \quad \dfrac{\texttt{student}}{N : \lambda x.\mathbf{student}(x)}}{NP : \lambda G.\forall x(\mathbf{student}(x) \to G(x))}> \quad \dfrac{\texttt{runs}}{S\backslash NP : \lambda x.\mathbf{run}(x)}}{S : \forall x(\mathbf{student}(x) \to \mathbf{run}(x))}<$$

Quantifier: Standard analysis (Steedman, 2000)

$$
\dfrac{
  \dfrac{\texttt{every}}{NP/N \;:\; \lambda F \lambda G.\forall x(F(x) \rightarrow G(x))}
  \quad
  \dfrac{\texttt{student}}{N \;:\; \lambda x.\mathbf{student}(x)}
}{
  \color{red}{S/(S\backslash NP)} \;:\; \lambda G.\forall x(\mathbf{student}(x) \rightarrow G(x))
}{>}
\quad
\dfrac{\texttt{runs}}{S\backslash NP \;:\; \lambda x.\mathbf{run}(x)}
$$

$$
\dfrac{S/(S\backslash NP) \;:\; \lambda G.\forall x(\mathbf{student}(x) \rightarrow G(x)) \qquad S\backslash NP \;:\; \lambda x.\mathbf{run}(x)}{S \;:\; \forall x(\mathbf{student}(x) \rightarrow \mathbf{run}(x))}{<}
$$

Quantifier: Standard analysis (Steedman, 2000)

$$
\dfrac{\dfrac{\dfrac{\texttt{every}}{\begin{array}{c}(S/(S\backslash NP))/N \\ : \lambda F \lambda G.\forall x(F(x) \to G(x))\end{array}} \quad \dfrac{\texttt{student}}{\begin{array}{c}N \\ : \lambda x.\mathbf{student}\,(x)\end{array}}}{\begin{array}{c}S/(S\backslash NP) \\ : \lambda G.\forall x(\mathbf{student}\,(x) \to G(x))\end{array}}> \quad \dfrac{\texttt{runs}}{\begin{array}{c}S\backslash NP \\ : \lambda x.\mathbf{run}\,(x)\end{array}}}{\begin{array}{c}S \\ : \forall x(\mathbf{student}\,(x) \to \mathbf{run}\,(x))\end{array}}<
$$

Quantifier: Standard analysis (Steedman, 2000)

$$\dfrac{\dfrac{\dfrac{\texttt{every}}{\begin{array}{c}(S/(S\backslash NP))/N \\ : \lambda F \lambda G.\forall x(F(x) \to G(x))\end{array}} \quad \dfrac{\texttt{student}}{\begin{array}{c}N \\ : \lambda x.\mathbf{student}\,(x)\end{array}}}{\begin{array}{c}S/(S\backslash NP) \\ : \lambda G.\forall x(\mathbf{student}\,(x) \to G(x))\end{array}}> \quad \dfrac{\texttt{runs}}{\begin{array}{c}S\backslash NP \\ : \lambda x.\mathbf{run}\,(x)\end{array}}}{\begin{array}{c}S \\ : \forall x(\mathbf{student}\,(x) \to \mathbf{run}\,(x))\end{array}}<$$

Output of the C&C parser

$$\dfrac{\dfrac{\dfrac{\texttt{every}}{\begin{array}{c}NP/N \\ : \lambda F.\lambda G.\forall x(F(x) \to G(x))\end{array}} \quad \dfrac{\texttt{student}}{\begin{array}{c}N \\ : \lambda x.\mathbf{student}\,(x)\end{array}}}{\begin{array}{c}NP \\ : \lambda G.\forall x(\mathbf{student}\,(x) \to G(x))\end{array}}> \quad \dfrac{\texttt{run}}{\begin{array}{c}S\backslash NP \\ : \lambda x.\mathbf{run}\,(x)\end{array}}}{\begin{array}{c}S \\ : \forall x(\mathbf{student}\,(x) \to \mathbf{run}\,(x))\end{array}}<$$

Quantifier: Standard analysis (Steedman, 2000)

$$
\dfrac{\dfrac{\dfrac{\texttt{every}}{(S/(S\backslash NP))/N \; : \lambda F \lambda G. \forall x(F(x) \to G(x))} \quad \dfrac{\texttt{student}}{N \; : \lambda x. \mathbf{student}(x)}}{S/(S\backslash NP) \; : \lambda G. \forall x(\mathbf{student}(x) \to G(x))}> \quad \dfrac{\texttt{runs}}{S\backslash NP \; : \lambda x. \mathbf{run}(x)}}{S \; : \forall x(\mathbf{student}(x) \to \mathbf{run}(x))}<
$$

Output of the C&C parser

$$
\dfrac{\dfrac{\dfrac{\texttt{every}}{NP/N \; : \lambda F. \lambda G. \forall x(F(x) \to G(x))} \quad \dfrac{\texttt{student}}{N \; : \lambda x. \mathbf{student}(x)}}{NP \; : \lambda G. \forall x(\mathbf{student}(x) \to G(x))}> \quad \dfrac{\texttt{run}}{S\backslash NP \; : \lambda Q. Q(\lambda x. \mathbf{run}(x))}}{S \; : \forall x(\mathbf{student}(x) \to \mathbf{run}(x))}<
$$

Continuation-based semantics of post NP-modifiers

- The standard analysis of NP-modifiers

Every	+	[student	+	who works]
NP/N		N		$N\backslash N$
$\lambda F \lambda G. \forall x(Fx \rightarrow Gx)$		$\lambda x.(\text{student}(x) \wedge \text{work}(x))$		

Continuation-based semantics of post NP-modifiers

- The standard analysis of NP-modifiers

$$\underset{\substack{NP/N \\ \lambda F \lambda G.\forall x(Fx \rightarrow Gx)}}{\text{Every}} + \boxed{[\ \underset{N}{\text{student}} + \underset{N\backslash N}{\text{who works}}\ ]}$$

$$\lambda x.(\text{student}(x) \wedge \text{work}(x))$$

- The C&C parser (NP-S analysis)

$$\boxed{[\ \underset{NP/N}{\text{Every}} + \underset{N}{\text{student}}\ ]} + \underset{NP\backslash NP}{\text{who works}}$$

Continuation-based semantics of post NP-modifiers

- The standard analysis of NP-modifiers

$$\begin{array}{ccc} \text{Every} & + & [\ \text{student} \ + \ \text{who works}\ ] \\ NP/N & N & N\backslash N \\ \lambda F \lambda G.\forall x(Fx \rightarrow Gx) & \multicolumn{2}{c}{\lambda x.(\text{student}(x) \wedge \text{work}(x))} \end{array}$$

- The C&C parser (NP-S analysis)

$$\begin{array}{ccc} [\ \text{Every} \ + \ \text{student}\ ] & + & \text{who works} \\ NP/N \quad N & & NP\backslash NP \end{array}$$

- Boxer's output for "Every student who works comes."

$$\forall x(\text{student}(x) \rightarrow \text{work}(x) \wedge \text{come}(x))$$

Continuation-based semantics of post NP-modifiers

- Solution: NPs have an extra argument position

  [Every + student] + who works

  NP/N $\quad N \quad$ $NP\backslash NP$

  $\lambda F \lambda G.\forall x(\text{student}(x) \wedge Fx \to Gx)$ $\quad$ $\lambda Q \lambda F \lambda G.Q(\lambda x.(\text{work}(x) \wedge Fx))(G)$

  Cf. Bach and Cooper's (1979) NP-S analysis; Champollion's (2014) event semantics

Continuation-based semantics of post NP-modifiers

- Solution: NPs have an extra argument position

  [Every + student] + who works

  NP/N $\quad N \quad$ $NP\backslash NP$

  $\lambda F \lambda G.\forall x(\text{student}(x) \wedge Fx \rightarrow Gx)$ $\quad$ $\lambda Q \lambda F \lambda G.Q(\lambda x.(\text{work}(x) \wedge Fx))(G)$

  Cf. Bach and Cooper's (1979) NP-S analysis; Champollion's (2014) event semantics

- VPs reset the extra argument

  Every student who works + comes

  NP $\quad$ $NP\backslash S$

  $\lambda F \lambda G.\forall x(\text{student}(x) \wedge \text{work}(x) \wedge Fx \rightarrow Gx)$ $\quad$ $\lambda Q.Q(\lambda x.\text{True})(\lambda x.\text{come}(x))$

- Output: $\forall x(\text{student}(x) \wedge \text{work}(x) \wedge \text{True} \rightarrow \text{come}(x))$

Continuation-based semantics of post NP-modifiers

- Solution: NPs have an extra argument position

[Every	+	student]	+	who works
NP/N		N		$NP\backslash NP$
$\lambda F \lambda G.\forall x(\text{student}(x) \wedge Fx \to Gx)$				$\lambda Q \lambda F \lambda G.Q(\lambda x.(\text{work}(x) \wedge Fx))(G)$

Cf. Bach and Cooper's (1979) NP-S analysis; Champollion's (2014) event semantics

- VPs reset the extra argument

Every student who works	+	comes
NP		$NP\backslash S$
$\lambda F \lambda G.\forall x(\text{student}(x) \wedge \text{work}(x) \wedge Fx \to Gx)$		$\lambda Q.Q(\lambda x.\text{True})(\lambda x.\text{come}(x))$

- Output: $\forall x(\text{student}(x) \wedge \text{work}(x) \wedge \text{True} \to \text{come}(x))$
- Advantages:
  1. Bare quantifiers: *everyone in the room*
  2. Non-restrictive relative clauses: *John, who is the president, ...*
  3. NP-modifers in Japanese:

     hataraku subete-no gakusei (*every student who works*)

     works every student

HOL as representation language

1. Generalized quantifiers
   Most students work $\rightsquigarrow$ $\mathrm{most}(\lambda.\mathrm{student}(x), \lambda x.\mathrm{work}(x))$
2. Modals
   John might come $\rightsquigarrow$ $\mathrm{might}(\mathrm{come}(j))$
3. Veridical and anti-veridical predicates
   Someone managed to come $\rightsquigarrow$ $\exists x(\mathrm{manage}(x, \mathrm{come}(x)))$
   Someone failed to come $\rightsquigarrow$ $\exists x(\mathrm{fail}(x, \mathrm{come}(x)))$
4. Intensional (privative) adjectives
   John is a former student $\rightsquigarrow$ $\mathrm{former}(\mathrm{john}, \lambda x.\mathrm{student}(x))$
5. Attitude verbs
   John knows that some student came. $\rightsquigarrow$
   $$\mathrm{know}(\mathrm{j}, \exists x(\mathrm{student}(x) \wedge \mathrm{come}(x)))$$

- Alternative: FOL decomposition/reification (Hobbs, 1985)
  $$\forall w_1 (R_{\mathrm{john}}\, w_0\, w_1 \rightarrow \exists x(\mathrm{student}(w_1, x) \wedge \mathrm{come}(w_1, x)))$$

HOL as representation language

E : type of entities
Prop : type of propositions

Examples	**Semantic Types**
most	$(E \rightarrow Prop) \rightarrow (E \rightarrow Prop) \rightarrow Prop$
might	$Prop \rightarrow Prop$
manage	$Prop \rightarrow E \rightarrow Prop$
former	$(E \rightarrow Prop) \rightarrow (E \rightarrow Prop)$
know	$Prop \rightarrow E \rightarrow Prop$

Using Coq as HOL theorem prover

Proof Assistant Coq

- Inferences in the style of natural deduction
- Rich language (Dependent Type Theory)
- Powerful automation (Tactics)
  - Built-in tactics for first-order inferences and equational reasoning
  - User-defined tactics for higher-order inferences (Ltac)

Axioms for non-first-order constructions

Inference pattern	**Axiom**
Existential import	$\forall F\,\forall G\,(\text{most}(F,G) \rightarrow \exists x(Fx \wedge Gx))$
Conservativity	$\forall F\,\forall G\,(\text{most}(F,G) \rightarrow \text{most}(F,\lambda x.(Fx \wedge Gx)))$
Monotonicity (right-upward)	$\forall F\,\forall G\,\forall H\,(\text{most}(F,G) \rightarrow (\forall x(Gx \rightarrow Hx) \rightarrow \text{most}(F,H)))$
Veridicality	$\forall x \forall P(\text{manage}(x,P) \rightarrow P)$ $\forall x \forall P(\text{know}(x,P) \rightarrow P)$
Anti-veridicality	$\forall x \forall P(\text{fail}(x,P) \rightarrow \neg P)$

Lexical entries

1. For closed words: lexical entries directly assigned to surface form: 129 entries

Example

- category: NP/N
- semantics: $\lambda F \lambda G \lambda H. \forall x(Fx \wedge Gx \rightarrow H)$
- surf: every

2. For open words: schematic lexical entry (semantic templates) assigned to syntactic categories: 100 entries

Example

- category: N
- semantics: $\lambda x.\mathrm{E}(x)$

2. Experiments on FraCaS and JSeM

Experiment 1: FraCaS

- The FraCaS test suite (Cooper et al., 1994): the textual inference problems to test theories of formal and computational semantics
- Most problems do not require lexical/world knowledge, but contain linguistically challenging problems.
- Three types of answer: yes, no, unknown
- Single-premise problems (55%) and multiple-premise problems (45%)

Example

fracas-026 answer: yes (the premises entail the hypothesis)
P1 Most Europeans are resident in Europe.
P2 All Europeans are people.
P3 All people who are resident in Europe can travel freely within Europe.
H Most Europeans can travel freely within Europe.

fracas-038 answer: no (the premise contradicts the hypothesis)
P1 No delegate finished the report.
H Some delegate finished the report on time.

Results (Mineshima et al. 2015)

- Nutcracker = C&C parser + Boxer* + FOL prover (*bliksem*)
  + FOL model-builder (*mice*) + WordNet

Accuracy

Section	#	Ours	Nut	L & S 13	Tian 14
Quantifiers	74	.77	.53	.62	.80
Plurals	33	.67	.52	–	–
Adjectives	22	.68	.32	–	–
Comparatives	31	.48	.45	–	–
Verbs	8	.62	.62	–	–
Attitudes	13	.77	.46	–	–
Total	181	.69*	.50	–	–

* Total accuracy drops to 59% when ablating the higher-order rules

* 30 seconds time-out (after that, the system outputs "unknown")

- L &S 13 = Lewis and Steedman (2013): CCG + FOL prover
- Tian 14 = Tian et al. (2014): DCS-based inference system

Speed

Parsing and inference	sec./problem
CCG Parsing (C &C parser)	3.76
Our system with higher-order inference	3.72
Our system w/o higher-order inference	3.46
Nutcracker with first-order inference (first-order prover + model builder)	11.23

Why not reach 100% accuracy?

A reviewer: why not reach an accuracy close to 100% if tuned on a test set of only about 100 examples?

- syntactic parse error (no output CCG tree)
- disambiguation
  - syntactic (eg. PP-attachment)
  - semantic (eg. negation scope, collective vs. distributive)
- lack of lexical semantics (eg. verb aspect classification)
- the existing analyses are not good:
  - comparative/superlative
  - numerals
- context-dependency (anaphora and ellipsis)
- gold labels are wrong/problematic

Why not reach 100% accuracy?

A reviewer: why not reach an accuracy close to 100% if tuned on a test set of only about 100 examples?

- syntactic parse error (no output CCG tree)
- disambiguation
  - syntactic (eg. PP-attachment)
  - semantic (eg. negation scope, collective vs. distributive)
- lack of lexical semantics (eg. verb aspect classification)
- the existing analyses are not good:
  - comparative/superlative
  - numerals
- context-dependency (anaphora and ellipsis)
- gold labels are wrong/problematic

Fixing parse/disambiguation errors improved the accuracy:

- combining two CCG parsers (C&C + EasyCCG):
- Accuracy of Quantifier Sec. : 77% to 95%

ccg2lambda for Japanese

- Japanese CCGbank (Uematsu et al., 2015)
  - Base category : only S and NP
  - a few non-lexical type-shifting rules:
    eg. the relativizer from $S\backslash NP$ to NP/NP
- Japanese CCG parser Jigg (Noji and Miyao, 2016)
- Size of semantic lexicon:

	Japanese	English
Semantics templates for syntactic categories	37	129
Lexical entries for particular lexical items	113	100

Adding Neo-Davidsonian Event Semantics

Syntactic category	Semantic representation
NP	$\lambda NF.\exists x(N(\mathrm{Base}, x) \wedge F(x))$
$S\backslash NP_{ga}$	$\lambda QK.Q(\lambda I.I, \lambda x.\exists v(K(\mathrm{Base}, v) \wedge (\mathbf{Nom}\,(v) = x)))$
$S\backslash NP_{ga}\backslash NP_o$	$\lambda Q_2 Q_1 K.Q_1(\lambda I.I, \lambda x_1.Q_2(\lambda I.I, \lambda x_2.\exists v(K(\mathrm{Base}, v)$ $\wedge\, (\mathbf{Nom}\,(v) = x_1) \wedge (\mathbf{Acc}\,(v) = x_2)))))$
S/S	$\lambda SK.S(\lambda Jv.K(\lambda v'.(J(v') \wedge \mathrm{Base}(v')), v))$
NP/NP	$\lambda QNF.Q(\lambda Gx.N(\lambda y.(\mathrm{Base}(y) \wedge G(y)), x), F)$

Semantic types

$$\mathrm{T} \ ::= \ \mathrm{E} \mid \mathrm{Ev} \mid \mathrm{Prop} \mid \mathrm{T}_1 \Rightarrow \mathrm{T}_2$$

Mapping from syntactic categories to semantic types

$$\begin{aligned} NP^{\bullet} &= ((\mathrm{E} \Rightarrow \mathrm{Prop}) \Rightarrow \mathrm{E} \Rightarrow \mathrm{Prop}) \Rightarrow (\mathrm{E} \Rightarrow \mathrm{Prop}) \Rightarrow \mathrm{Prop} \\ S^{\bullet} &= ((\mathrm{Ev} \Rightarrow \mathrm{Prop}) \Rightarrow \mathrm{Ev} \Rightarrow \mathrm{Prop}) \Rightarrow \mathrm{Prop} \\ (C1/C2)^{\bullet} &= (C1\backslash C2)^{\bullet} = C2^{\bullet} \Rightarrow C1^{\bullet} \end{aligned}$$

Experiment 2 : JSeM

- Dataset: Japanese Semantics Test Suite (JSeM) (Kawazoe et al., 2015)
  `http://researchmap.jp/community-inf/JSeM/`
- Two separate parts
  - translation of English FraCaS Problems
  - a set of problems specific to Japanese syntax/semantics
- each problem is tagged with:
  - phenomena type (quantifier, adjective, negation, etc.)
  - inference type (logical entailment, presupposition)
- answer labels: yes, no, unknown
- single-premised problems (66%) and multi-premised problems (34%)

Results : Mineshima et al. (2016)

- five phenomena: quantifier, plural, adjective, verb, and attitude
- inference type: logical entailment
- Distribution of gold answer labels: yes 297, no 50, unknown 176
- Morphological analyzer: Kuromoji

Section	#Problem	Gold	System	SLC
Quantifier	337	**92.3**	78.0	88.4
Plural	41	**68.3**	56.1	51.2
Adjective	65	**67.7**	63.1	44.6
Verb	36	**77.8**	75.0	55.5
Attitude	44	**88.6**	86.4	75.0
Total	523	**86.0**	75.0	76.7

- Plain : system syntactic parse (1-best)
- Gold : gold syntactic parse (manually selected from *n*-best parses)
- SLC: supervised learning classifier (NTCIR RITE baseline tool)
  `http://www.cl.ecei.tohoku.ac.jp/rite2/doku.php`

3. ccg2lambda + Dependent Type Semantics

Dependent Type Semantics (DTS)

- ESSLLI2016 course:
  Daisuke Bekki and Koji Mineshima
  An Introduction to Dependent Type Semantics
  `http://esslli2016.unibz.it/?page_id=216`

Notation

	Martin-Löf	Logic	Agda	DTS
Π-type	$(\Pi x : A)\, B$	$\forall x : A.\, B$	$(x : A) \to B$	$(x : A) \to B$
Σ-type	$(\Sigma x : A)\, B$	$\exists x : A.\, B$	$(x : A) \times B$	$\begin{bmatrix} x : A \\ B \end{bmatrix}$
Function-type Implication	$A \to B$	$A \to B$	$A \to B$	$A \to B$
Product-type Conjunction	$A \wedge B$	$A \wedge B$	$A \times B$	$\begin{bmatrix} A \\ B \end{bmatrix}$

Dependent Types

1. Σ and Π types: generalization of $\wedge$ and $\rightarrow$

   $$\exists x.\, \mathrm{walk}\, x \;\rightsquigarrow\; \begin{bmatrix} x : \mathrm{entity} \\ \mathrm{walk}(x) \end{bmatrix}$$

   $$\forall x.\, \mathrm{walk}\, x \;\rightsquigarrow\; (x : \mathrm{entity}) \rightarrow \mathrm{walk}(x)$$

2. propositions-as-types

   $$\begin{bmatrix} x : \mathrm{entity} \\ \mathrm{walk}(x) \end{bmatrix} : \mathrm{type}$$

   $$(j, p) : \begin{bmatrix} x : \mathrm{entity} \\ \mathrm{walk}(x) \end{bmatrix}$$

   (j, p) is a proof-term of the proposition someone walks

Dependent Types

In FOL

(1) *Someone* entered. *He* smiled.

$\exists x\,(\mathrm{enter}\,x) \wedge \mathrm{smile}\,x$

In Dependent Type Theory

(2) *Someone* entered. *He* smiled.

$$\begin{bmatrix} u : \begin{bmatrix} x : \mathrm{entity} \\ \mathrm{enter}(x) \end{bmatrix} \\ \mathrm{smile}(\pi_1 u) \end{bmatrix}$$

Underspecified terms

What is the semantic representation (SR) for (3)?

(3) The elevator is clean.

(4) There is an elevator and it is clean.

Presupposition Assertion

▷ Anaphoric dependency across the two-dimensions!

SR for (3) in DTS:

$$(5)\quad \mathbf{clean}\left(\pi_1\left(@_i \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}\right)\right)$$

- The annotated type A in $@_i A$ represents the presupposition

Underspecified terms

(5) $\textbf{clean}\left(\pi_1\left(@_i\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}\right)\right)$

- For (5) to be well-formed, one needs to construct a term for the type $\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}$
- Take the first projection π_1 of a pair (x, p) where we have x : **entity** and p : **elevator**(x).

Nested presupposition

John's sister is happy.

$$\textbf{happy}\left(\pi_1\left(@_i \begin{bmatrix} x : \textbf{entity} \\ \textbf{sister}(x, \textbf{john}) \end{bmatrix}\right)\right)$$

John's sister's husband is happy.

$$\textbf{happy}\left(\pi_1\left(@_1 \begin{bmatrix} x : \textbf{entity} \\ \textbf{husband}\left(x, \pi_1\left(@_2 \begin{bmatrix} y : \textbf{entity} \\ \textbf{sister}(y, \textbf{john} \end{bmatrix})\right)\right) \end{bmatrix}\right)\right)$$

Pipeline

Text

$\downarrow$

CCG parsing + semantic composition

$\downarrow$

Underspecified SR

$\downarrow$

Type checking + proof construction

$\downarrow$

@-elimination / accommodation

$\downarrow$

Fully specified SR

$\downarrow$

Σ-type elimination

$\downarrow$

Theorem proving

System demonstration

Examples:

(6) a. A farmer who owns a donkey beats it.
b. The farmer owns the donkey.
c. Someone met his daughter yesterday.

Example

CCG Parsing and semantic composition:

The elevator is clean

$$\rightsquigarrow \ \textbf{clean}\left(\pi_1\left(@_1\left[\begin{array}{l} x : \textbf{entity} \\ \textbf{elevator}(x) \end{array}\right]\right)\right)$$

Type Checking

$$\frac{}{\textbf{clean}\left(\pi_1\left(@_1\boxed{\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}}\right)\right) : \text{type}}\ (\Pi E)$$

Type Checking

$$\dfrac{\dfrac{}{\textbf{clean} : \textbf{entity} \to \text{type}}}{\textbf{clean}\left(\pi_1\left(@_1 \boxed{\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}}\right)\right) : \text{type}} \; (\Pi E)$$

Type Checking

$$\cfrac{\cfrac{}{\textbf{clean} : \textbf{entity} \to \text{type}} \qquad \cfrac{}{\pi_1\left(@_1 \boxed{\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}}\right) : \textbf{entity}}\,(\Sigma E)}{\textbf{clean}\left(\pi_1\left(@_1 \boxed{\begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}}\right)\right) : \text{type}}\,(\Pi E)$$

Type Checking

$$
\dfrac{
  \dfrac{}{\mathbf{clean} : \mathbf{entity} \to \mathrm{type}}
  \qquad
  \dfrac{
    \dfrac{}{@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}}\,(@)
  }{
    \pi_1 \left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} \right) : \mathbf{entity}
  }\,(\Sigma E)
}{
  \mathbf{clean} \left(\pi_1 \left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} \right) \right) : \mathrm{type}
}\,(\Pi E)
$$

Type Checking

$$
\dfrac{\dfrac{}{\mathbf{clean} : \mathbf{entity} \to \mathrm{type}} \qquad \dfrac{\dfrac{\begin{matrix}\vdots\\ \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} : \mathrm{type}\end{matrix}}{@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}}\,(@)}{\pi_1\left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}\right) : \mathbf{entity}}\,(\Sigma E)}{\mathbf{clean}\left(\pi_1\left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}\right)\right) : \mathrm{type}}\,(\Pi E)
$$

Type Checking

$$
\dfrac{
  \textbf{clean} : \textbf{entity} \rightarrow \text{type}
  \qquad
  \dfrac{
    \dfrac{
      \begin{array}{c}\vdots\\ \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix} : \text{type}\end{array}
      \qquad
      \begin{array}{c}\vdots\\ ? : \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}\end{array}
    }{
      @_1 \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix} : \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix}
    }\,(@)
  }{
    \pi_1 \left(@_1 \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix} \right) : \textbf{entity}
  }\,(\Sigma E)
}{
  \textbf{clean} \left(\pi_1 \left(@_1 \begin{bmatrix} x : \textbf{entity} \\ \textbf{elevator}(x) \end{bmatrix} \right) \right) : \text{type}
}\,(\Pi E)
$$

Type Checking

$$
\dfrac{
\dfrac{}{\mathbf{clean} : \mathbf{entity} \to \mathrm{type}}
\qquad
\dfrac{
\dfrac{
\begin{matrix} \vdots \\ \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} : \mathrm{type} \end{matrix}
\qquad
\begin{matrix} \vdots \\ ? : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} \end{matrix}
}{
@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix}
}\,(@)
}{
\pi_1 \left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} \right) : \mathbf{entity}
}\,(\Sigma E)
}{
\mathbf{clean} \left(\pi_1 \left(@_1 \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{elevator}(x) \end{bmatrix} \right) \right) : \mathrm{type}
}\,(\Pi E)
$$

Option 1. Proof construction + @-elimination

Suppose we have

$$\mathcal{K} \equiv t : \textbf{entity},\ u : \textbf{elevator}(t).$$

Then we can prove:

$$\mathcal{K} \vdash (t, u) : \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{elevator}(x) \end{array}\right]$$

Replace the @-term with a constructed proof term:

$$\begin{aligned} \textbf{clean}\left(\pi_1\left(@_1 \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{elevator}(x) \end{array}\right]\right)\right) &\rightsquigarrow \textbf{clean}\,(\pi_1(t, u)) \\ &\rightarrow_\beta \textbf{clean}\,(t) \end{aligned}$$

Option 2. Accommodation

- Σ-accommodation:
  John's daughter is happy.
  ⇝ John has a daughter and she is happy.

$$\textbf{happy}\left(\pi_1\left(@_1\begin{bmatrix} x : \textbf{entity} \\ \textbf{daughter}(x, j)\end{bmatrix}\right)\right) \rightsquigarrow \begin{bmatrix} u : \begin{bmatrix} x : \textbf{entity} \\ \textbf{daughter}(x, j)\end{bmatrix} \\ \textbf{happy}(\pi_1 u)\end{bmatrix}$$

- Π-accommodation:
  John's daughter is happy.
  ⇝ If John has a daughter, she is happy.

$$\textbf{happy}\left(\pi_1\left(@_1\begin{bmatrix} x : \textbf{entity} \\ \textbf{daughter}(x, j)\end{bmatrix}\right)\right)$$
$$\rightsquigarrow \left(u : \begin{bmatrix} x : \textbf{entity} \\ \textbf{daughter}(x, j)\end{bmatrix}\right) \to \textbf{happy}(\pi_1 u)$$

Σ-type elimination

Obtain the Σ-free form using the equivalence:

$$\left[\begin{array}{l} t : \left[\begin{array}{l} u : A \\ B \end{array}\right] \\ C \end{array}\right] \equiv \left[\begin{array}{l} u : A \\ \left[\begin{array}{l} v : B \\ C\,(\pi_1 t \mapsto u, \pi_2 t \mapsto v) \end{array}\right] \end{array}\right]$$

$$\left(t : \left[\begin{array}{l} u : A \\ B \end{array}\right]\right) \to C \equiv (u : A) \to (v : B) \to C\,(\pi_1 t \mapsto u, \pi_2 t \mapsto v)$$

Example

$$\left[\begin{array}{l} t : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{entity} \\ \text{boy}(x) \end{array}\right] \\ \text{enter}(\pi_1 u) \end{array}\right] \\ \text{whistle}(\pi_1 \pi_1 t) \end{array}\right] \equiv \left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{entity} \\ \text{boy}(x) \end{array}\right] \\ \left[\begin{array}{l} \text{enter}(\pi_1 u) \\ \text{whistle}(\pi_1 u) \end{array}\right] \end{array}\right]$$

$$\equiv \left[\begin{array}{l} x : \text{entity} \\ \left[\begin{array}{l} \text{boy}(x) \\ \left[\begin{array}{l} \text{enter}(x) \\ \text{whistle}(x) \end{array}\right] \end{array}\right] \end{array}\right]$$

System demonstration

Example:

P1.	Every farmer who owns a donkey beats it.
P2.	John owns a donkey.
P3.	John is a farmer.
C	John beats a donkey.

Reference I

Abzianidze, L. (2015) “A Tableau Prover for Natural Logic and Language”, In the Proceedings of *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp.2492–2502.

Bekki, D. and K. Mineshima. (2016) “Context-Passing and Underspecification in Dependent Type Semantics”, In: *Modern Perspectives in Type Theoretical Semantics*, Studies of Linguistics and Philosophy. Springer.

Bos, J., S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier. (2004) “Wide-coverage semantic representations from a CCG parser”, In the Proceedings of *Proceedings of the 20th international conference on Computational Linguistics*. pp.1240–1246.

Butler, A. and K. Yoshimoto. (2012) “Banking meaning representations from treebanks”, *Linguistic Issues in Language Technology* **7**(1).

Kawazoe, A., R. Tanaka, K. Mineshima, and D. Bekki. (2015) “An Inference Problem Set for Evaluating Semantic Theories and Semantic Processing Systems for Japanese”, In the Proceedings of *Proceedings of LENLS12*. pp.67–73.

Reference II

Mineshima, K., P. Martínez-Gómez, Y. Miyao, and D. Bekki. (2015) "Higher-order logical inference with compositional semantics", In the Proceedings of *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp.2055–2061.

Moot, R. (2010) "Wide-coverage French syntax and semantics using Grail", In the Proceedings of *TALN 2010*.

Noji, H. and Y. Miyao. (2016) "Jigg: a framework for an easy natural language processing pipeline", In the Proceedings of *Proceedings of ACL 2016 System Demonstrations*. pp.103–108.

Uematsu, S., T. Matsuzaki, H. Hanaoka, Y. Miyao, and H. Mima. (2015) "Integrating multiple dependency corpora for inducing wide-coverage Japanese CCG resources", *ACM Transactions on Asian and Low-Resource Language Information Processing* **14**(1), pp.1–24.