Linguistic expressions:
- "Bob sings"

Logical translations:
- \( \text{sings(bob)} \)
- Could be \( p_{1218}(e_{397}) \)

Denotation:
- \( [[\text{bob}]] = \) some specific person (in some context)
- \( [[\text{sings(bob)}]] = ??? \)

Types on translations:
- \( \text{bob} : e \) (for entity)
- \( \text{sings(bob)} : t \) (for truth-value)
Compositional Semantics

- So now we have meanings for the words
- How do we know how to combine words?
- Associate a combination rule with each grammar rule:
  \[ S : \beta(\alpha) \rightarrow NP : \alpha \rightarrow VP : \beta \] (function application)
  \[ VP : \lambda x. \alpha(x) \land \beta(x) \rightarrow VP : \alpha \land \emptyset \rightarrow VP : \beta \] (intersection)
- Example:

Other Cases

- Transitive verbs:
  - likes : \lambda x. \lambda y. \text{likes}(y,x)
  - Two-place predicates of type e \rightarrow \{e \rightarrow t\}.
  - likes Amy : \lambda y. \text{likes}(y,Amy) is just like a one-place predicate.
- Quantifiers:
  - What does "Everyone" mean here?
  - Everyone : \lambda f. \forall x. f(x)
  - Mostly works, but some problems
    - Have to change our NP/VP rule.
    - Won’t work for "Amy likes everyone."
  - "Everyone likes someone."
  - This gets tricky quickly!

Denotation

- What do we do with logical translations?
  - Translation language (logical form) has fewer ambiguities
  - Can check truth value against a database
    - Denotation ("evaluation") calculated using the database
  - More usefully: assert truth and modify a database
  - Questions: check whether a statement in a corpus entails the (question, answer) pair:
    - "Bob sings and dances" \rightarrow "Who sings?" + "Bob"
    - Chain together facts and use them for comprehension

Indefinites

- First try
  - "Bob ate a waffle" : \text{ate}(bob,waffle)
  - "Amy ate a waffle" : \text{ate}(amy,waffle)

- Can't be right!
  - \exists x : \text{waffle}(x) \land \text{ate}(bob,x)
  - What does the translation of "a" have to be?
  - What about "the"?
  - What about "every"?
Grounding

- **Grounding**
  - So why does the translation \( \text{likes} : \exists x \forall y \cdot \text{likes}(y,x) \) have anything to do with actual liking?
  - It doesn’t (unless the denotation model says so)
  - Sometimes that’s enough: wire up **bought** to the appropriate entry in a database

- **Meaning postulates**
  - Insist, e.g. \( \forall x, y \cdot \text{likes}(x,y) \rightarrow \text{knows}(x,y) \)
  - This gets into lexical semantics issues

- **Statistical version?**

Tense and Events

- **In general, you don’t get far with verbs as predicates**
- **Better to have event variables** \( e \)
  - “Alice danced”: \( \text{danced}(\text{alice}) \)
  - \( \exists e : \text{dance}(e) \land \text{agent}(e, \text{alice}) \land (\text{time}(e) < \text{now}) \)
- **Event variables let you talk about non-trivial tense / aspect structures**
  - “Alice had been dancing when Bob sneezed”
  - \( \exists e, e' : \text{dance}(e) \land \text{agent}(e, \text{alice}) \land \text{sneeze}(e') \land \text{agent}(e', \text{bob}) \land (\text{start}(e) < \text{start}(e') \land \text{end}(e) = \text{end}(e')) \land (\text{time}(e') < \text{now}) \)

Adverbs

- **What about adverbs?**
  - “Bob sings terribly”
  - \( \text{terribly(sings)(bob)} \)?
  - \( (\text{terribly(sings)})(\text{bob}) \)?
  - \( \exists e \cdot \text{present}(e) \land \text{type}(e, \text{singing}) \land \text{agent}(e, \text{bob}) \land \text{manner}(e, \text{terrible}) \)?
  - It’s really not this simple...

Propositional Attitudes

- “Bob thinks that I am a gummi bear”
  - \( \text{thinks}(\text{bob}, \text{gummi(\text{me})}) \) ?
  - \( \text{thinks}(\text{bob}, \text{“I am a gummi bear”}) \) ?
  - \( \text{thinks}(\text{bob}, ^X \text{gummi(\text{me})}) \) ?

- **Usual solution involves intensions \((^X)\) which are, roughly, the set of possible worlds (or conditions) in which \( X \) is true**

- **Hard to deal with computationally**
  - Modeling other agents models, etc
  - Can come up in simple dialog scenarios, e.g., if you want to talk about what your bill claims you bought vs. what you actually bought
Trickier Stuff

- Non-Intersective Adjectives
  - green ball: \( \lambda x.\left[\text{green}(x) \land \text{ball}(x)\right] \)
  - fake diamond: \( \lambda x.\left[\text{fake}(x) \land \text{diamond}(x)\right] \)

- Generalized Quantifiers
  - the: \( \lambda f.\left[\text{unique-member}(f)\right] \)
  - all: \( \lambda f. \lambda g \left[\forall x.f(x) \Rightarrow g(x)\right] \)
  - most?
  - Could do with more general second order predicates, too (why worse?)
    - the\((\text{cat}, \text{meows})\), all\((\text{cat}, \text{meows})\)

- Generics
  - "Cats like naps"
  - "The players scored a goal"

- Pronouns (and bound anaphora)
  - "If you have a dime, put \( \) in the meter."

- ... the list goes on and on!

Multiple Quantifiers

- Quantifier scope
  - Groucho Marx celebrates quantifier order ambiguity:
    "In this country a woman gives birth every 15 min. Our job is to find that woman and stop her."

- Deciding between readings
  - "Bob bought a pumpkin every Halloween"
  - "Bob uses a phone as an alarm each morning"
  - Multiple ways to work this out
    - Make it syntactic (movement)
    - Make it lexical (type-shifting)

Modeling Uncertainty

- Big difference between statistical disambiguation and statistical reasoning.
  - The scout saw the enemy soldiers with night goggles.
    - With probabilistic parsers, can say things like "72% belief that the PP attaches to the NP."
    - That means that probably the enemy has night vision goggles.
    - However, you can't throw a logical assertion into a theorem prover with 72% confidence.
    - Use this to decide the expected utility of calling reinforcements?

- In short, we need probabilistic reasoning, not just probabilistic disambiguation followed by symbolic reasoning

Logical Form Translation
CCG Parsing

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus
  - Can have spurious ambiguities (why?)

John \vdash NP : john'
shares \vdash NP : shares'
boys \vdash (S\NP)/NP : \lambda x.\lambda y.boys'xy
sleeps \vdash S\NP : \lambda x.sleeps'x
well \vdash (S\NP)(S\NP) : \lambda f.\lambda x.well'(fx)

John (S\NP)/NP NP
boys NP
shares

Mapping to LF: Zettlemoyer & Collins 05/07

The task:
Input: List one way flights to Prague.
Output: \lambda x.\text{flight}(x) \wedge \text{one}_\text{way}(x) \wedge \text{to}(x,\text{PRG})

Challenging learning problem:
- Derivations (or parses) are not annotated
- Approach: [Zettlemoyer & Collins 2005]
- Learn a lexicon and parameters for a weighted Combinatory Categorial Grammar (CCG)

[Slides from Luke Zettlemoyer]

Background

- Combinatory Categorial Grammar (CCG)
- Weighted CCGs
- Learning lexical entries: GENLEX

CCG Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights</td>
<td>N : \lambda x.\text{flight}(x)</td>
</tr>
<tr>
<td>to</td>
<td>(N\NP)/NP : \lambda x.\lambda y.\text{f}(x) \wedge \text{to}(y,x)</td>
</tr>
<tr>
<td>Prague</td>
<td>NP : PRG</td>
</tr>
<tr>
<td>New York city</td>
<td>NP : NYC</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Parsing Rules (Combinators)

**Application**
- \( X/Y : f \quad Y : a \Rightarrow X : f(a) \)
- \( Y : a \quad X \Rightarrow Y : f \Rightarrow X : f(a) \)

**Composition**
- \( X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x)) \)
- \( Y/Z : f \quad X \Rightarrow Y/Z : \lambda x.f(g(x)) \)

**Additional rules:**
- Type Raising
- Crossed Composition

### CCG Parsing

#### Show me flights to Prague

<table>
<thead>
<tr>
<th>S/N</th>
<th>N</th>
<th>(N\N) / NP</th>
<th>NP</th>
<th>PRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Af.f</td>
<td>( \lambda x.\text{flight}(x) )</td>
<td>( \lambda y.\lambda f.\lambda x.\text{flight}(y) \land \text{to}(x,y) )</td>
<td>N\N</td>
<td>( \lambda x.\text{flight}(x) \land \text{to}(x,\text{PRG}) )</td>
</tr>
<tr>
<td>( \lambda x.\text{flight}(x) \land \text{to}(x,\text{PRG}) )</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Weighted CCG

Given a log-linear model with a CCG lexicon \( \Lambda \), a feature vector \( f \), and weights \( w \).

- The best parse is:
  \[
  y^* = \arg \max_y w \cdot f(x,y)
  \]

Where we consider all possible parses \( y \) for the sentence \( x \) given the lexicon \( \Lambda \).

### Lexical Generation

#### Input Training Example

- **Sentence:** Show me flights to Prague.
- **Logic Form:** \( \lambda x.\text{flight}(x) \land \text{to}(x,\text{PRG}) \)

#### Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me</td>
<td>S/N : Af.f</td>
</tr>
<tr>
<td>flights</td>
<td>N : ( \lambda x.\text{flight}(x) )</td>
</tr>
<tr>
<td>to</td>
<td>(N\N) / NP : ( \lambda x.\lambda f.\lambda y.\text{flight}(y) \land \text{to}(y,x) )</td>
</tr>
<tr>
<td>Prague</td>
<td>NP : PRG</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GENLEX: Substrings X Categories

**Input Training Example**

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Logic Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me flights to Prague.</td>
<td>$\lambda x.\text{flight}(x) \land \text{to}(x, \text{PRG})$</td>
</tr>
</tbody>
</table>

**All possible substrings:**

- Show
- me
- flights
- ...
- Show me
- Show me flights
- Show me flights to
- ...

**Categories created by rules that trigger on the logical form:**

- NP : $\text{PRG}$
- N : $\lambda x.\text{flight}(x)$
- (S\NP)/NP : $\lambda x,\lambda y.\text{to}(y, x)$
- (N\N)/NP : $\lambda y, \lambda f, \lambda x. \ldots$

**Output Lexicon**

[Zettlemoyer & Collins 2005]

---

**Robustness**

**The lexical entries that work for:**

Show me the latest flight from Boston to Prague on Friday

**Will not parse:**

Boston to Prague the latest on Friday

---

**Relaxed Parsing Rules**

- **Two changes**
  - Add application and composition rules that relax word order
  - Add type shifting rules to recover missing words

**These rules significantly relax the grammar**

- Introduce features to count the number of times each new rule is used in a parse

---

**Review: Application**

$$X/Y : f \quad Y : a \quad \Rightarrow \quad X : f(a)$$

$$Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : f(a)$$
Disharmonic Application

- Reverse the direction of the principal category:
  \[
  \begin{align*}
  X \setminus Y : f & \quad Y : a \quad \Rightarrow \quad X : f(a) \\
  Y : a & \quad X \setminus Y : f \quad \Rightarrow \quad X : f(a)
  \end{align*}
  \]

$$\text{flights} \quad \text{one way}$$

$$\begin{array}{c}
\text{flights} \\
\text{from} \quad \text{to}
\end{array}$$

$$\text{flights} \quad \text{one way} \quad \text{to Prague}$$

**Insert missing semantic content**

- NP : c \quad \Rightarrow \quad N \setminus N : f. \quad \lambda x. f(x) \land p(x, c)

**Inputs:** Training set \(\{(x, y)\} \mid i = 1 \ldots n\) of sentences and logical forms. Initial lexicon \(\Lambda\). Initial parameters \(w\). Number of iterations \(T\).

**Training:** For \(i = 1 \ldots T, i = 1 \ldots n\):

1. **Step 1: Check Correctness**
   - Let \(y^* = \text{argmax}_y w \cdot f(x_i, y)\)
   - If \(L(y^*) \neq z_i\), go to the next example

2. **Step 2: Lexical Generation**
   - Set \(\lambda = \Lambda \cup \text{GENLEX}(x_i, z_i)\)
   - Let \(\lambda = \text{argmax}_\lambda \max_{w} w \cdot f(x_i, y)\)
   - Define \(\lambda_i\) to be the lexical entries in \(y^*\)
   - Set lexicon to \(\Lambda = \Lambda \cup \lambda_i\)

3. **Step 3: Update Parameters**
   - Let \(y^* = \text{argmax}_y w \cdot f(x_i, y)\)
   - If \(L(y^*) \neq z_i\),
     - Set \(w = w + f(x_i, y^*) - f(x_i, y^*)\)

**Output:** Lexicon \(\Lambda\) and parameters \(w\). 

**Missing content words**

**Insert missing semantic content**

- NP : c \quad \Rightarrow \quad N \setminus N : f. \quad \lambda x. f(x) \land p(x, c)
Related Work for Evaluation

Hidden Vector State Model: He and Young 2006
- Learns a probabilistic push-down automaton with EM
- Is integrated with speech recognition

λ-WASP: Wong & Mooney 2007
- Builds a synchronous CFG with statistical machine translation techniques
- Easily applied to different languages

Zettlemoyer and Collins 2005
- Uses GENLEX with maximum likelihood batch training and stricter grammar

Two Natural Language Interfaces

ATIS (travel planning)
- Manually-transcribed speech queries
- 4500 training examples
- 500 example development set
- 500 test examples

Geo880 (geography)
- Edited sentences
- 600 training examples
- 280 test examples

Evaluation Metrics

Precision, Recall, and F-measure for:
- Completely correct logical forms
- Attribute / value partial credit

\[ x.\text{flight}(x) \land \text{from}(x, \text{BOS}) \land \text{to}(x, \text{PRG}) \]

is represented as:

\[ \{ \text{from} = \text{BOS}, \text{to} = \text{PRG} \} \]

Two-Pass Parsing

Simple method to improve recall:
- For each test sentence that can not be parsed:
  - Reparse with word skipping
  - Every skipped word adds a constant penalty
  - Output the highest scoring new parse
### ATIS Test Set [Z+C 2007]

Exact Match Accuracy:

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>90.61</td>
<td>81.92</td>
<td><strong>86.05</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>85.75</td>
<td>84.60</td>
<td>85.16</td>
</tr>
</tbody>
</table>

### Geo880 Test Set

Exact Match Accuracy:

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>95.49</td>
<td>83.20</td>
<td><strong>88.93</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>91.63</td>
<td>86.07</td>
<td>88.76</td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2005</td>
<td>96.25</td>
<td>79.29</td>
<td>86.95</td>
</tr>
<tr>
<td>Wong &amp; Mooney 2007</td>
<td>93.72</td>
<td>80.00</td>
<td>86.31</td>
</tr>
</tbody>
</table>