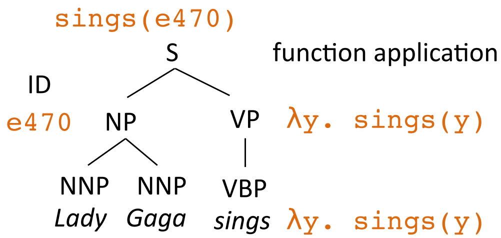
Montague Semantics



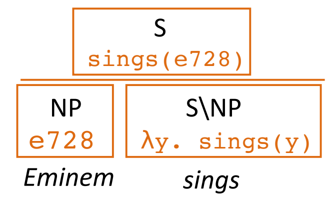
function application: apply this to e470

takes one argument (y, the entity) and returns a logical form `sings(y)`

- We can use the syntactic parse as a bridge to the lambda-calculus representation, build up a logical form (our model) *compositionally*

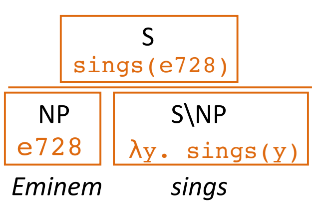
Combinatory Categorial Grammar

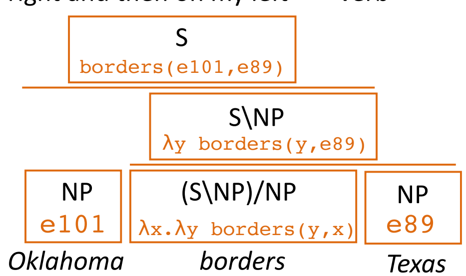
- Steedman+Szabolcsi (1980s): formalism bridging syntax and semantics
- Parallel derivations of syntactic parse and lambda calculus expression
- Syntactic categories (for this lecture): S, NP, "slash" categories
- S\NP: "if I combine with an NP on my left side, I form a sentence" — verb
- When you apply this, there has to be a parallel instance of function application on the semantics side



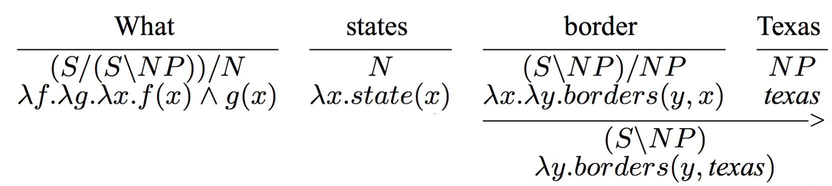
Combinatory Categorial Grammar

- Steedman+Szabolcsi (1980s): formalism bridging syntax and semantics
- Syntactic categories (for this lecture): S, NP, "slash" categories
  - S\NP: "if I combine with an NP on my left side, I form a sentence" — verb
  - (S\NP)/NP: "I need an NP on my right and then on my left" — verb with a direct object





CCG Parsing



- "What" is a ***very*** complex type: needs a noun and needs a S\NP to form a sentence. S\NP is basically a verb phrase (*border Texas*)

Zettlemoyer and Collins (2005)

CCG Parsing

$$
\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas
\end{array}
$$

$$
\xrightarrow{>} \quad S/(S\backslash NP) \quad \lambda g.\lambda x.state(x) \wedge g(x) \qquad \xrightarrow{>} \quad (S\backslash NP) \quad \lambda y.borders(y, texas)
$$

$$
\xrightarrow{>} \quad S \quad \lambda x.state(x) \wedge borders(x, texas)
$$

- "What" is a ***very*** complex type: needs a noun and needs a S\NP to form a sentence. S\NP is basically a verb phrase (*border Texas*)
- *What* in this case knows that there are two predicates (*states* and *border Texas*). This is not a general thing

Zettlemoyer and Collins (2005)

CCG Parsing

- These question are *compositional*: we can build bigger ones out of smaller pieces

What states border Texas?

What states border states bordering Texas?

What states border states bordering states bordering Texas?

CCG Parsing

- Many ways to build these parsers
- One approach: run a "supertagger" (tags the sentence with complex labels), then run the parser

$$
\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas
\end{array}
$$

  - Parsing is easy once you have the tags, so we've reduced it to a (hard) tagging problem

Zettlemoyer and Collins (2005)

Training CCG Parsers

- Training data looks like pairs of sentences and logical forms

What states border Texas	`λx. state(x) ∧ borders(x, e89)`
What borders Texas	`λx. borders(x, e89)`
…	

- Unlike PCFGs, we don't know which words yielded which fragments of CCG
- Requires an "unsupervised" approach like Model 1 for word alignment

Zettlemoyer and Collins (2005)

Seq2seq Semantic Parsing

Semantic Parsing as Translation

"what states border Texas"

↓

```
lambda x ( state ( x ) and border ( x , e89 ) ) )
```

- Write down a linearized form of the semantic parse, train seq2seq models to directly translate into this representation
- What are some benefits of this approach compared to grammar-based?
- What might be some concerns about this approach? How do we mitigate them?

Jia and Liang (2016)

Handling Invariances

"what states border Texas" *"what states border Ohio"*

- Parsing-based approaches handle these the same way
  - Possible divergences: features, different weights in the lexicon
- Can we get seq2seq semantic parsers to handle these the same way?
- Key idea: do data augmentation by synthetically creating more data from a single example

Semantic Parsing as Translation

Geo
x: *"what is the population of iowa ?"*
y:
```
_answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```

ATIS
x: *"can you list all flights from chicago to milwaukee"*
y:
```
( _lambda $0 e ( _and
  ( _flight $0 )
  ( _from $0 chicago : _ci )
  ( _to $0 milwaukee : _ci ) ) )
```

Overnight
x: *"when is the weekly standup"*
y:
```
( call listValue ( call
    getProperty meeting.weekly_standup
    ( string start_time ) ) )
```

- Prolog
- Lambda calculus
- Other DSLs

- Handle all of these with uniform machinery!

Jia and Liang (2016)

Semantic Parsing as Translation

	Geo	ATIS
Previous Work		
Zettlemoyer and Collins (2007)		**84.6**
Kwiatkowski et al. (2010)	88.9	
Liang et al. (2011)[2]	91.1	
Kwiatkowski et al. (2011)	88.6	82.8
Poon (2013)		83.5
Zhao and Huang (2015)	88.9	84.2
Our Model		
No Recombination	85.0	76.3
AbsEntities	85.4	79.9
AbsWholePhrases	87.5	
Concat-2	84.6	79.0
Concat-3		77.5
AWP + AE	88.9	
AE + C2		78.8
AWP + AE + C2	**89.3**	
AE + C3		83.3

- Three forms of data augmentation all help
- Results on these tasks are still not as strong as hand-tuned systems from 10 years ago, but the same simple model can do well at all problems

Jia and Liang (2016)

Applications

- GeoQuery (Zelle and Mooney, 1996): answering questions about states (~80% accuracy)
- Jobs: answering questions about job postings (~80% accuracy)
- ATIS: flight search
- Can do well on all of these tasks if you handcraft systems and use plenty of training data: these domains aren't that rich

Next Time

- QA from raw text: how do we answer a question about a passage?
- Neural networks for QA
- Final project discussion