Executable Semantic Parsing

ESSLLI

Class 2

Instructor: Jonathan Berant

Class 1 reminder

We learned to manually write down logical forms for various natural language constructs

- Entities
- Relations
- Superlatives
- Events
- $\cdots$

But how do we build logical forms automatically?

Outline

- **Compositionality**
  - Grammars and derivations
  - Example
  - Lexicon
  - Modeling
  - CCG
  - Exercise

Outline

- Compositionality
  - **Grammars and derivations**
  - Example
  - Lexicon
  - Modeling
  - CCG
  - Exercise

Grammars and derivations

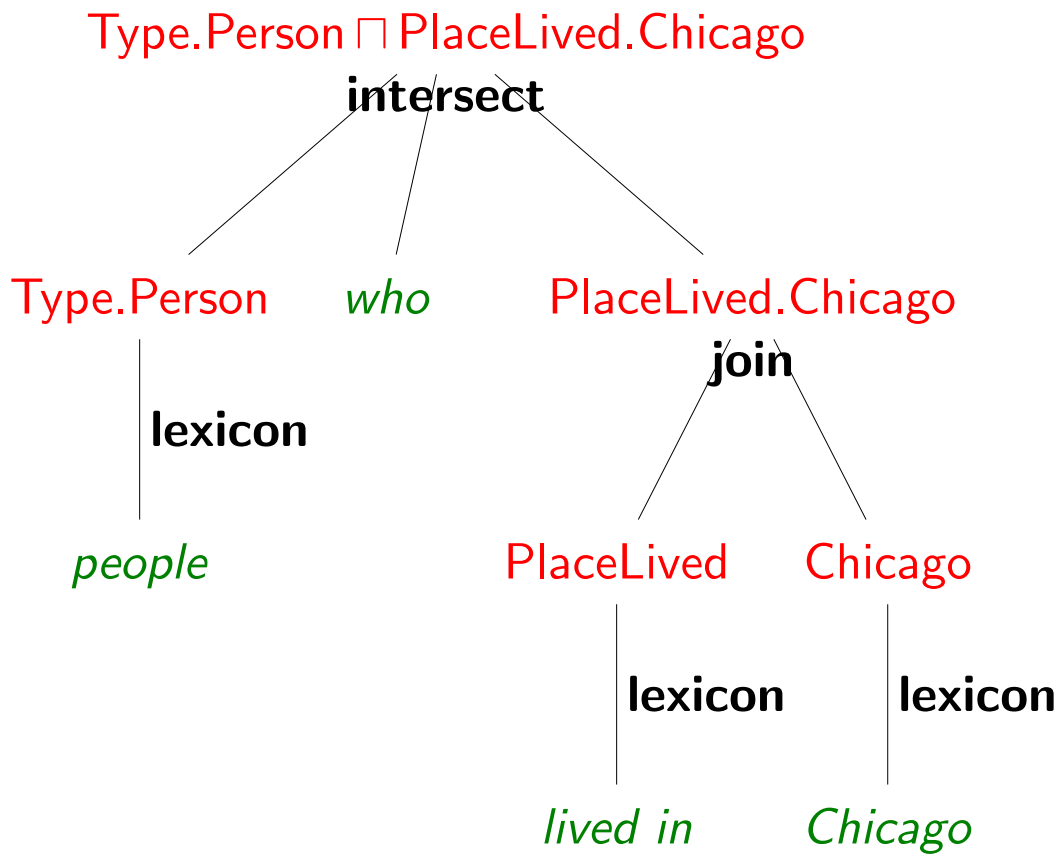
Simple grammar

(lexicon)	`Set`	$\Rightarrow$ `Phrase`	Chicago	$\Rightarrow$	*Chicago*
(lexicon)	`Binary`	$\Rightarrow$ `Phrase`	PlacesLived	$\Rightarrow$	*lived in*
(join)	`Set`	$\Rightarrow$ `Binary Set`	$b.s$	$\Rightarrow$	$b\ s$
(intersect)	`Root`	$\Rightarrow$ `Set Set`	$s_1 \sqcap s_2$	$\Rightarrow$	$s_1\ s_2$

Grammars and derivations

Simple grammar

(lexicon)	`Set`	$\Rightarrow$ `Phrase`	Chicago	$\Rightarrow$	*Chicago*
(lexicon)	`Binary`	$\Rightarrow$ `Phrase`	PlacesLived	$\Rightarrow$	*lived in*
(join)	`Set`	$\Rightarrow$ `Binary Set`	$b.s$	$\Rightarrow$	$b\ s$
(intersect)	`Root`	$\Rightarrow$ `Set Set`	$s_1 \sqcap s_2$	$\Rightarrow$	$s_1\ s_2$



Grammar

Simple grammar

(lexicon)	`Set`	$\Rightarrow$ `Phrase`	Chicago	$\Rightarrow$	*Chicago*
(lexicon)	`Binary`	$\Rightarrow$ `Phrase`	PlacesLived	$\Rightarrow$	*lived in*
(join)	`Set`	$\Rightarrow$ `Binary Set`	$b.s$	$\Rightarrow$	$b\ s$
(intersect)	`Root`	$\Rightarrow$ `Set Set`	$s_1 \sqcap s_2$	$\Rightarrow$	$s_1\ s_2$

A grammar G is a 4-tuple:

$\mathcal{V}$:	*chicago, live in*
$\mathcal{N}$:	$\textsc{Binary}, \textsc{Set}, \textsc{Root}$
$\textsc{Root} \in \mathcal{N}$:	start symbol
$\mathcal{R}$:	grammar rules

Grammar rules

Simple grammar

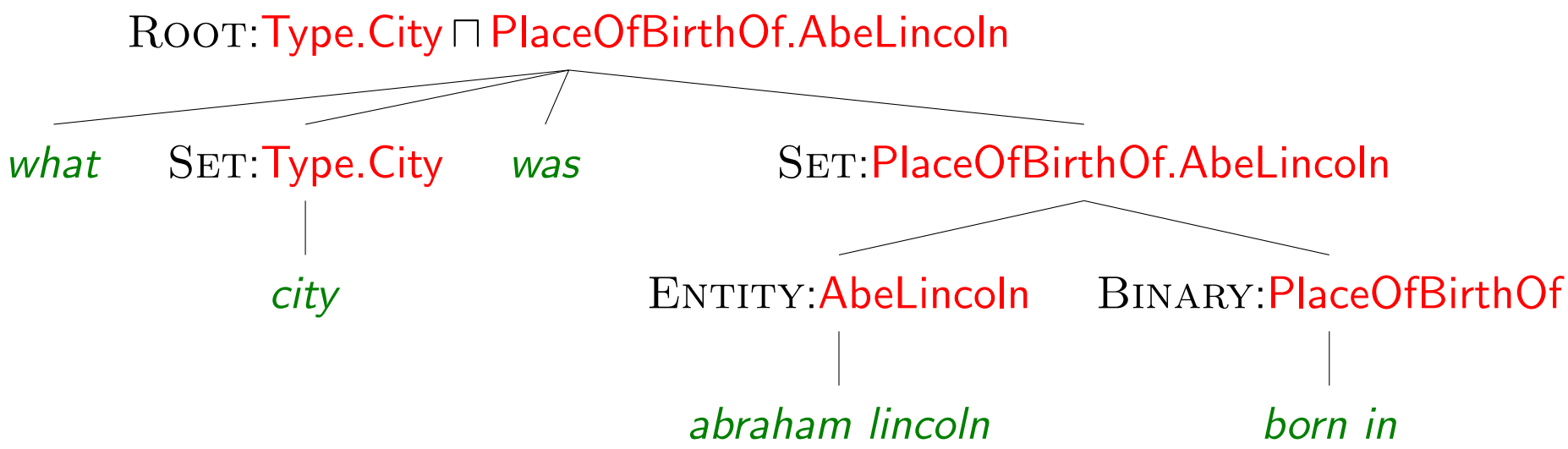
(lexicon)	`Set`	$\Rightarrow$ `Phrase`	Chicago	$\Rightarrow$	*Chicago*
(lexicon)	`Binary`	$\Rightarrow$ `Phrase`	PlacesLived	$\Rightarrow$	*lived in*
(join)	`Set`	$\Rightarrow$ `Binary Set`	$b.s$	$\Rightarrow$	$b\ s$
(intersect)	`Root`	$\Rightarrow$ `Set Set`	$s_1 \sqcap s_2$	$\Rightarrow$	$s_1\ s_2$

A rule $r \in \mathcal{R}$ has:

$A \in \mathcal{N}$:	left-hand-side (LHS) non-terminal
$\alpha \in (\mathcal{N} \cup \mathcal{V})^+$	right-hand-side
f	semantic function for building derivations

Semantic functions are the key component!

Derivation



A derivation tree $d_{i:j}^{A}$ over a span $x = (w_i, \ldots, w_{j-1})$:

Spans $(w_i, \ldots, w_{j-1})$

Has category $d.c = A \in \mathcal{N}$

Has logical form $d.z$

Has children that are derivations or terminals

Semantic functions

A function $f : \mathcal{D}^k \rightarrow 2^{\mathcal{D}}$

Example:

- $r = A \rightarrow B\ C\ [f]$
- $D^A_{i:j} = f(d^B_{i:k}, d^C_{k:j})$

Lexicon function:

$$\textsc{Lex}(\textit{lincoln}) = \left\{ \begin{array}{c} \textsc{Entity}:\text{AbeLincoln} \\ | \\ \textit{lincoln} \end{array}, \begin{array}{c} \textsc{Entity}:\text{LincolnFilm} \\ | \\ \textit{lincoln} \end{array}, \ldots \right\}$$

Semantic functions

Join function:

$$\textsc{Join}\left(\begin{array}{c}\textsc{Entity}\text{:AbeLincoln}\\ \triangle\\ \textit{abraham lincoln}\end{array},\ \begin{array}{c}\textsc{Binary}\text{:PlaceOfBirthOf}\\ \triangle\\ \textit{born}\end{array}\right) = \left\{\begin{array}{c}\textsc{Set}\text{:PlaceOfBirthOf.AbeLincoln}\\ \begin{array}{cc}\textsc{Entity}\text{:AbeLincoln} & \textsc{Binary}\text{:PlaceOfBirthOf}\\ \triangle & \triangle\\ \textit{abraham lincoln} & \textit{born}\end{array}\end{array}\right\}$$

Merge function:

$$\textsc{Intersect}\left(\begin{array}{c}\textsc{Set}\text{:Type.City}\\ \triangle\\ \textit{city}\end{array},\ \begin{array}{c}\textsc{Set}\text{:ReleaseDateOf.LincolnFilm}\\ \triangle\\ \textit{abraham lincoln born}\end{array}\right) = \{\}$$

Return sets (unlike CFGs)

Semantic function

IdentityFn	Copy logical form
SelectFn	Select one child logical form
DateFn	Convert date descriptions to date
FilterPosFn	Filter words by POS

Semantic function

IdentityFn	Copy logical form
SelectFn	Select one child logical form
DateFn	Convert date descriptions to date
FilterPosFn	Filter words by POS
ContextFn	Integrate arbitrary context

What city was abraham lincoln born?

How about his wife?

Semantic function

IdentityFn	Copy logical form
SelectFn	Select one child logical form
DateFn	Convert date descriptions to date
FilterPosFn	Filter words by POS
ContextFn	Integrate arbitrary context

What city was abraham lincoln born?

How about his wife?

Flexible! arbitrary logic can be used

$\lambda x.\textsc{Border}(x, \textsc{Italy})$		$\lambda y.\lambda x.\textsc{Border}(x, y)$	$\textsc{Italy}$
$S \backslash N$	$\Rightarrow$	$(S \backslash N)/N$	N
bordering Italy		bordering	Italy

Outline

- Compositionality
  - Grammars and derivations
  - **Example**
  - Lexicon
  - Modeling
  - CCG
  - Exercise

Example

```
./run @mode=simple-lambdadcs \

-Grammar.inPaths esslli_2016/class2_demo.grammar \

-SimpleLexicon.inPaths esslli_2016/class2_demo.lexicon
```

Example

california

the golden state

cities in the golden state

towns located in california

rivers bordering california

Example

california

the golden state

cities in the golden state

towns located in california

rivers bordering california

Is the last utterance meaningful?

How can we avoid such logical forms?

Outline

- Compositionality
  - Grammars and derivations
  - Example
  - **Lexicon**
  - Modeling
  - CCG
  - Exercise

The lexicon problem

california	California
the golden state	California
cities	Type.Citytown
towns	Type.CityTown
in	ContainedBy

How is the lexicon generated?

The lexicon problem

california	California
the golden state	California
cities	Type.Citytown
towns	Type.CityTown
in	ContainedBy

How is the lexicon generated?

- Annotation
- Exhaustive search
- String matching
- Supervised alignment
- Unsupervised alginment
- Learning

Annotation

where	Type.Location
most	$\lambda x.\lambda y.\textsc{argmax}(y, x)$
how many	$\lambda x.\textsc{count}(x)$

Useful for function words

Annotation

where	Type.Location
most	$\lambda x.\lambda y.\textsc{argmax}(y, x)$
how many	$\lambda x.\textsc{count}(x)$

Useful for function words

city	Type.CityTown
lake	Type.Lake
area	Area

Useful for small domains

Annotation

where	Type.Location
most	$\lambda x.\lambda y.\textsc{argmax}(y, x)$
how many	$\lambda x.\textsc{count}(x)$

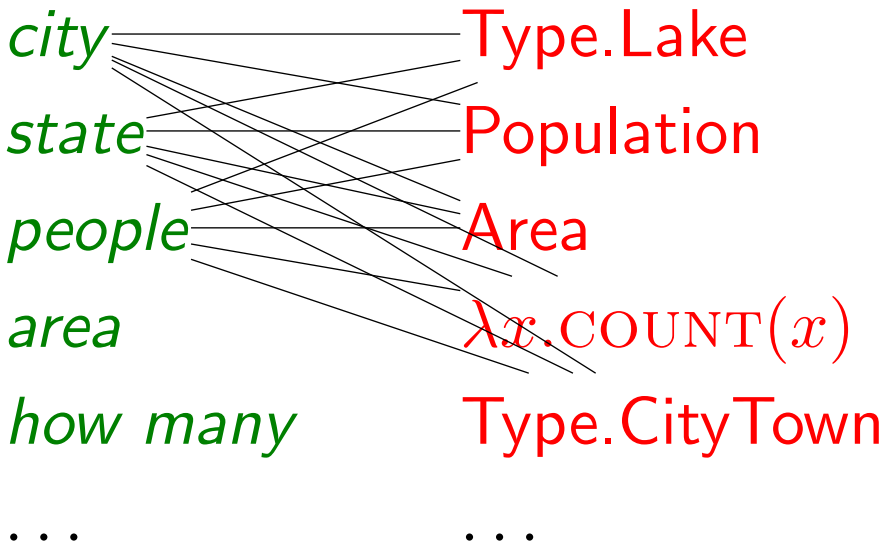
Useful for function words

city	Type.CityTown
lake	Type.Lake
area	Area

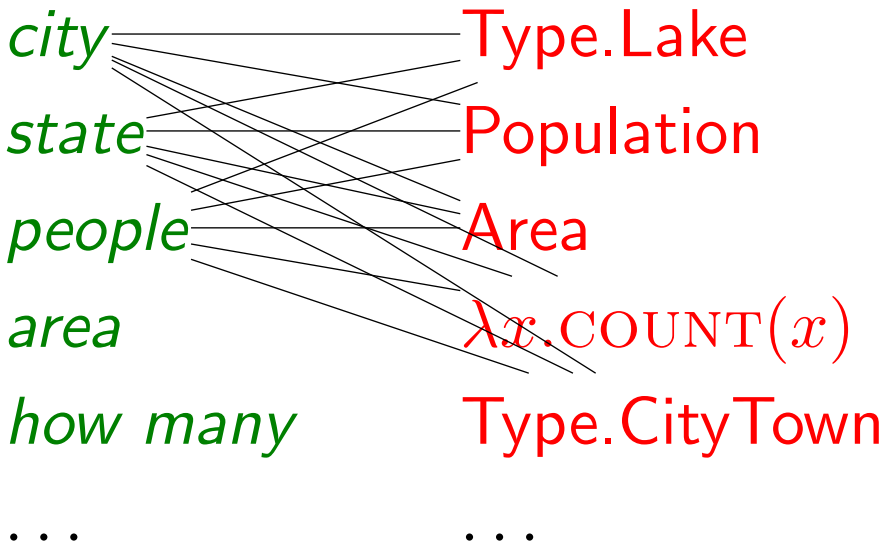
Useful for small domains

Large domains? Different languages?

Exhaustive search



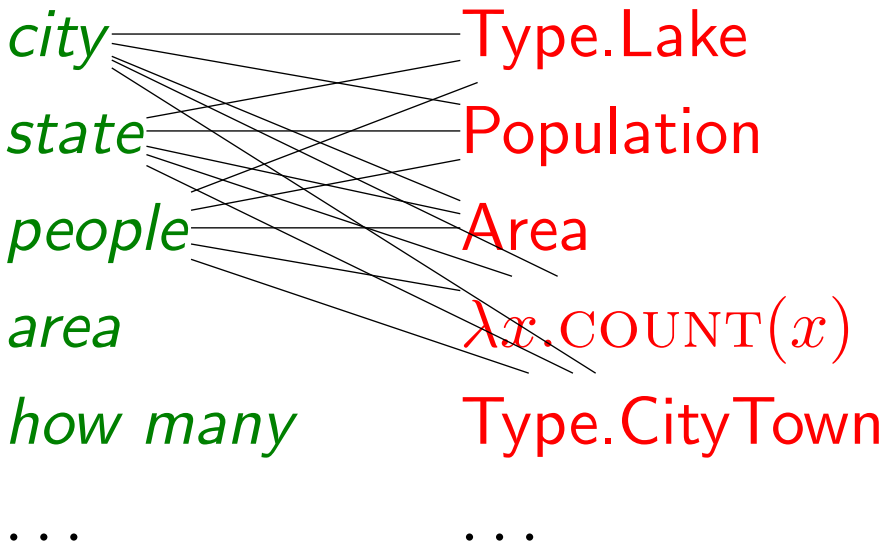
Exhaustive search



Size of lexicon: $|\mathcal{V}| \times |\mathcal{L}|$

Intractable for large domains

Exhaustive search



Size of lexicon: $|\mathcal{V}| \times |\mathcal{L}|$

Intractable for large domains

Exhaustive?

actress	$\textsc{Profession.Actor} \sqcap \textsc{Gender.Female}$
relative	$\lambda x.\textsc{Parent}.x \sqcup \textsc{Sibling}.x \sqcup \textsc{Child}.x$
expat	$\mu x.\text{BirthPlaceOf}.x \neq \textsc{argmax}(x, \text{PlacesLived})$

String match

Knowledge-bases often have language descriptions

area Area

- Language specific
- Limited coverage and noisy

String match

Knowledge-bases often have language descriptions

area Area

- Language specific
- Limited coverage and noisy
- Useful as a feature at training time

Supervised alignment

Assume access to (x, z) language-logical form pairs

State with the largest area

$\textsc{argmax}(\textsc{Type.State}, \textsc{Area})$

Supervised alignment

Assume access to (x, z) language-logical form pairs

State with the largest area

$\textsc{argmax}(\textsc{Type.State}, \textsc{Area})$

Define patterns over (x, z) to generate candidate lexicon entries

State	State
with the largest	Type.State
largest	$\lambda x.\text{Type}.x$
area	Area
	$\lambda x.\lambda y.\textsc{Argmax}(x, y)$

Supervised alignment

Assume access to (x, z) language-logical form pairs

State with the largest area

$\textsc{argmax}(\textsc{Type.State}, \textsc{Area})$

Define patterns over (x, z) to generate candidate lexicon entries

State	State
with the largest	Type.State
largest	$\lambda x.\text{Type}.x$
area	Area
	$\lambda x.\lambda y.\textsc{Argmax}(x, y)$

Keep counts for each lexical entry as a feature

Supervised alignment

Assume access to (x, z) language-logical form pairs

State with the largest area

$\textsc{argmax}(\textsc{Type.State}, \textsc{Area})$

Define patterns over (x, z) to generate candidate lexicon entries

State	State
with the largest	Type.State
largest	$\lambda x.\text{Type}.x$
area	Area
	$\lambda x.\lambda y.\textsc{Argmax}(x, y)$

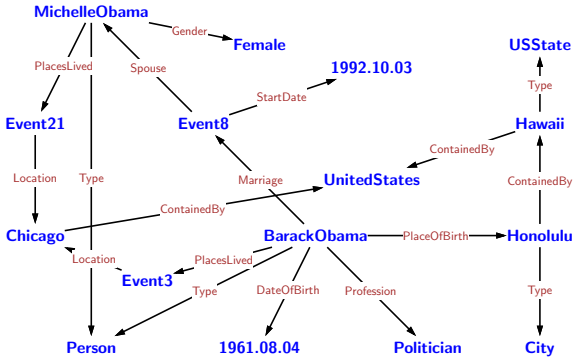
Keep counts for each lexical entry as a feature

Predefined patterns

Generalization?

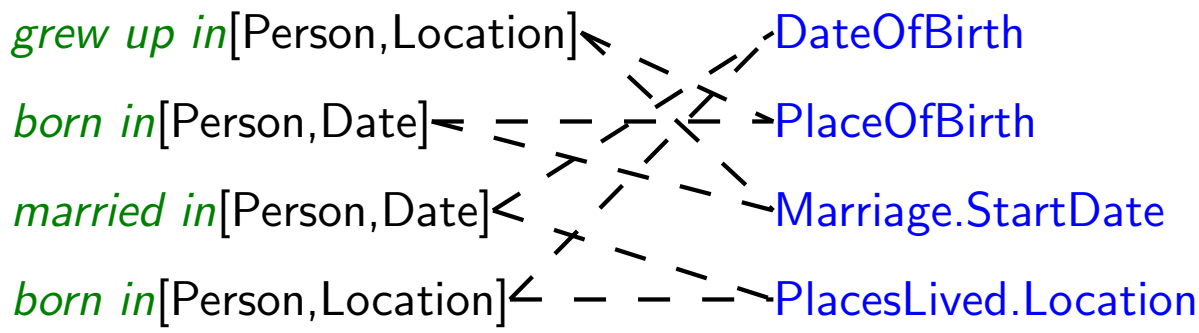
Unsupervised alignment

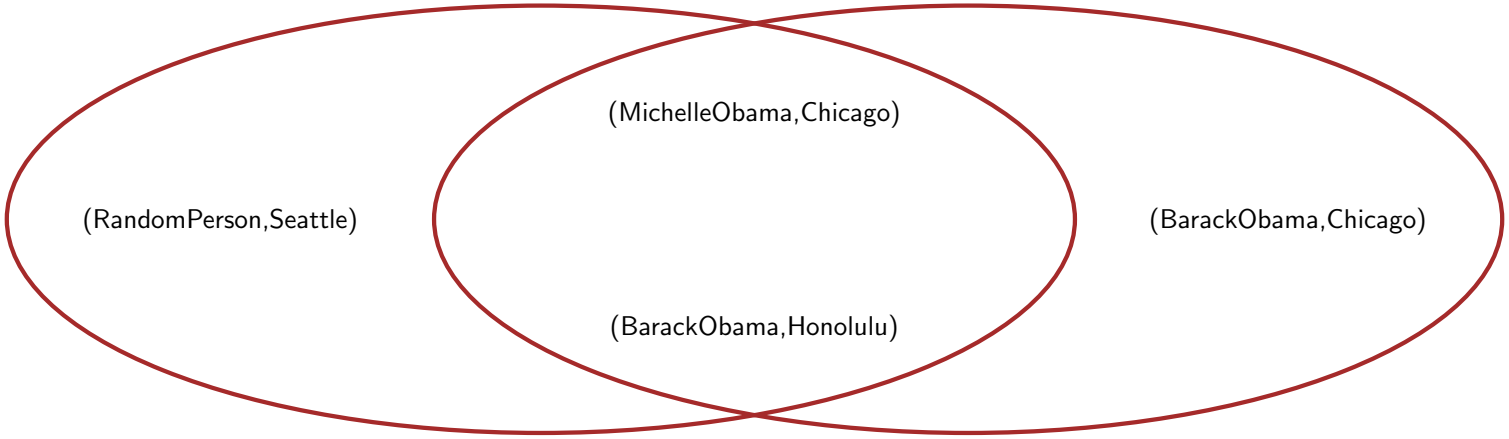
(*Barack Obama*, *was born in*, *Honolulu*)
(*Albert Einstein*, *was born in*, *Ulm*)
(*Barack Obama*, *lived in*, *Chicago*)
... 15M triples ...



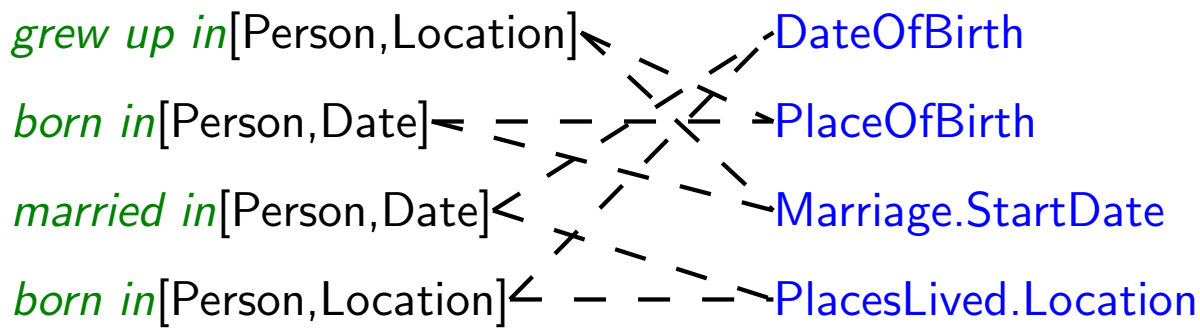
(BarackObama, HasBirthplace, Honolulu)
(Albert Einstein, HasBirthplace, Ulm)
(BarackObama, PlacesLived.Location, Chicago)
... 600M triples ...

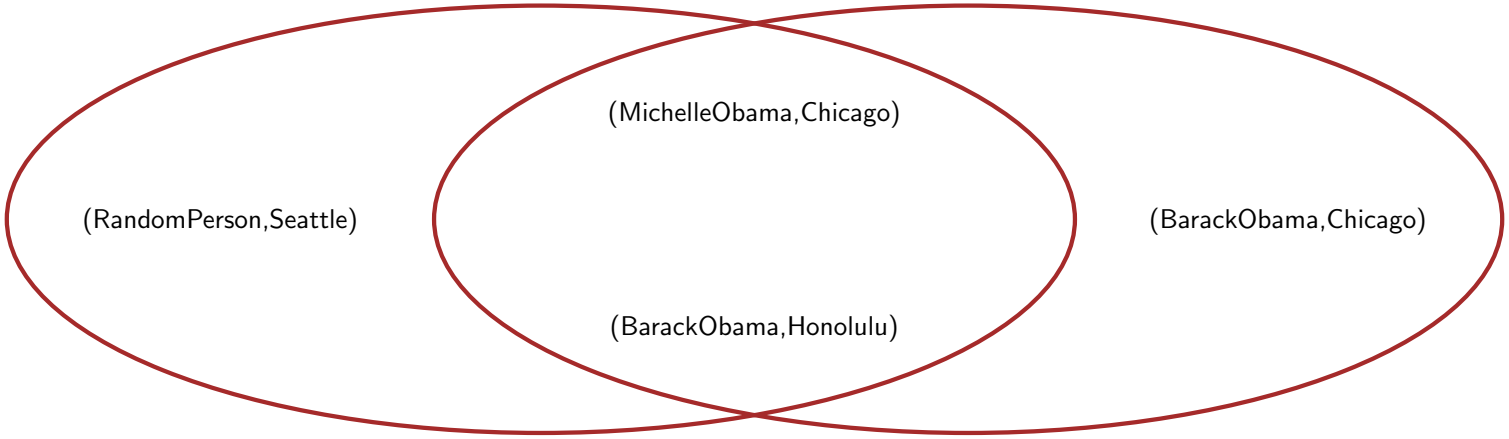
Unsupervised alignment





Unsupervised alignment

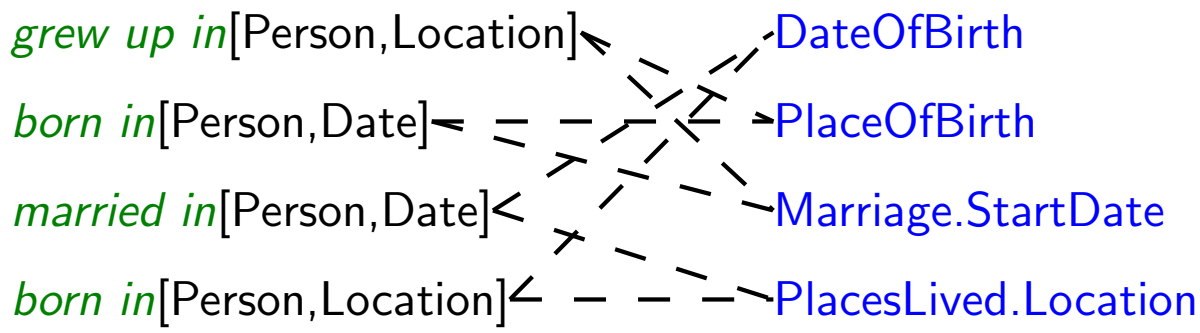


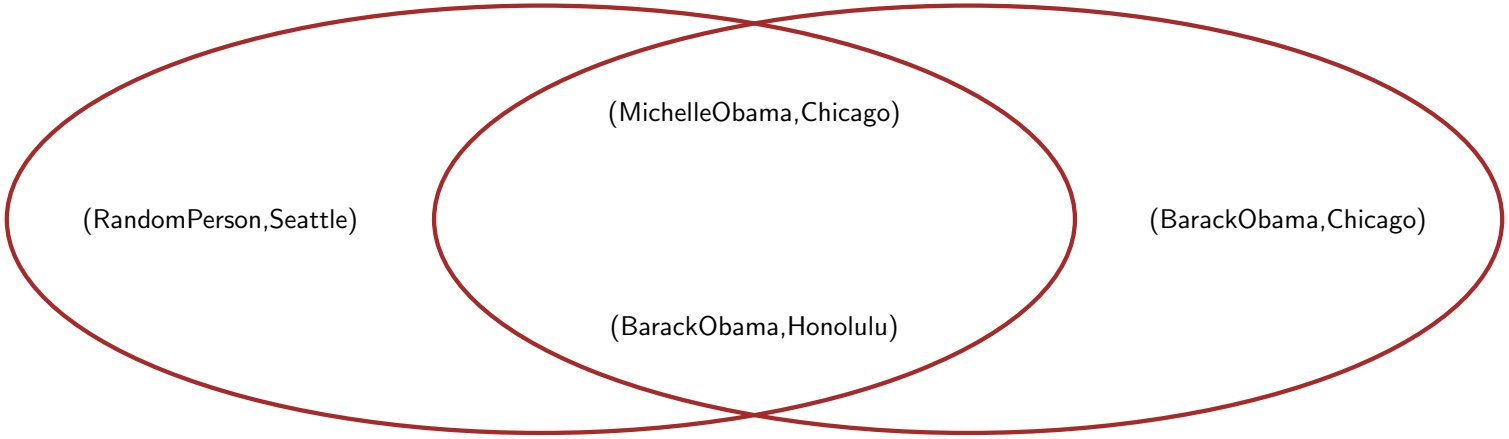


Alignment features

phrase-count:15,765
intersection-count: 6,048

Unsupervised alignment





Alignment features

phrase-count:15,765
intersection-count: 6,048

Generalizes to test set

Restricted domain

Advanced

- Change lexicon at training time
- batch voting
- Generative models

[Kwiatkowski et al., 2010; Kwiatkowski et al. 2011; Artzi et al., 2014; Krishnamurthy, 2015]

Outline

- Compositionality
  - Grammars and derivations
  - Example
  - Lexicon
  - **Modeling**
  - CCG
  - Exercise

Bridging/trace predicates

Problem: Language defines semantics implicitly

Italy's language	Type.Language $\sqcap$ LanguageOf.Italy
Where is Beijing	Type.Location $\sqcap$ Contains.Beijing
Boston flight	Type.Flight $\sqcap$ To.Boston

Bridging/trace predicates

Problem: Language defines semantics implicitly

Italy's language	Type.Language $\sqcap$ LanguageOf.Italy
Where is Beijing	Type.Location $\sqcap$ Contains.Beijing
Boston flight	Type.Flight $\sqcap$ To.Boston

Lexicon?

's	LanguageOf
is	Contains
???	To

Naive solution

language	Type.Language
language	LanguageOf
where	Type.Location
where	Contains
flight	Type.Filght
flight	To
flight	From

More pressure on lexicon learning

Naive solution

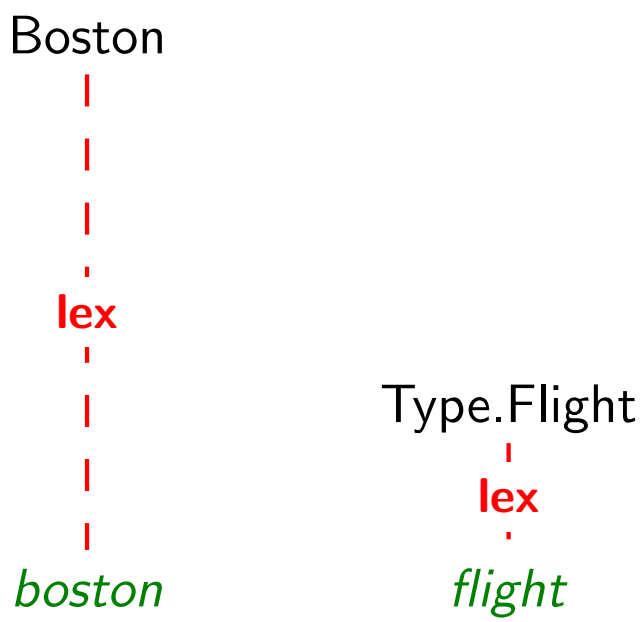
language	Type.Language
language	LanguageOf
where	Type.Location
where	Contains
flight	Type.Filght
flight	To
flight	From

More pressure on lexicon learning

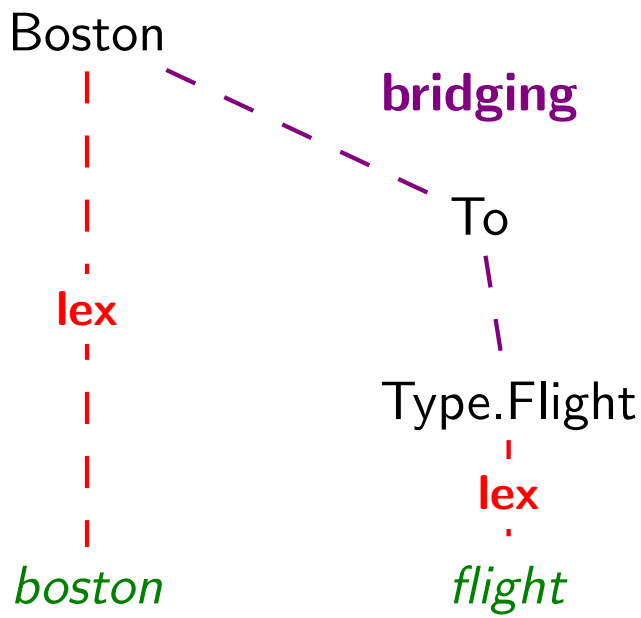
Infer missing predicates from context

Can be implemented as a semantic function! (BridgeFn)

Bridging 1: two unaries



Bridging 1: two unaries

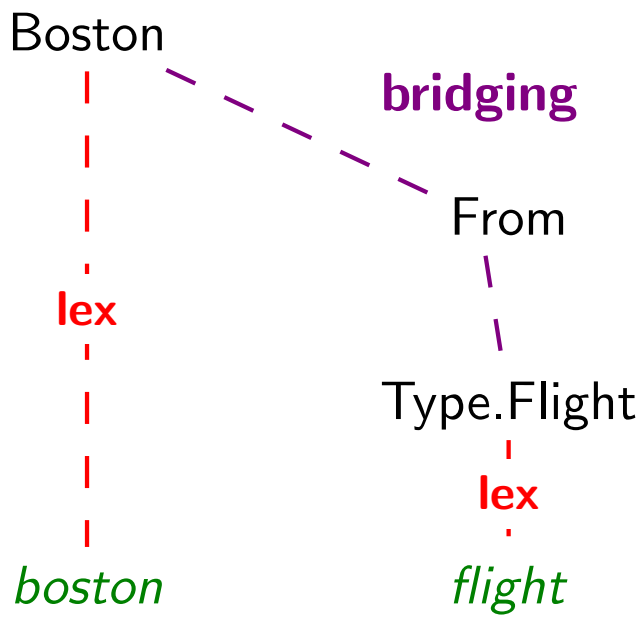


Type.Flight $\sqcap$ To.Boston

Insert any type-matching binary

(bridge first) `Set` $\Rightarrow$ `Set` `Set`

Bridging 1: two unaries



Type.Flight $\sqcap$ To.Boston

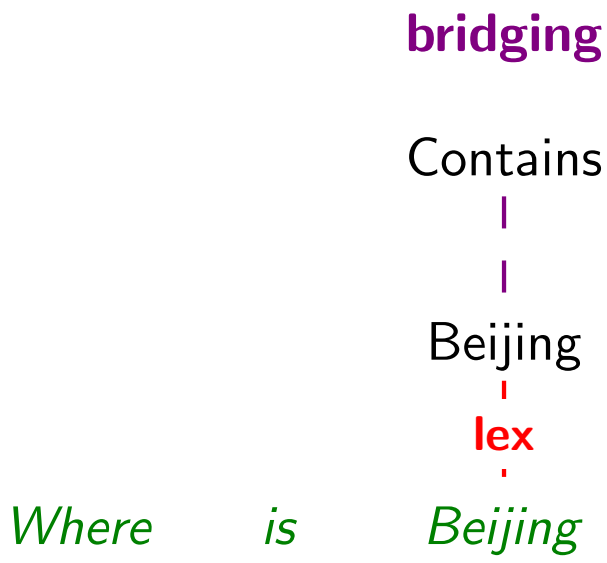
Insert any type-matching binary

(bridge first) `Set` $\Rightarrow$ `Set` `Set`

Learn to choose implicit predicate

Bridging 2: single unary

Bridging 2: single unary



Insert any type-matching binary

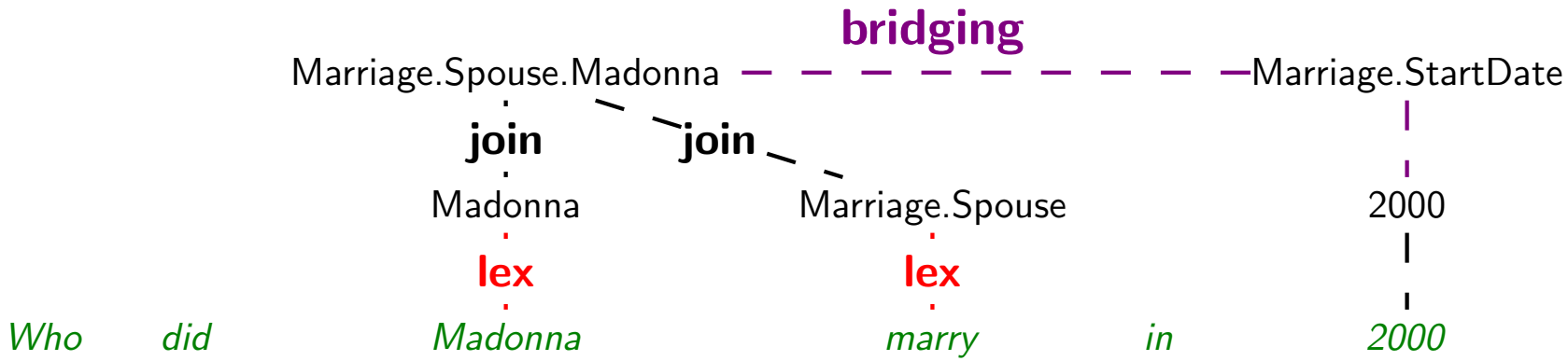
Contains.Beijing

(bridge) `Set` $\Rightarrow$ `Set`

More uncertainty

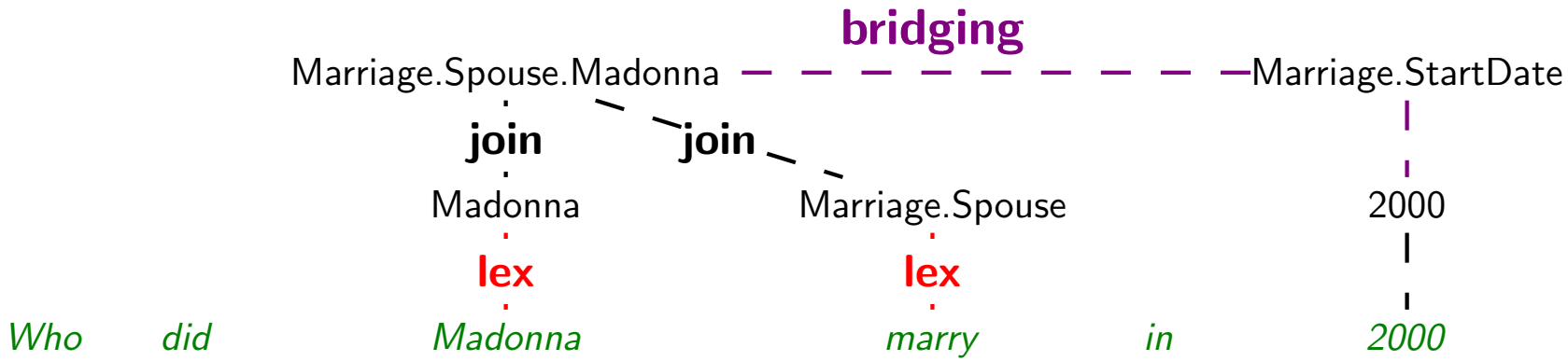
Bridging 3: n-ary relations

Bridging 3: n-ary relations



$$\text{Marriage.}(\text{Spouse.Madonna} \sqcap \text{StartDate.2000})$$

Bridging 3: n-ary relations



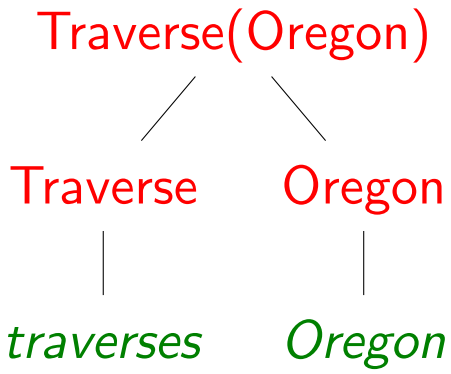
$$\text{Marriage.}(\text{Spouse.Madonna} \sqcap \text{StartDate.2000})$$

Later: a more general solution called **floating parser**

- Wang et al., 2015; Pasupat and Liang, 2015

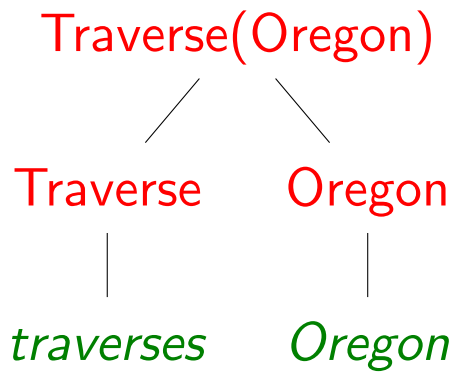
Type raising

Traverses Oregon

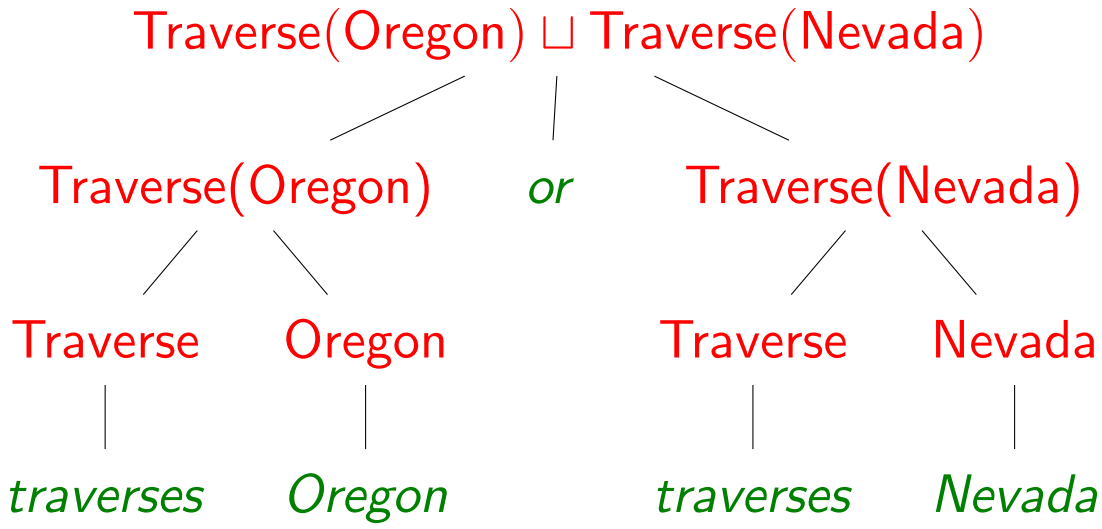


Type raising

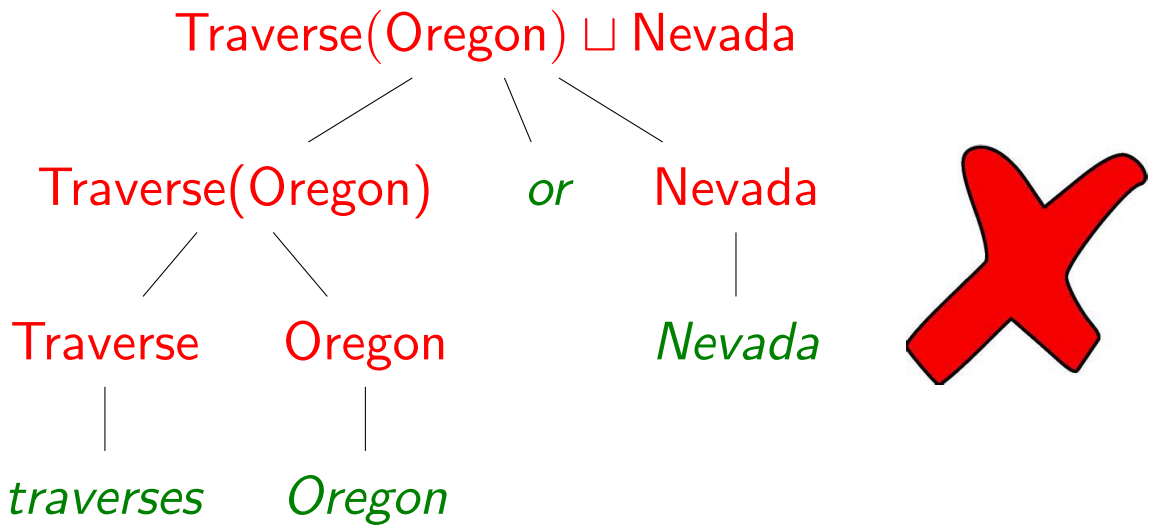
Traverses Oregon



traverses Oregon or traverses Nevada



Type raising



How to apply the binary on multiple arguments?

Solution 1

$$(\mathsf{ConstFn}\ \lambda x.\lambda y.\lambda b.b(x) \sqcup b(y)) \quad \texttt{Disj} \quad \Rightarrow \quad \textit{or}$$

Solution 1

$$(\text{ConstFn } \lambda x.\lambda y.\lambda b.b(x) \sqcup b(y)) \quad \texttt{Disj} \quad \Rightarrow \quad \textit{or}$$

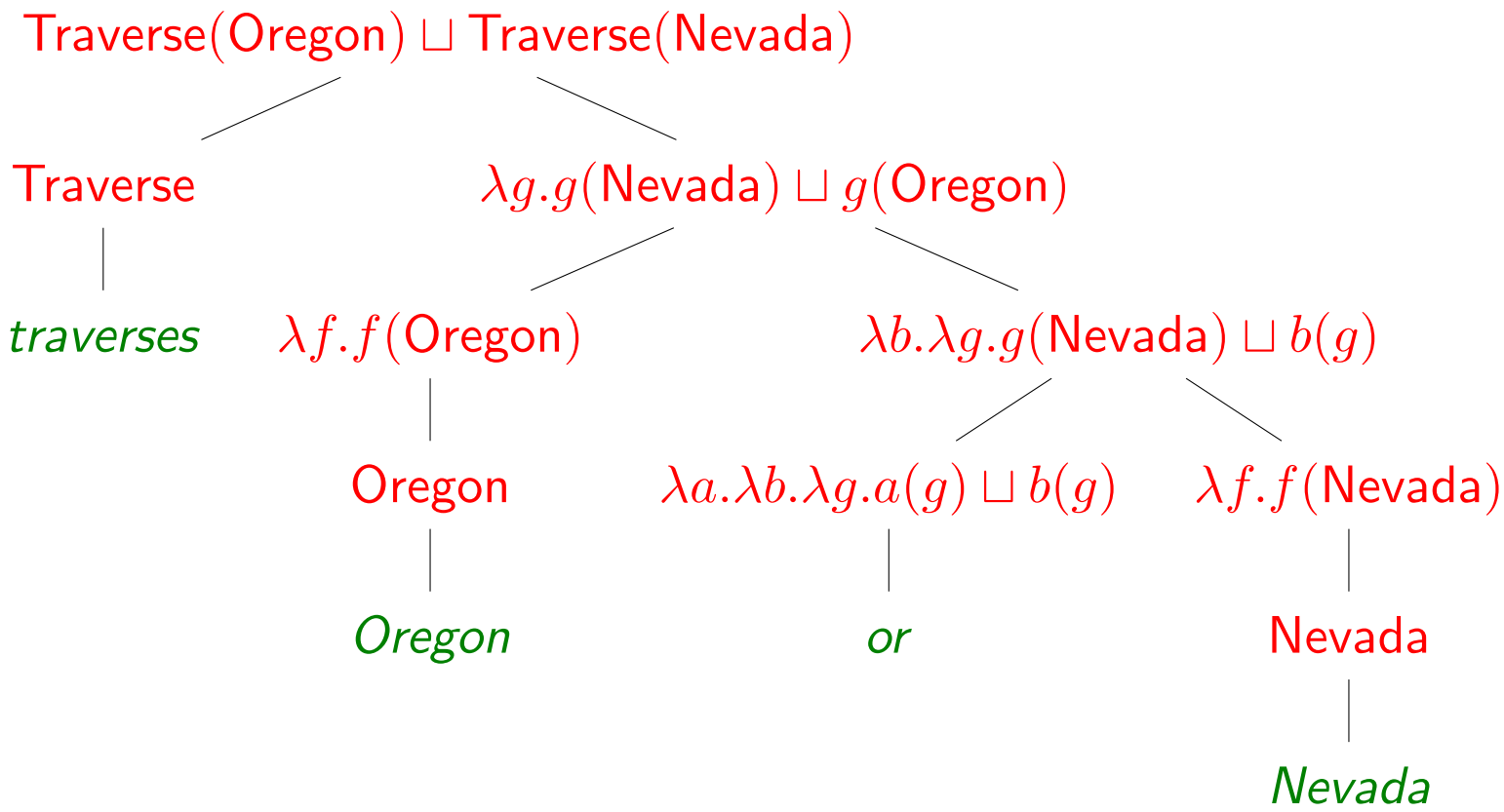
Traverses Oregon or Nevada or Colorado

Another lexical entry?

Type raising

Type raise the entity to a function that is applied on a binary

(ApplyFn $\lambda x.\lambda f.f(x)$)	`FuncSet` $\Rightarrow$ `Set`
(ConstFn $\lambda a.\lambda b.\lambda g.a(g) \sqcup b(g)$)	`Disj` $\Rightarrow$ *or*



Outline

- Compositionality
  - Grammars and derivations
  - Example
  - Lexicon
  - Modeling
  - **CCG**
  - Exercise

CCG

Standard formalism in semantics

CCG

Standard formalism in semantics

Lexicon:

$$\textit{fun} \vdash \texttt{ADJ} : \lambda x.\mathsf{fun}(x)$$
$$\textit{fun} \vdash \texttt{N/N} : \lambda f.\lambda x.\mathsf{fun}(x) \wedge f(x)$$

CCG

Standard formalism in semantics

Lexicon:

$$\textit{fun} \vdash \texttt{ADJ} : \lambda x.\mathsf{fun}(x)$$
$$\textit{fun} \vdash \texttt{N/N} : \lambda f.\lambda x.\mathsf{fun}(x) \wedge f(x)$$

Handful of combinators (grammar)

$$A/B : f \quad B : g \Rightarrow A : f(g) \quad (>)$$
$$B : g \quad A\backslash B : f \Rightarrow A : f(g) \quad (<)$$

CCG

Main difference:

- CCG pushed complexity from grammar to lexicon
- lambda-DCS tries to push it from lexicon to learning

CCG

Main difference:

- CCG pushed complexity from grammar to lexicon
- lambda-DCS tries to push it from lexicon to learning

Rationale:

- CCG originates in linguistics
- Lambda-DCS originates in NLP

CCG

Main difference:

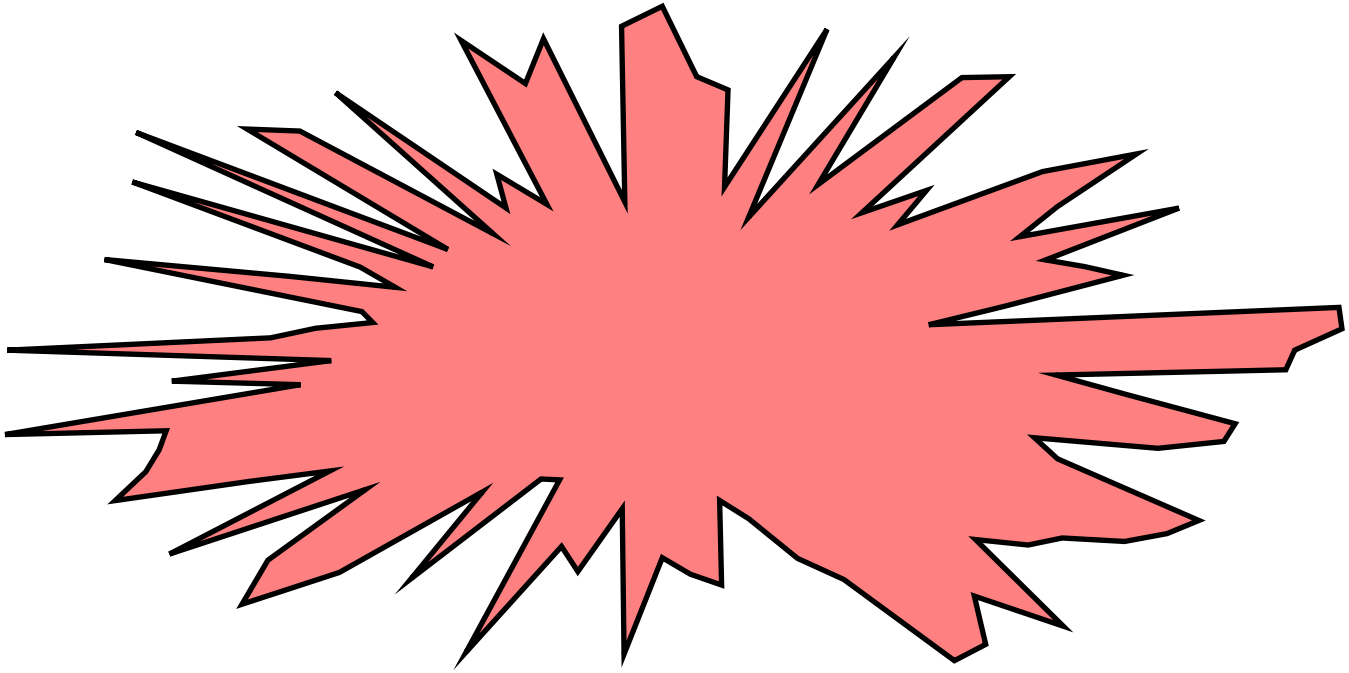
- CCG pushed complexity from grammar to lexicon
- lambda-DCS tries to push it from lexicon to learning

Rationale:

- CCG originates in linguistics
- Lambda-DCS originates in NLP
- CCG-based semantic parsing addressed many issues with type raising, factored lexicons, etc. [Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011]

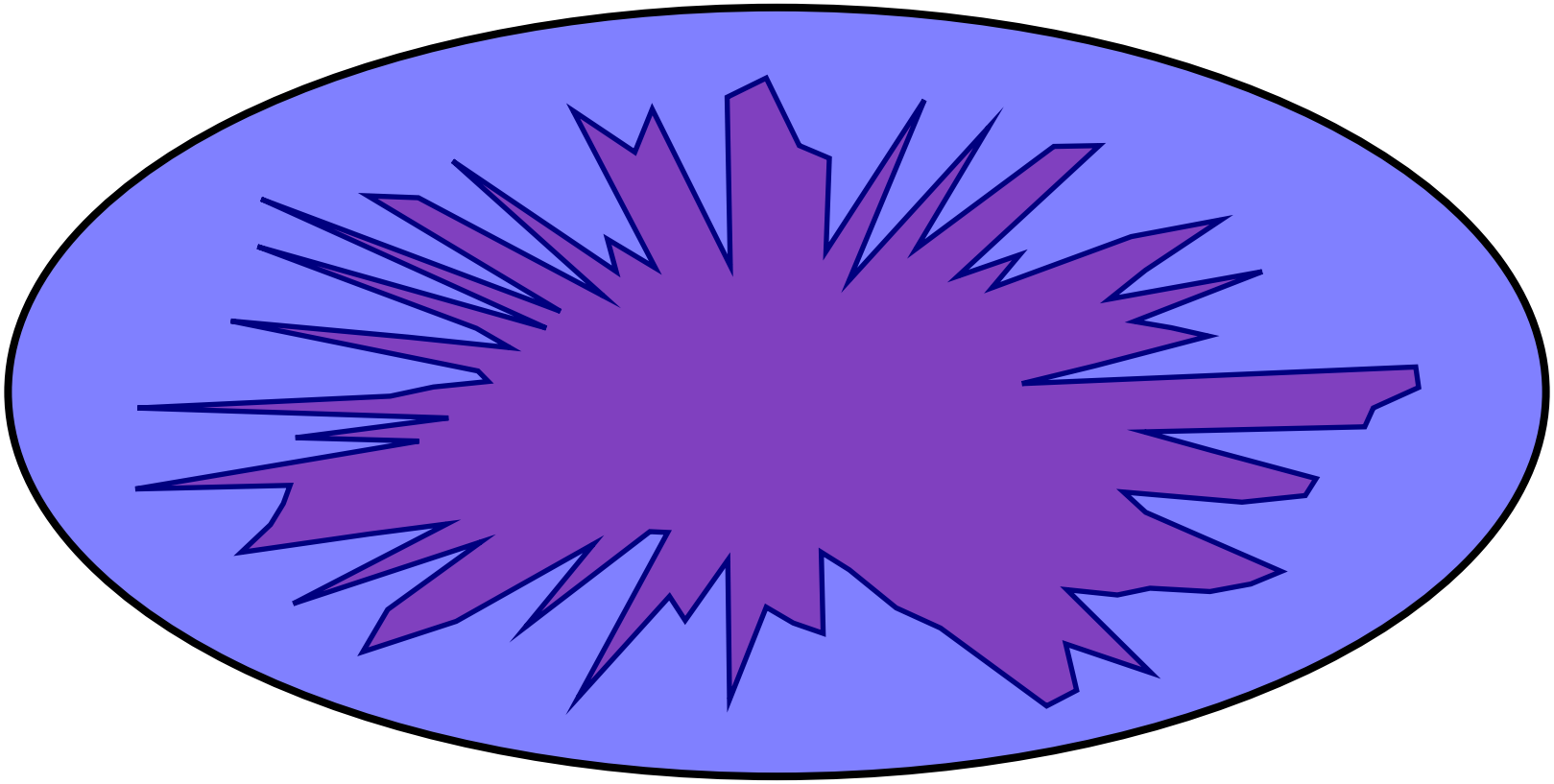
Philosophy: simple grammar and lexicon

- Capturing correct derivations only requires complex rules



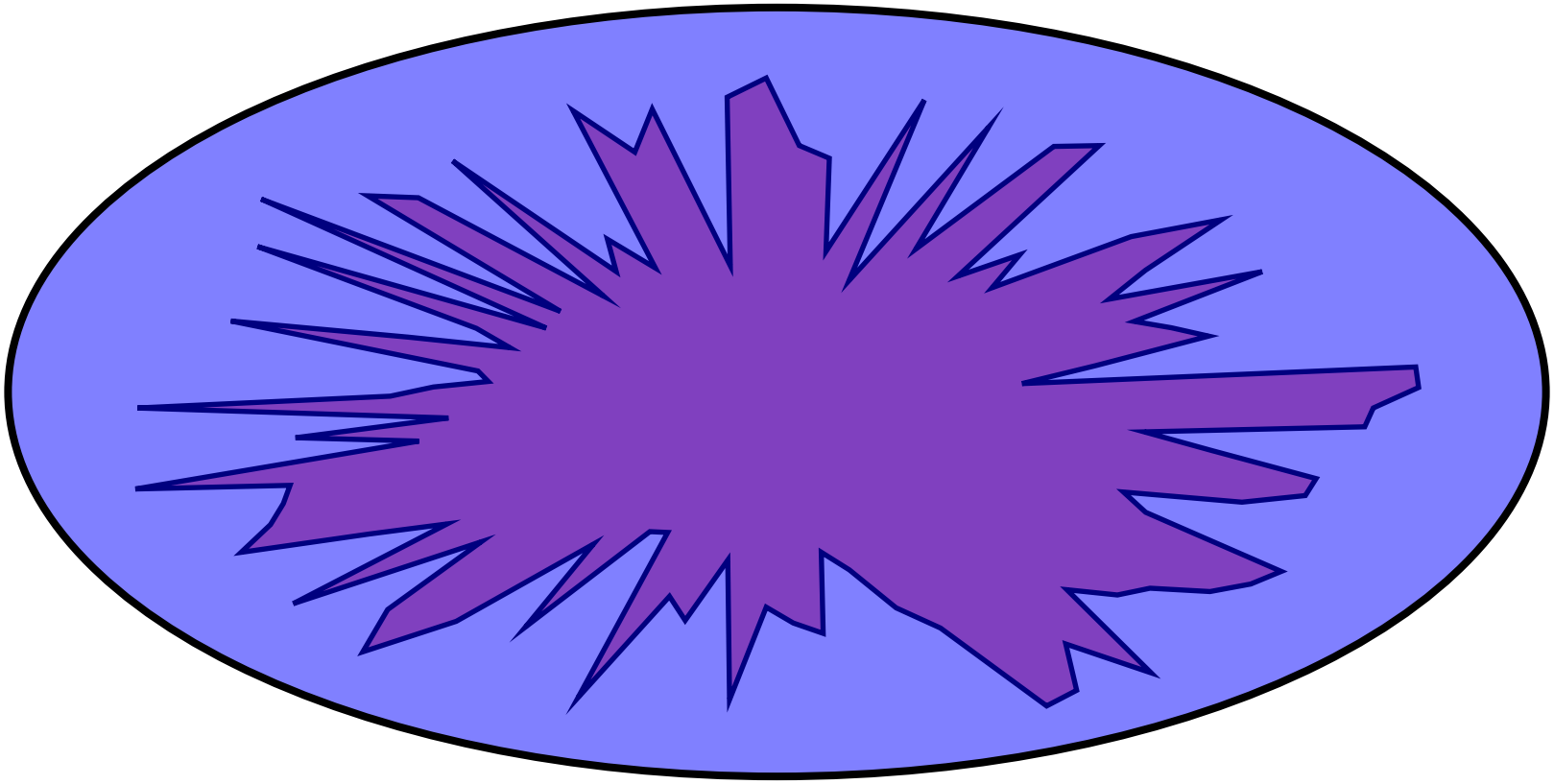
Philosophy: simple grammar and lexicon

- Capturing correct derivations only requires complex rules
- Simple rules generate overapproximation of good derivations



Philosophy: simple grammar and lexicon

- Capturing correct derivations only requires complex rules
- Simple rules generate overapproximation of good derivations



Disambiguate at learning time!

Outline

- Compositionality
  - Grammars and derivations
  - Example
  - Lexicon
  - Modeling
  - CCG
  - **Exercise**

Exercise

Write a grammar that parses utterances

You can use class2_demo.lexicon and class2_demo.grammar that are on the website

They are also at /home/shared/esslli/semparse

You can also start with ConstantFn entries that parse the entire uttearnce and decompose

Exercise

State with the largest area

Top 5 cities by area

Major cities with at least 140,000 inhabitants

How many states have capitals with at least 200,000 inhabitants

State bordering Oregon and Nevada

Check if the grammar parses the sentence:

States bordering Oregon and Nevada and Arizona

Summary

- Grammars build derivations which compositionally provide logical forms

- Lexicons map phrases to basic logical forms

- Learning lexicons is hard so we prefer to push the complexity to learning