Semantic Parsing

Full understanding?

- We're doing natural language *understanding*, right?
- Are we there yet? Do we fully *understand*?
  - With VSMs? Dependency parses? Relation extraction?
  - Arguably, all are steps toward to NLU … but are they sufficient?
- What aspects of meaning are we still unable to capture?
  - Higher-arity relations, events with multiple participants, temporal aspects, negation, disjunction, quantification, propositional attitudes, modals, …

Logic games from LSAT (& old GRE)

Six sculptures — C, D, E, F, G, H — are to be exhibited in rooms 1, 2, and 3 of an art gallery.

- Sculptures C and E may not be exhibited in the same room.
- Sculptures D and G must be exhibited in the same room.
- If sculptures E and F are exhibited in the same room, no other sculpture may be exhibited in that room.
- At least one sculpture must be exhibited in each room, and no more than three sculptures may be exhibited in any room.

If sculpture D is exhibited in room 3 and sculptures E and F are exhibited in room 1, which of the following may be true?

A. Sculpture C is exhibited in room 1.
B. Sculpture H is exhibited in room 1.
C. Sculpture G is exhibited in room 2.
D. Sculptures C and H are exhibited in the same room.
E. Sculptures G and F are exhibited in the same room.

Travel reservations

Yes, hi, I need to book a flight for myself and my husband from Boston to SFO, or Oakland would be OK too. We need to fly out on Friday the 12th, and then I could come back on Sunday evening or Monday morning, but he won't return until Wednesday the 18th, because he's staying for business. No flights with more than one stop, and we don't want to fly on United because we hate their guts.

SHRDLU (Winograd 1972)

Find a block which is taller than the one you are holding and put it into the box.

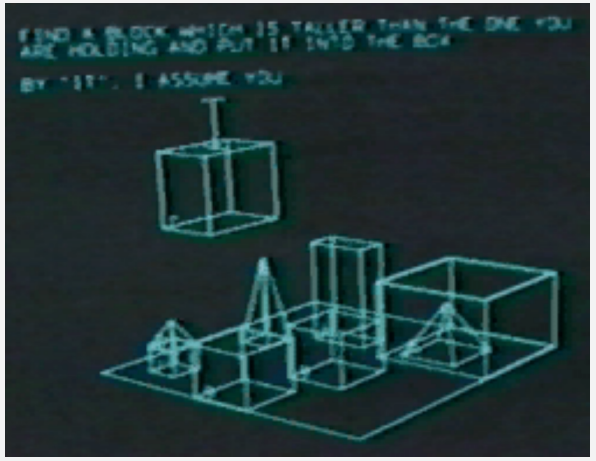
OK.

How many blocks are not in the box?

FOUR OF THEM.

Is at least one of them narrower than the one which I told you to pick up?

YES, THE RED CUBE.



http://youtube.com/watch?v=8SvD-lNg0TA

http://hci.stanford.edu/winograd/shrdlu/

CHAT-80

- Developed 1979-82 by Fernando Pereira & David Warren
- Proof-of-concept natural language interface to database
- Could answer questions about geography
- Implemented in Prolog
- Hand-built lexicon & grammar
- Highly influential NLIDB system

Things you could ask CHAT-80

- Is there more than one country in each continent?
- What countries border Denmark?
- What are the countries from which a river flows into the Black_Sea?
- What is the total area of countries south of the Equator and not in Australasia?
- Which country bordering the Mediterranean borders a country that is bordered by a country whose population exceeds the population of India?
- How far is London from Paris? I don't understand!

The CHAT-80 database

```
% Facts about countries.
% country(Country, Region, Latitude, Longitude,
%   Area(sqmiles), Population, Capital, Currency)
country(andorra, southern_europe, 42, -1, 179, 25000,
andorra_la_villa, franc_peseta).
country(angola, southern_africa, -12, -18, 481351,
5810000, luanda, ?).
country(argentina, south_america, -35, 66, 1072067,
23920000, buenos_aires, peso).

capital(C,Cap) :- country(C,_,_,_,_,_,Cap,_).
```

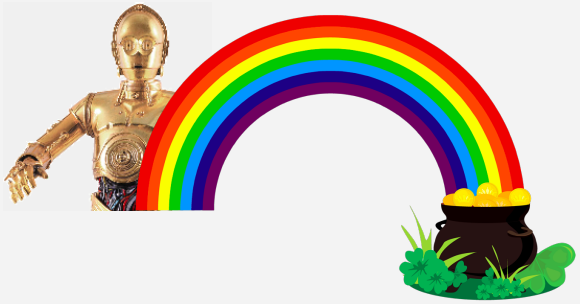
The CHAT-80 grammar

```
/* Sentences */
sentence(S) --> declarative(S), terminator(.) .
sentence(S) --> wh_question(S), terminator(?) .
sentence(S) --> yn_question(S), terminator(?) .
sentence(S) --> imperative(S), terminator(!) .

/* Noun Phrase */
np(np(Agmt,Pronoun,[]),Agmt,NPCase,def,_,Set,Nil) -->
  {is_pp(Set)},
  pers_pron(Pronoun,Agmt,Case),
  {empty(Nil), role(Case,decl,NPCase)}.

/* Prepositional Phrase */
pp(pp(Prep,Arg),Case,Set,Mask) -->
  prep(Prep),
  {prep_case(NPCase)},
  np(Arg,_,NPCase,_,Case,Set,Mask).
```

Precision vs. robustness

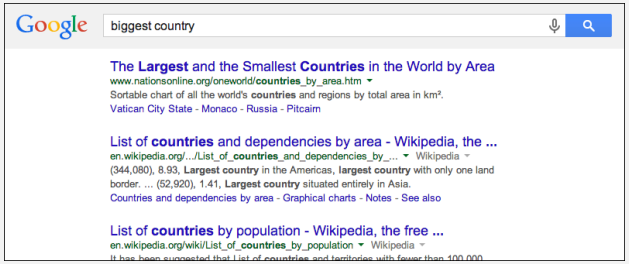
Precise, complete understanding

SHRDLU CHAT-80



Brittle, narrow coverage

Robust, broad coverage



Fuzzy, partial understanding

Carbon emissions

Which country has the highest CO2 emissions?
What about highest per capita?
Which had the biggest increase over the last five years?
What fraction was from European countries?

Baseball statistics

Pitchers who have struck out four batters in one inning
Players who have stolen at least 100 bases in a season
Complete games with fewer than 90 pitches
Most home runs hit in one game

Voice commands

How do I get to the Ferry Building by bike
Book a table for four at Nopa on Friday after 9pm
Text my wife I'm going to be twenty minutes late
Add House of Cards to my Netflix queue at the top

Semantic parsing

If we want to understand natural language *completely* and *precisely*, we need to do semantic parsing.

That is, translate natural language into a formal meaning representation on which a machine can act.

First, we need to define our goal.

What should we choose as our target output representation of meaning?

Database queries

To facilitate data exploration and analysis, you might want to parse natural language into database queries:

which country had the highest carbon emissions last year

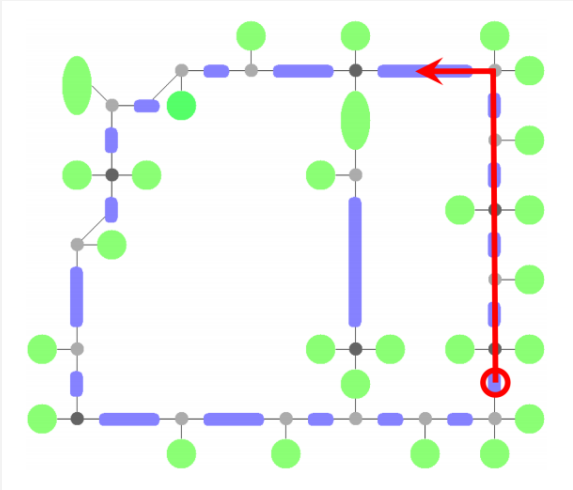
```
SELECT    country.name
FROM      country, co2_emissions
WHERE     country.id = co2_emissions.country_id
AND       co2_emissions.year = 2013
ORDER BY  co2_emissions.volume DESC
LIMIT     1;
```

Robot control

For a robot control application, you might want a custom-designed procedural language:

Go to the third junction and take a left.

```
(do-sequentially
  (do-n-times 3
    (do-sequentially
      (move-to forward-loc)
      (do-until
        (junction current-loc)
        (move-to forward-loc))))
  (turn-left))
```



Intents and arguments

For smartphone voice commands, you might want relatively simple meaning representations, with *intents* and *arguments*:

directions to SF by train

```
(TravelQuery
  (Destination /m/0d6lp)
  (Mode TRANSIT))
```

angelina jolie net worth

```
(FactoidQuery
  (Entity /m/0f4vbz)
  (Attribute /person/net_worth))
```

weather friday austin tx

```
(WeatherQuery
  (Location /m/0vzm)
  (Date 2013-12-13))
```

text my wife on my way

```
(SendMessage
  (Recipient 0x31cbf492)
  (MessageType SMS)
  (Subject "on my way"))
```

play sunny by boney m

```
(PlayMedia
  (MediaType MUSIC)
  (SongTitle "sunny")
  (MusicArtist /m/017mh))
```

is REI open on sunday

```
(LocalQuery
  (QueryType OPENING_HOURS)
  (Location /m/02nx4d)
  (Date 2013-12-15))
```

Demo: wit.ai

For a very simple NLU system based on identifying intents and arguments, check out this startup:

http://wit.ai/

First-order logic

Blackburn & Bos make a strong argument for using first-order logic as the meaning representation.

Powerful, flexible, general.

Can subsume most other representations as special cases.

John walks	`walk(john)`
John loves Mary	`love(john, mary)`
Every man loves Mary	$\forall$`x (man(x)` $\rightarrow$ `love(x, mary))`

(Lambda calculus will be the vehicle; first-order logic will be the final destination.)

FOL syntax, in a nutshell

- FOL symbols
  - Constants: john, mary
  - Predicates & relations: man, walks, loves
  - Variables: *x*, *y*
  - Logical connectives: $\wedge$ $\vee$ $\neg$ $\rightarrow$
  - Quantifiers: $\forall$ $\exists$
  - Other punctuation: parens, commas

"content words" (user-defined)

"function words"

- FOL formulae
  - Atomic formulae: loves(john, mary)
  - Connective applications: man(john) $\wedge$ loves(john, mary)
  - Quantified formulae: $\exists x$ (man(x))

An NLU pipeline

- English sentences

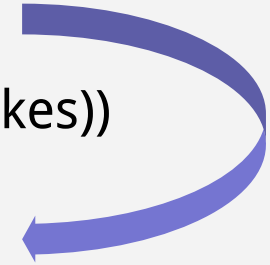
  John smokes. Everyone who smokes snores.

- Syntactic analysis

  (S (NP John) (VP smokes))

- Semantic analysis

  smoke(john)



Focus of semantic parsing

- Inference

  $\forall$ x.smoke(x) $\rightarrow$ snore(x), smoke(john)

  $\Rightarrow$ snore(john)

From language to logic

How can we design a general algorithm for translating from natural language into logical formulae?

John walks	walk(john)
John loves Mary	love(john, mary)
A man walks	$\exists$ x.man(x) $\wedge$ walk(x)
A man loves Mary	$\exists$ x.man(x) $\wedge$ love(x, mary)
John and Mary walk	walk(john) $\wedge$ walk(mary)
Every man walks	$\forall$ x.man(x) $\rightarrow$ walk(x)
Every man loves a woman	$\forall$ x.man(x) $\rightarrow$ $\exists$ y.woman(y) $\wedge$ love(x, y)

We don't want to simply memorize these pairs, because that won't generalize to new sentences.

Machine translation (MT)

How can we design a general algorithm for translating from one language into another?

John walks	Jean marche
John loves Mary	Jean aime Marie
A man walks	Un homme marche
A man loves Mary	Un homme aime Marie
John and Mary walk	Jean et Marie marche
Every man walks	Chaque homme marche
Every man loves a woman	Chaque homme aime une femme

In MT, we break the input into pieces, translate the pieces, and then put the pieces back together.

A logical lexicon (first attempt)

John walks	walk(john)
John loves Mary	love(john, mary)
A man walks	$\exists$x.man(x) $\wedge$ walk(x)
A man loves Mary	$\exists$x.man(x) $\wedge$ love(x, mary)
John and Mary walk	walk(john) $\wedge$ walk(mary)
Every man walks	$\forall$x.man(x) $\rightarrow$ walk(x)
Every man loves a woman	$\forall$x.man(x) $\rightarrow$ $\exists$y.woman(y) $\wedge$ love(x, y)

John : john
Mary : mary

walks : walk(?)
loves : love(?, ?)

man : man(?)
woman : woman(?)

and : $\wedge$

a : $\exists$x.? $\wedge$?
every : $\forall$x.? $\rightarrow$?

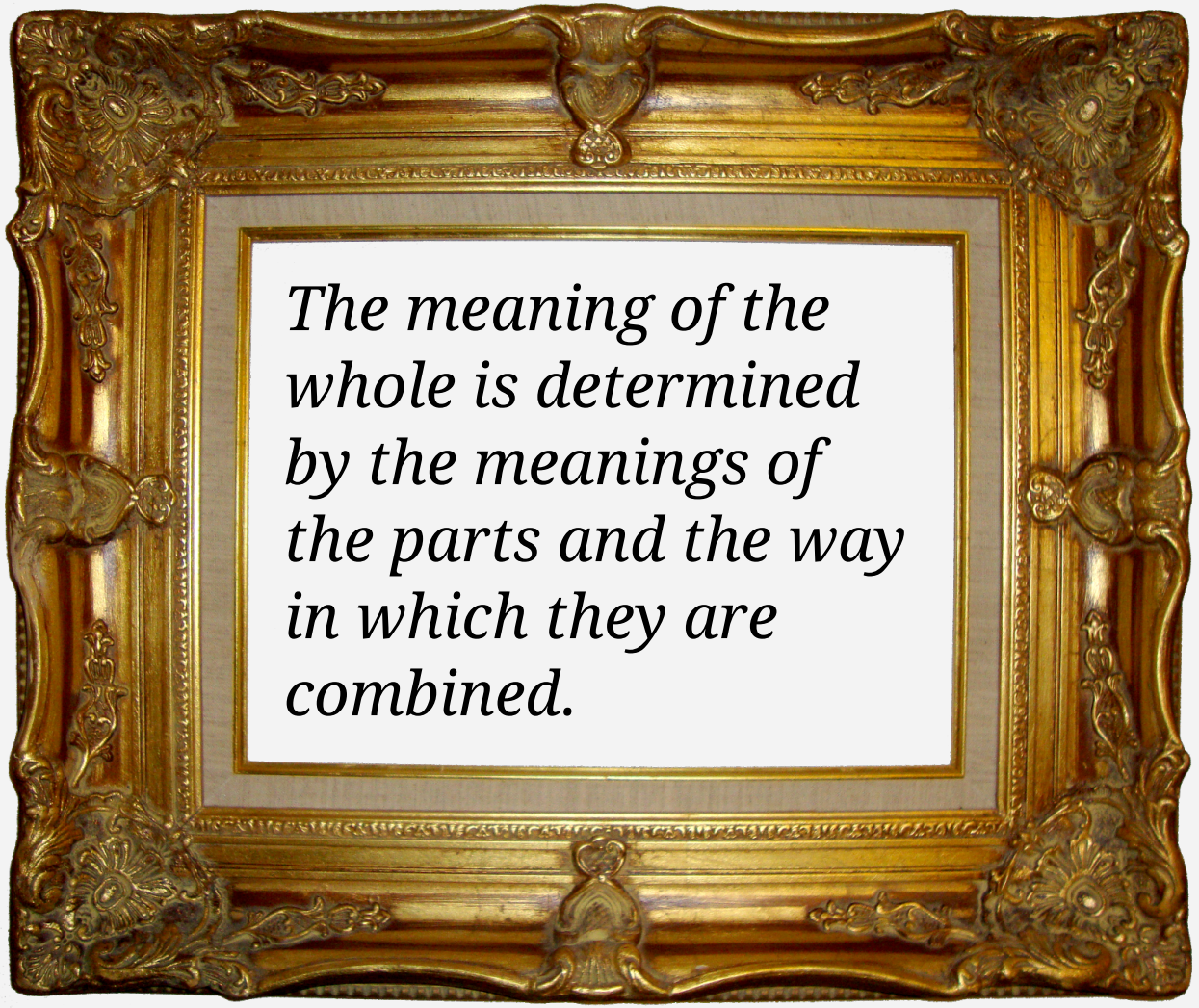
Compositional semantics

Now how do we put the pieces back together?

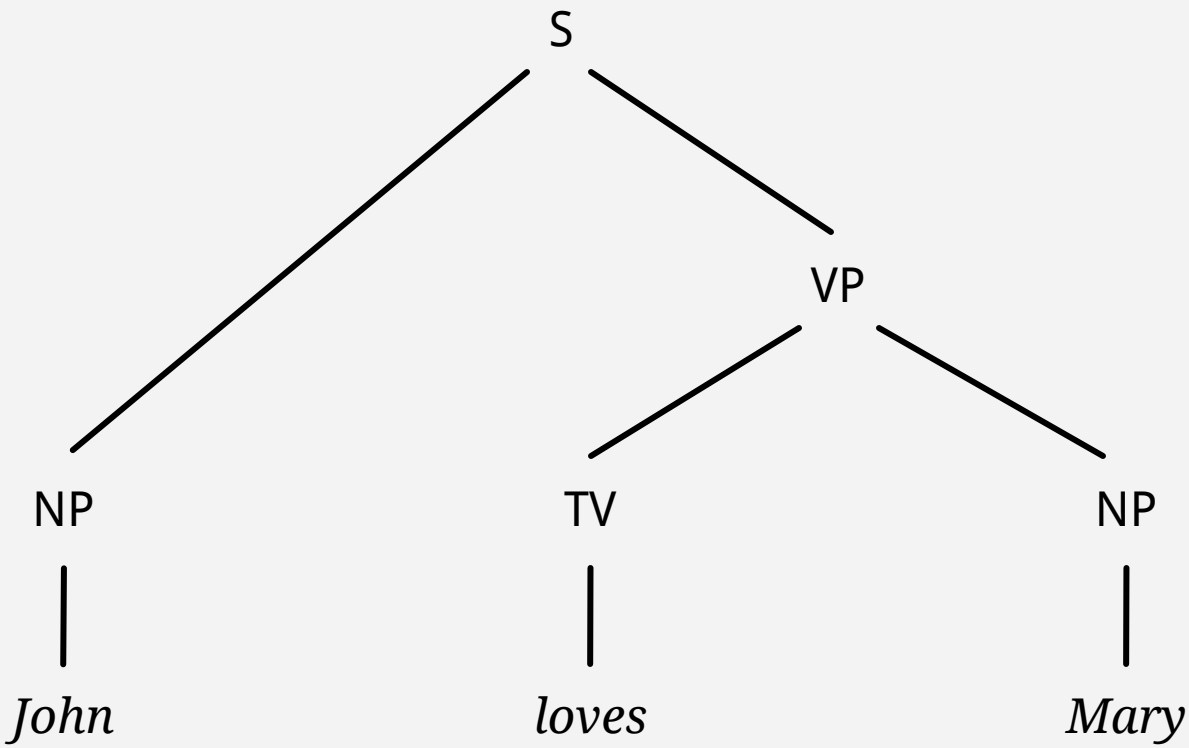
Idea: syntax-driven compositional semantics

1. Parse sentence to get syntax tree
2. Look up the semantics of each word in lexicon
3. Build the semantics for each constituent bottom-up, by combining the semantics of its children

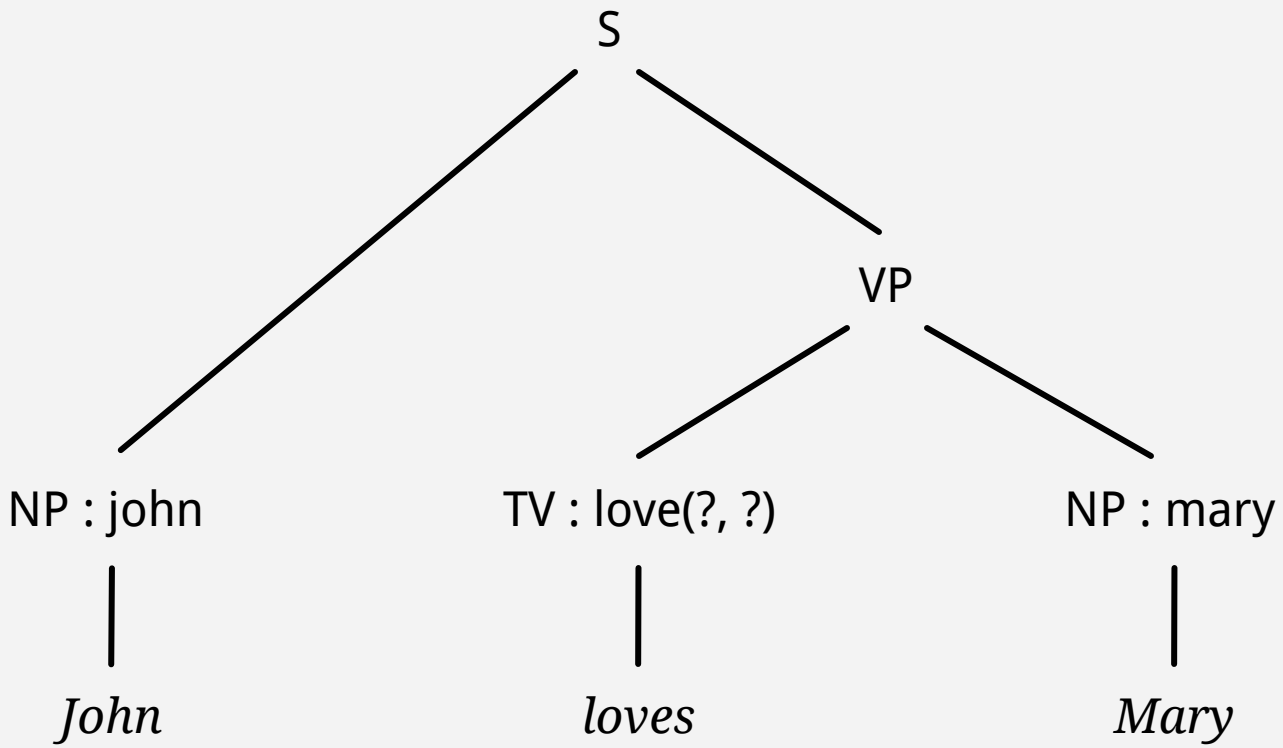
Principle of compositionality



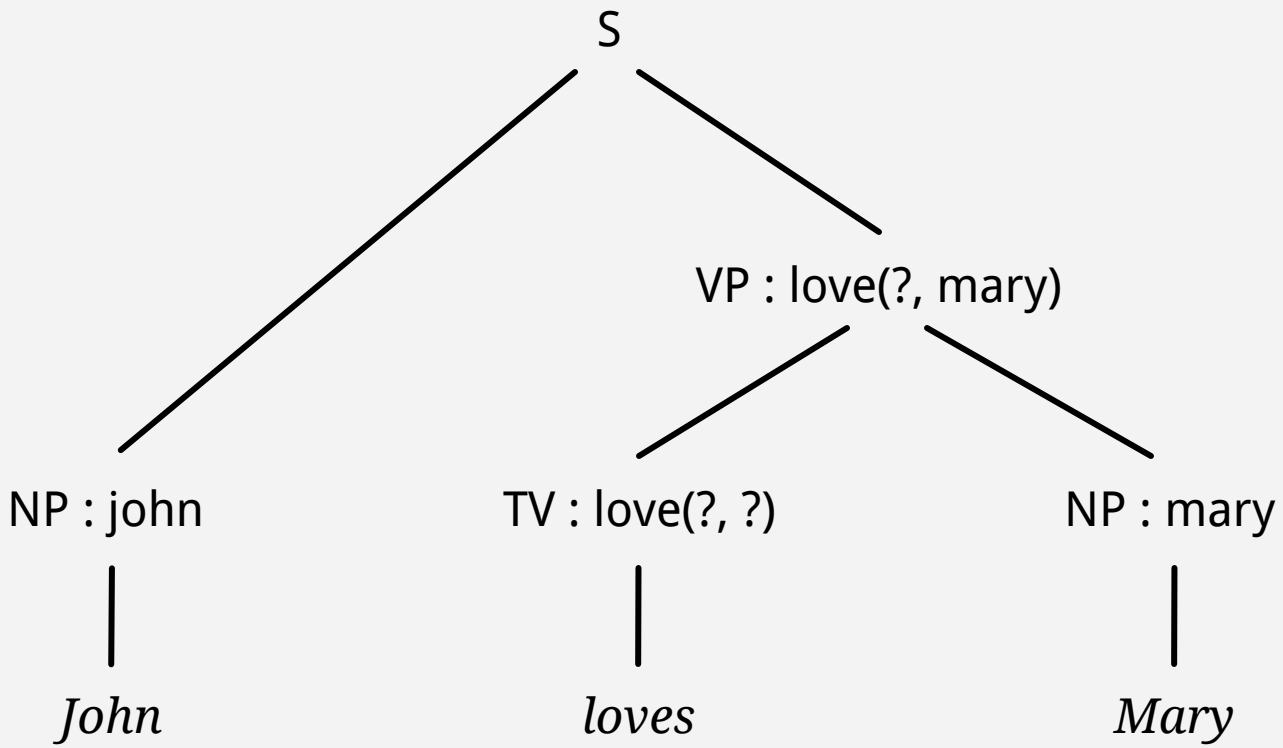
Example: syntactic analysis



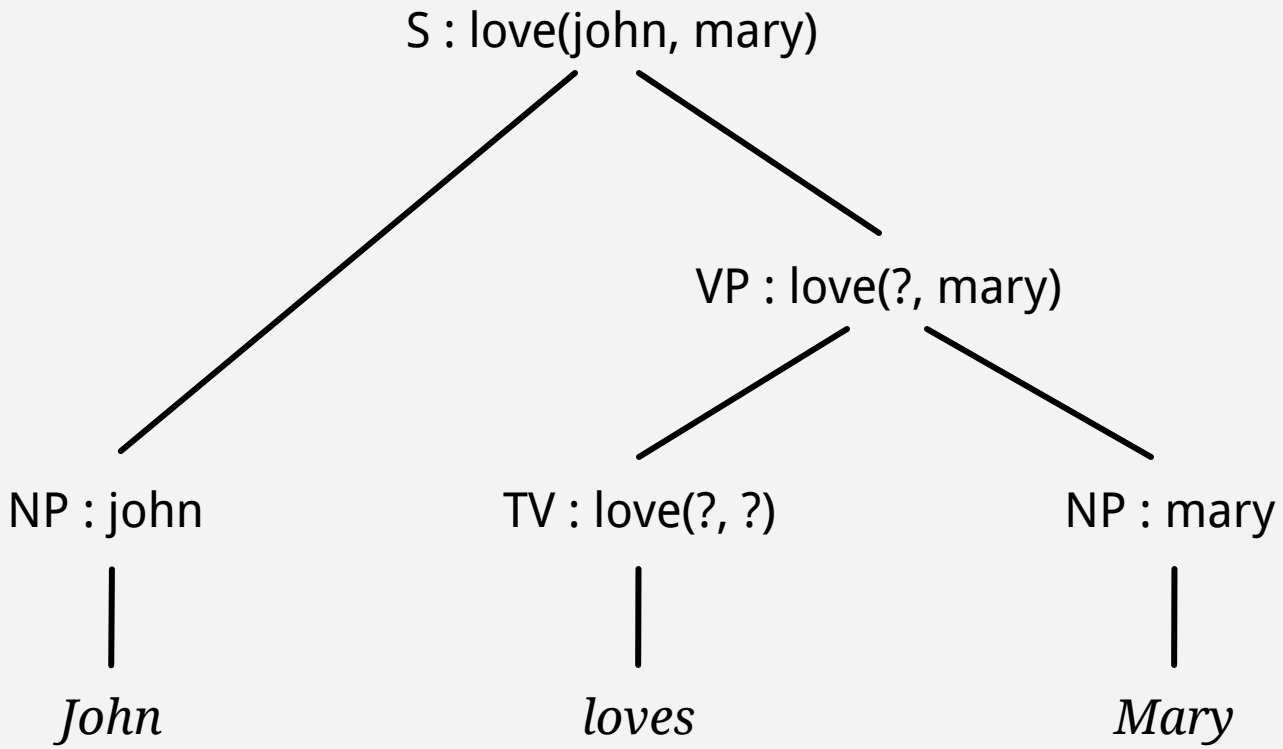
Example: semantic lexicon



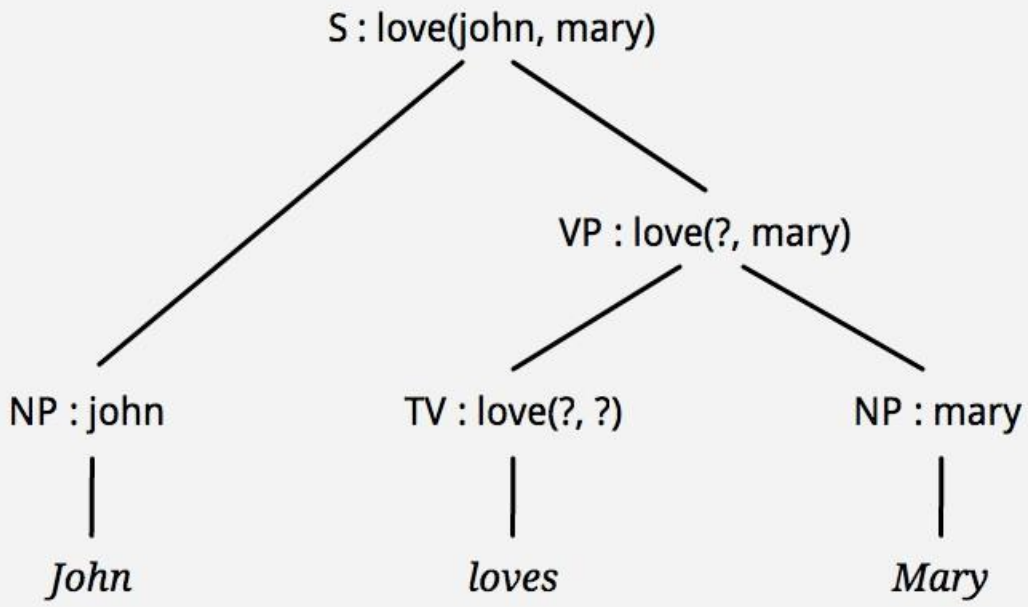
Example: semantic composition



Example: semantic composition



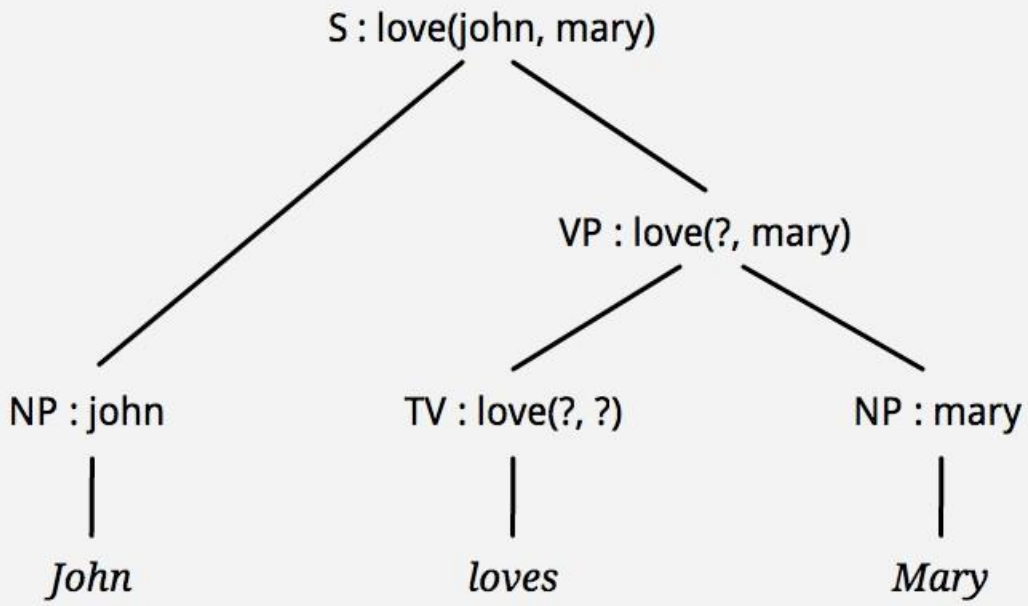
Compositionality



The meaning of the sentence is constructed from:

- the meaning of the words (i.e., the lexicon)
- paralleling the syntactic construction (i.e., the semantic rules)

Systematicity



How do we know how to construct the VP?

love(?, mary) OR love(mary, ?)

How can we *specify* in which way the bits & pieces combine?

Systematicity (continued)

- How do we want to represents parts of formulae?

  E.g. for the VP *loves Mary* ?

love(?, mary)	bad: not FOL
love(x, mary)	bad: no control over free variable

- Familiar well-formed formulae (sentences)

$\forall$ x (love(x, mary))	*Everyone loves Mary*
$\exists$ x (love(mary, x))	*Mary loves someone*

Lambda abstraction

- Add a new operator λ to bind free variables

  λx.love(x, mary) *loves Mary*

- The new meta-logical symbol λ marks missing information in the object language (λ-)FOL
- We *abstract* over x
- Just like in programming languages!

  Python: `lambda x: x % 2 == 0`

  Ruby: `lambda {|x| x % 2 == 0}`

- How do we combine these new formulae and terms?

Super glue

- Gluing together formulae/terms with function application

  (λx.love(x, mary)) @ john

  (λx.love(x, mary))(john)

- How do we get back to the familiar love(john, mary) ?

- FA triggers a simple operation: *beta reduction*

  replace the λ-bound variable by the argument throughout the body

Beta reduction

(λx.love(x, mary)) (john)

1. Strip off the λ prefix

   (love(x, mary)) (john)

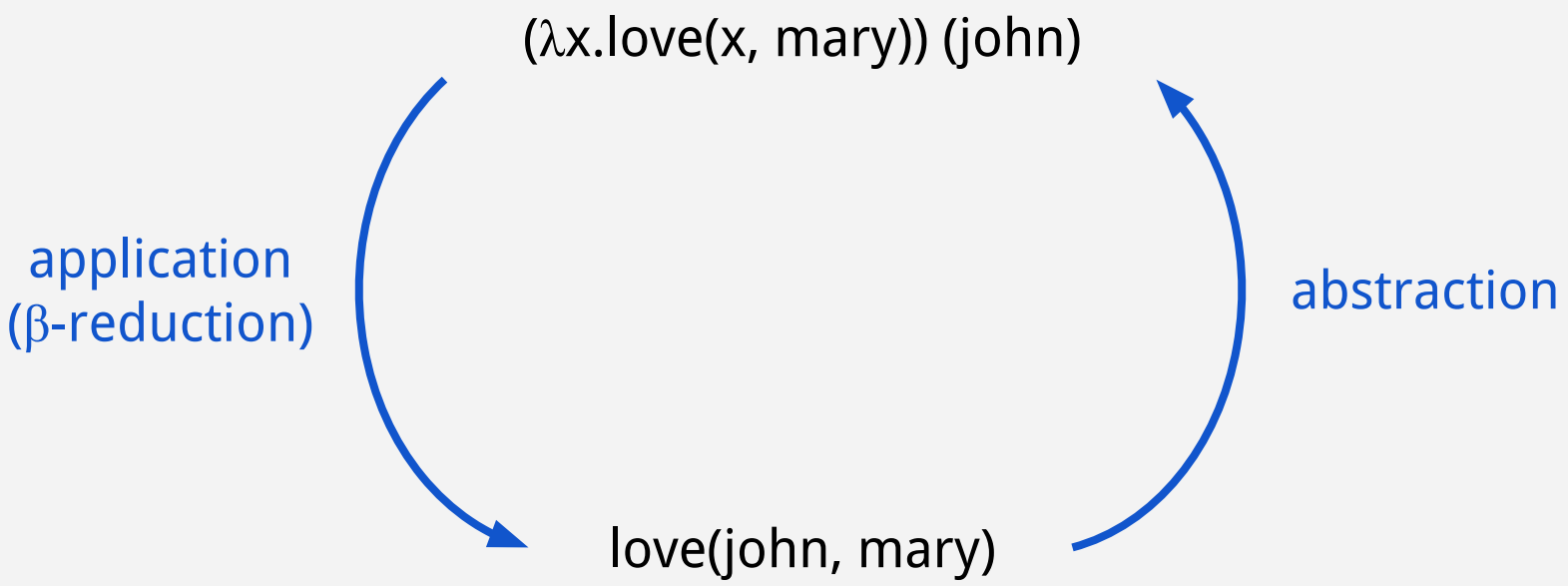
2. Remove the argument

   love(x, mary)

3. Replace all occurrences of λ-bound variable by argument

   love(john, mary)

Application vs. abstraction



Semantic construction with lambdas

S : (λx.love(x, mary))(john)
= love(john, mary)

VP : (λy.λx.love(x, y))(mary)
= λx.love(x, mary)

NP : john

TV : λy.λx.love(x, y)

NP : mary

John

loves

Mary

A semantic grammar

Lexicon

John ← NP : john

Mary ← NP : mary

loves ← TV : λy.λx.love(x, y)

Composition rules

VP : f(a) → TV : f NP : a

S : f(a) → NP : a VP : f

Note the *semantic attachments* — these are augmented CFG rules

Note the use of function application to glue things together

For binary rules, four possibilities for semantics of parent (what?)

Montague semantics

This approach to formal semantics was pioneered by Richard Montague (1930-1971)

"... I reject the contention that an important theoretical difference exists between formal and natural languages ..."

nltk.sem [Garrette & Klein 2008]

The nltk.sem package contains Python code for:

- First-order logic & typed lambda calculus
- Theorem proving, model building, & model checking
- DRT & DRSs
- Cooper storage, hole semantics, glue semantics
- Linear logic
- A (partial) implementation of Chat-80!

http://nltk.googlecode.com/svn/trunk/doc/api/nltk.sem-module.html

nltk.sem.logic

```
>>> import nltk
>>> from nltk.sem import logic
>>> logic.demo()
>>> parser = logic.LogicParser(type_check=True)

>>> man = parser.parse("\ y.man(y)")
>>> woman = parser.parse("\ x.woman(x)")
>>> love = parser.parse("\ R x.R(\ y.love(x,y))")
>>> every = parser.parse("\ P Q.all x.(P(x) -> Q(x))")
>>> some = parser.parse("\ P Q.exists x.(P(x) & Q(x))")

>>> every(man).simplify()
<LambdaExpression \Q.all x.(man(x) -> Q(x))>

>>> love(some(woman)).simplify()
<LambdaExpression \x.exists z.(woman(z) & love(x, z))>

>>> every(man)(love(some(woman))).simplify()
<AllExpression all x.(man(x) -> exists z.(woman(z) & love(x, z)))>
```

What's missing?

OK, this all seems super duper, but … what's missing?

Can we solve these NLU challenges yet?

Why not?

Six sculptures — C, D, E, F, G, H — are to be exhibited in rooms 1, 2, and 3 of an art gallery.

- Sculptures C and E may not be exhibited in the same room.
- Sculptures D and G must be exhibited in the same room.
- If sculptures E and F are exhibited in the same room, no other sculpture may be exhibited in that room.
- At least one sculpture must be exhibited in each room, and no more than three sculptures may be exhibited in any room.

If sculpture D is exhibited in room 3 and sculptures E and F are exhibited in room 1, which of the following may be true?

A. Sculpture C is exhibited in room 1.
B. Sculpture H is exhibited in room 1.
C. Sculpture G is exhibited in room 2.
D. Sculptures C and H are exhibited in the same room.
E. Sculptures G and F are exhibited in the same room.

Yes, hi, I need to book a flight for myself and my husband from Boston to SFO, or Oakland would be OK too. We need to fly out on Friday the 12th, and then I could fly back on Sunday evening or Monday morning, but he won't return until Wednesday the 18th, because he's staying for business. No flights with more than one stop, and we don't want to fly on United because we hate their guts.

Zettlemoyer & Collins 2005

- Won Best Paper award at UAI-2005
- Initiated “modern era” of semantic parsing
- First of many semantic parsing papers from Zettlemoyer & collaborators
- Worthy of very close reading:
  if you really understand this paper,
  you really understand semantic parsing!

The problem

Learning to map sentences to logical form:

Texas borders Kansas

```
borders(texas, kansas)
```

Potential applications

- Natural language interfaces to databases
- Question answering
- Dialogue systems

Some training examples

Input: *What states border Texas?*
Output: `λx.state(x) ∧ borders(x, texas)`

Input: *What is the largest state?*
Output: `argmax(λx.state(x), λx.size(x))`

Input: *What states border the largest state?*
Output:
```
λx.state(x) ∧ borders(x,
  argmax(λy.state(y), λy.size(y)))
```

The approach

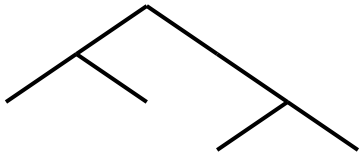
Learn lexical information (syntax & semantics) for words:

Texas := syntax = `NP` : semantics = `texas`

states := syntax = `N` : semantics = `λx.state(x)`

Learn to parse to logical form:

Input: *What states border Texas?*



Output: `λx.state(x) ∧ borders(x, texas)`

Background

- Combinatory categorial grammar (CCG)
  - Lexicon
  - Parsing rules (combinators)
- Probabilistic CCG (PCCG)

Combinatory categorial grammar (CCG)

- A grammar formalism introduced by Steedman (1996)
- A rich set of syntactic categories
  - In fact, an *infinite* set — vs. a few dozen in the Penn Treebank
  - Atomic categories + functional categories which combine them
- A small number of ways of combining categories
  - Chiefly function application, plus composition, type-raising, …
  - Vs. hundreds of rules in the Penn Treebank
- Facilitates tight coupling between syntax & semantics

The intuition behind CCG

- Many words & phrases behave like functions
- E.g., red is like a function that maps $ball$ into $red\ ball$
- $ball$ and $red\ ball$ are both noun-like things
- So, adjectives are like functions from nouns to nouns
- So let the syntactic category for adjectives reflect this

CCG categories

- Atomic categories: S, NP, N
  - S: sentence [semantic type `Bool`]: *Texas is a state*
  - NP: noun phrase [type `Ind`]: *Texas*
  - N: noun [type `Ind → Bool`]: *state*
- Functional categories: X/Y and X\Y
  - X/Y: takes arg of category Y to the right, returns category X
  - X\Y: takes arg of category Y to the left, returns category X
  - N/N: adjective [type `((Ind→Bool)→(Ind→Bool)`]: *big*
  - (S\NP)/NP: transitive verb [type `(Ind→(Ind→Bool)`]: *borders*

CCG lexicon

Words	Category Syntax : Semantics
Texas	`NP : texas`
state	`N : λx.state(x)`
borders	`(S\NP)/NP : λx.λy.borders(y, x)`
what	`(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)`

Parsing rules (combinators)

Forward application

```
X/Y : f        Y : a      ➡ X : f(a)

(S\NP)/NP :    NP :       ➡ S\NP :
λx.λy.borders(y, x)  texas     λy.borders(y, texas)
```

Reverse application

```
Y : a      X\Y : f              ➡ X : f(a)

NP :       (S\NP) :             ➡ S :
kansas     λy.borders(y, texas)      borders(kansas,
texas)
```

Parsing rules (combinators)

Forward composition

```
X/Y : f        Y/Z : g        ➡ X/Z : λx.f(g(x))

N/N :          N/N :          ➡ N/N :
λf.λx.f(x)       λf.λx.f(x)       λf.λx.f(x)
  ∧ big(x)       ∧ red(x)         ∧ big(x) ∧ red(x)
```

Reverse composition

```
Y\Z : g      X\Y : f      ➡ X\Z : λx.f(g(x))
```

Parsing rules (combinators)

Type-raising

`Y : x` ➡ `X/(X\Y) : λf.f(x)`

`NP : utah` ➡ `S/(S\NP) : λf.f(utah)`

CCG parsing

Texas	*borders*	*Kansas*
`NP : texas`	`(S\NP)/NP : λx.λy. borders(y, x)`	`NP : kansas`
	`S\NP : λy.borders(y, kansas)`	
	`S : borders(texas, kansas)`	

Parsing a question

What	*states*	*border*	*Texas*
`(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)`	`N : λx.state(x)`	`(S\NP)/NP : λx.λy.borders(y, x)`	`NP : texas`
`S/(S\NP) : λg.λx.state(x) ∧ g(x)`		`S\NP : λy.borders(y, texas)`	
`S : λx.state(x) ∧ borders(x, texas)`			

Probabilistic CCG (PCCG)

A log-linear model:

- A CCG for parsing
- Lexical features

  $f_i(L, S, T)$: number of times lexical item i is used in the parse T that maps from sentence S to logical form L
- A parameter vector θ with an entry for each f_i

PCCG distributions

Log-linear model defines a joint distribution:

$$P(L, T|S; \theta) = \frac{e^{f(L,T,S)\cdot\theta}}{\sum_{(L,T)} e^{f(L,T,S)\cdot\theta}}$$

Parses are a hidden variable:

$$P(L|S; \theta) = \sum_T P(L, T|S; \theta)$$

PCCG parsing algorithm

- Given sentence *S*, find most probable logical form *L*

$$\text{argmax}_L P(L|S;\theta) = \text{argmax}_L \sum_T P(L,T|S;\theta)$$

- Uses dynamic programming: “chart parsing”
  - Like the CKY algorithm for parsing in a PCFG (covered in CS224N!)
  - Chart contains one cell for every possible token span in sentence
  - Cells are populated bottom-up by combining smaller spans
- Uses a beam to track (& sum over) most likely trees
  - Inference is therefore approximate, but effective in practice

Learning

- Generating lexical items

  (Promiscuously generate any lexicon entries which might help parse.)

- Learning PCCG parameters / weights

  (Learn which lexical items are good by hill-climbing on training data.)

What makes learning hard?

Why is this a harder learning problem than, e.g., learning to parse the Penn Treebank?

- We have to learn the lexicon, not just rule weights
- Training data gives only logical forms, not parse trees

Lexical generation

Input: training example

Sentence	*Texas borders Kansas*
Logical form	`borders(texas, kansas)`

lexical generation (GENLEX)

Output: lexicon entries

Texas	:=	NP	:	`texas`
borders	:=	`(S\NP)/NP`	:	`λx.λy.borders(y, x)`
Kansas	:=	NP	:	`kansas`

GENLEX

Input: a training example (S_i, L_i)

Computation:

1. Create all token sequences (n-grams) in S_i
2. Create categories from L_i (using rules, below)
3. Create lexical entries from the cross product (i.e., the Cartesian product) of these two sets

Output: lexicon Λ

GENLEX step 1: generate n-grams

Input: sentence

Texas borders Kansas

Output: n-grams

Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas

GENLEX step 2: generate categories

Input: logical form

```
borders(Texas, Kansas)
```

Output: categories

. . .

. . .

. . .

. . .

. . .

. . .

Two GENLEX rules

Input trigger		Output category
a constant `c`	➡	`NP : c`
an arity-two predicate `p`	➡	`(S\NP)/NP : λx.λy.p(y, x)`

Example input: logical form

```
borders(Texas, Kansas)
```

Example output: categories

```
NP : texas
NP : kansas
(S\NP)/NP : λx.λy.borders(y, x)
```

All the GENLEX rules

Input trigger	Output category
a constant `c`	`NP : c`
arity-one predicate `p`	`N : λx.p(x)`
arity-one predicate `p`	`S\NP : λx.p(x)`
arity-two predicate `p`	`(S\NP)/NP : λx.λy.p(y, x)`
arity-two predicate `p`	`(S\NP)/NP : λx.λy.p(x, y)`
arity-one predicate `p`	`N/N : λg.λx.p(x) ∧ g(x)`
arity-two predicate `p` and constant `c`	`N/N : λg.λx.p(x, c) ∧ g(x)`
	`(N\N)/NP : λx.λg.λy.p(y, x) ∧ g(x)`
arity-two predicate `p`	`NP/N : λg.argmax|min(g(x), λx.f(x))`
arity-one function `f`	`S/NP : λx.f(x)`
arity-one function `f`	

All the GENLEX rules, with examples

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an arg max / min with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

GENLEX step 3: cross product

Input: training example

Sentence	*Texas borders Kansas*
Logical form	`borders(texas, kansas)`

Output: lexicon entries

Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas

×

```
NP : texas
NP : kansas
(S\NP)/NP :
  λx.λy.borders(y,
x)
```

GENLEX: output lexicon

Texas	:=	NP	:	texas
Texas	:=	NP	:	kansas
Texas	:=	(S\NP)/NP	:	λx.λy.borders(y, x)
borders	:=	NP	:	texas
borders	:=	NP	:	kansas
borders	:=	(S\NP)/NP	:	λx.λy.borders(y, x)
...		...		...
Texas borders Kansas	:=	NP	:	texas
Texas borders Kansas	:=	NP	:	kansas
Texas borders Kansas	:=	(S\NP)/NP	:	λx.λy.borders(y, x)

The initial lexicon

The initial lexicon Λ_0 has two types of entries:

- Domain independent

  what := `(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)`

- Domain dependent

  Texas := `NP : texas`

A simple learning algorithm

Inputs	Initial lexicon Λ_0 Training examples $E = \{(S_i, L_i) : i = 1...n\}$
Initialization	Create lexicon $\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} \mathrm{GENLEX}(S_i, L_i)$ Create initial parameters θ^0
Computation	Estimate parameters $\theta = \mathrm{SGD}(E, \theta^0, \Lambda^*)$
Output	Return $\mathrm{PCCG}(\Lambda^*, \theta)$

The final learning algorithm

Inputs Λ_0, $E = \{(S_i, L_i) : i = 1...n\}$

Initialization Create θ^0

Computation For $t = 1...T$

- Prune lexicon:
  - For each $(S_i, L_i) \in E$:
    - Set $\lambda = \Lambda_0 \cup \text{GENLEX}(S_i, L_i)$
    - Calculate $\pi = \text{PARSE}(S_i, L_i, \lambda, \theta^{t-1})$
    - Define λ_i to be all lexical items in π
  - Set $\Lambda^t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$
- Estimate parameters: $\theta^t = \text{SGD}(E, \theta^{t-1}, \Lambda^t)$

Output Return $\text{PCCG}(\Lambda^T, \theta^T)$

Experiments

Two database domains:

- Geo880
  - 600 training examples
  - 280 test examples
- Jobs640
  - 500 training examples
  - 140 test examples

Examples from Geo880

what cities in texas have the highest number of citizens?
what is the area of the state with the smallest population density?
what state is des moines located in?
what are the major cities in states through which the mississippi runs?
what are the major cities in the smallest state in the us?
what is the capital of ohio?
what is the population of denver?
what is the biggest city in nebraska?
what are the major cities in new mexico?
what is the capital of california?
what is the capital of utah?
what states border states that border states that border states that border texas?
what are the population of mississippi?
where is mount whitney?
what is the population of the state with the largest area?
what is the capital of iowa?
what is the most populous state through which the mississippi runs?
how many states border on the state whose capital is boston?
which states does the longest river cross?
what is the capital of new york?
what is the smallest city in arkansas?
how many people live in mississippi?
which state has the highest peak in the country?
what river runs through illinois?
how many people live in the capital of georgia?
what is largest capital?
how many states are in the usa?
how many big cities are in pennsylvania?
what state contains the highest point in the us?
where is san jose?
how many cities are in montana?
what states border michigan?
name the rivers in arkansas.
what rivers are in nevada?
could you tell me what is the highest point in the state of oregon?
what state which the mississippi runs through has the largest population?
what state borders new york?
which states border hawaii?
what is the population of atlanta ga?
which state is the smallest?
what is the largest city in missouri?
how much population does texas have?
give me the number of rivers in california?
how many states does iowa border?
what states border states that the ohio runs through?
which states border texas?
what is the population of dallas?
what is the state with the lowest point?
what is the capital of maryland?
what is the longest river in the states that border nebraska?

Evaluation

Test for completely correct semantics

- Precision:
  # correct / total # parsed
- Recall:
  # correct / total # sentences

Results

	Geo880		Jobs640	
	precision	recall	precision	recall
Z&C05	96.25	79.29	97.36	79.29
COCKTAIL	89.92	79.40	93.25	79.84

Examples of learned lexical entries

states	N	:	λx.state(x)
major	N/N	:	λg.λx.major(x) ∧ g(x)
population	N	:	λx.population(x)
cities	N	:	λx.city(x)
river	N	:	λx.river(x)
run through	(S\NP)/NP	:	λx.λy.traverse(y, x)
the largest	NP/N	:	λg.argmax(g, λx.size(x))
rivers	N	:	λx.river(x)
the highest	NP/N	:	λg.argmax(g, λx.elev(x))
the longest	NP/N	:	λg.argmax(g, λx.len(x))

Error analysis

Low recall: GENLEX is not general enough

- Fails to parse 10% of training examples

Some unparsed examples include:

- *Through which states does the Mississippi run?*
- *If I moved to California and learned SQL on Oracle could I find anything for 3000 on Unix?*

Zettlemoyer & Collins 2007

- Shifted from Geo880 to ATIS (~5,000 examples)

  on may four atlanta to denver delta flight 257

  show me information on delta from fort worth to philadelphia

  april 22nd dallas to miami the latest nighttime departure one way

- Non-standard CCG combinators with learned costs
  - e.g., relaxed function application, to allow flexible word order
  - e.g., role-hypothesizing type-shifting, to insert missing words

- New online learning algorithm: structured perceptron

Zettlemoyer & Collins 2009

- Tackles (discourse-)context-dependent interpretation
- Uses interaction sequences from ATIS dataset

  show me the flights from boston to philly
  show me the ones that leave in the morning
  what kind of plane is used on these flights

- Extends CCG, λ-calculus with referential expressions
  - E.g., *ones* := N : λx.!⟨e, t⟩(x), semantics to be recovered from context
- Derivations (features, scoring) now include context

Kwiatkowski et al. 2010

- Luke's first paper as a UW prof
- Introduces UBL: unification-based learning
- Initial lexicon pairs whole sentences with logical forms
- UBL iteratively splits lexical items into smaller parts
- Experiments in four languages, two different MRs

And the hits just keep on coming ...

Artzi & Zettlemoyer 2011, Bootstrapping Semantic Parsers from Conversations

Kwiatkowski et al. 2011, Lexical Generalization in CCG Grammar Induction for Semantic Parsing

Matuszek et al. 2012, Learning to Parse Natural Language Commands to a Robot Control System

Artzi & Zettlemoyer 2013, Weakly supervised learning of semantic parsers for mapping instructions to actions

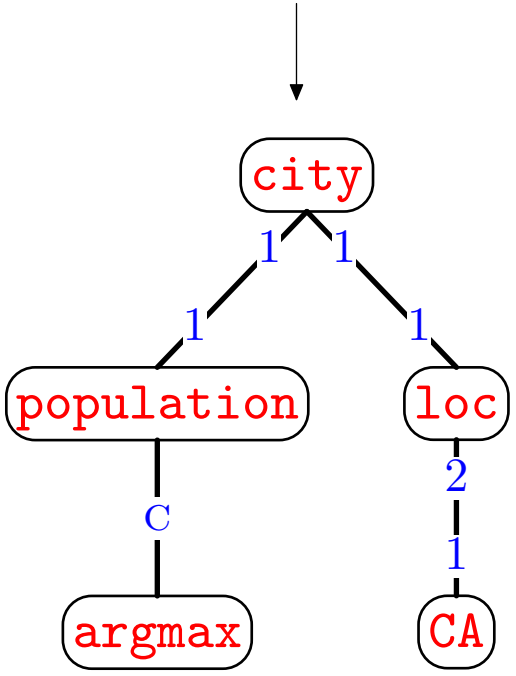
Kwiatkowski et al. 2013, Scaling Semantic Parsers with On-the-fly Ontology Matching

Dependency-Based Compositional Semantics (DCS)

What is the most populous city in California?

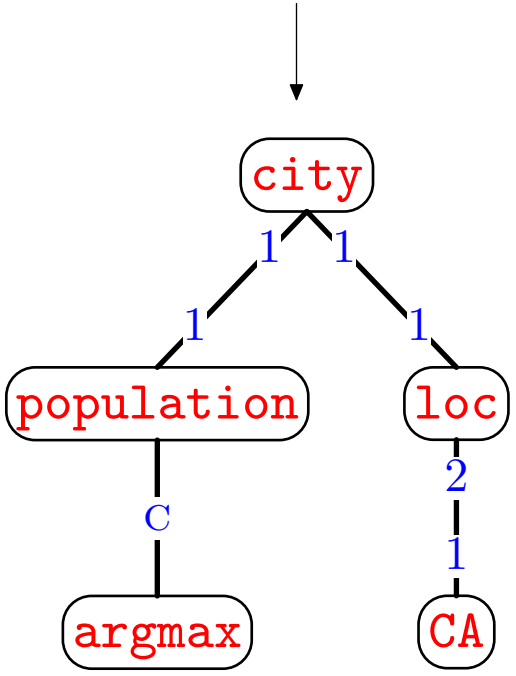
Dependency-Based Compositional Semantics (DCS)

What is the most populous city in California?



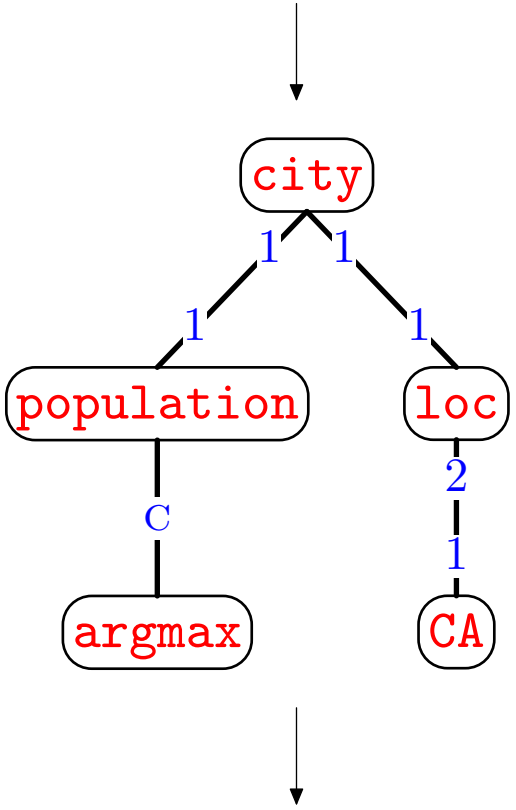
How to interpret the logical form?

What is the most populous city in California?



How to interpret the logical form?

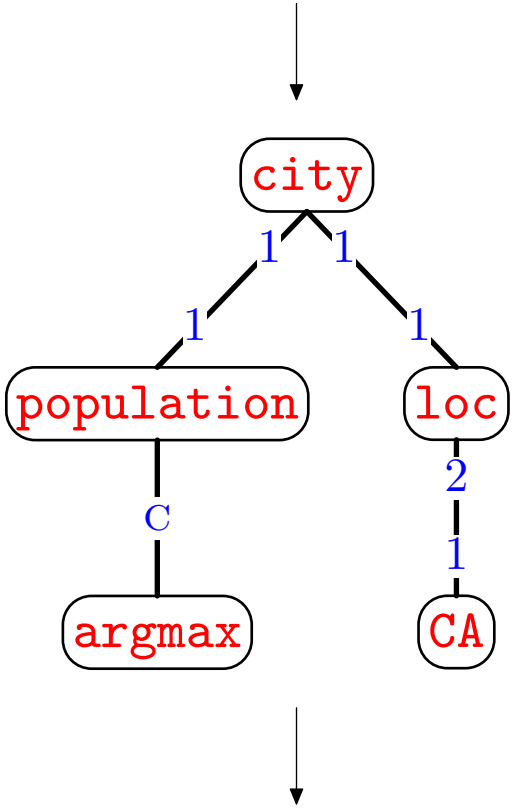
What is the most populous city in California?



Los Angeles

How to interpret the logical form?

What is the most populous city in California?



World/Database → *Los Angeles*

World/Database

`city`

San Francisco
Chicago
Boston
...

`state`

Alabama
Alaska
Arizona
...

`loc`

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

...

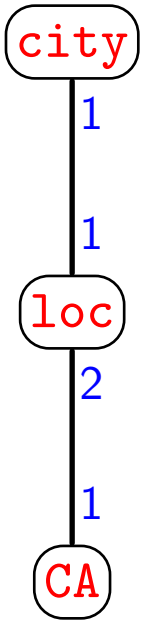
`border`

Washington	Oregon
Washington	Idaho
Oregon	Washington
...	...

...

Basic DCS Trees

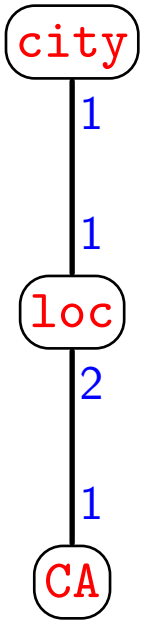
DCS tree



Database

Basic DCS Trees

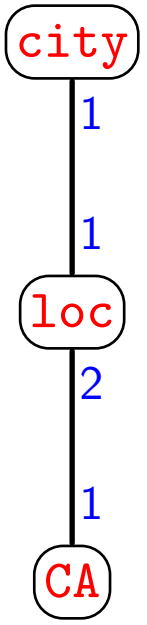
DCS tree **Constraints** **Database**



A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in$ `city`

Database

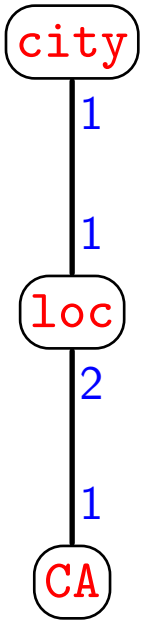
`city`

San Francisco
Chicago
Boston
...

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in$ city

$\ell \in$ loc

Database

city

San Francisco
Chicago
Boston
...

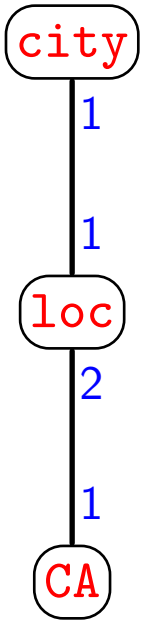
loc

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in$ city

$\ell \in$ loc

$s \in$ CA

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

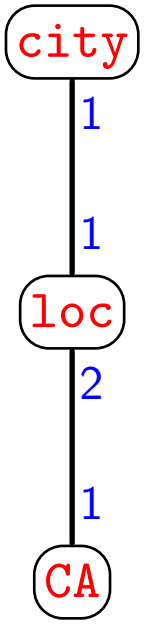
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

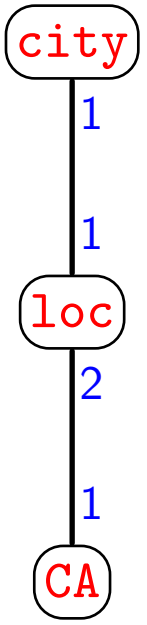
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$\ell_2 = s_1$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

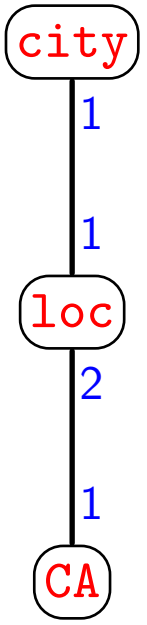
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$\ell_2 = s_1$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

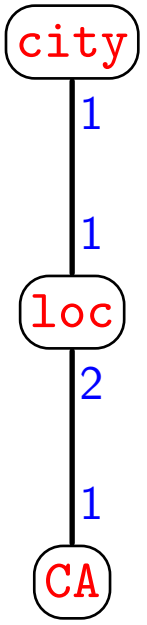
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$\ell_2 = s_1$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	**California**
Boston	Massachusetts
...	...

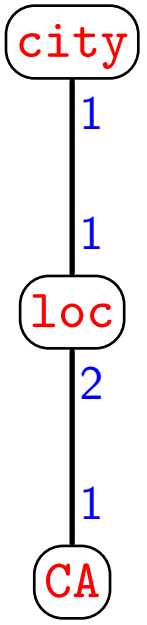
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$\ell_2 = s_1$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	**California**
Boston	Massachusetts
...	...

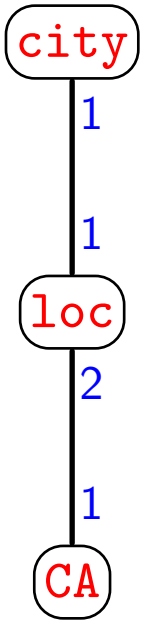
CA

California

A DCS tree encodes a **constraint satisfaction problem** (CSP)

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$\ell_2 = s_1$

$s \in \texttt{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

Mount Shasta	California
San Francisco	**California**
Boston	Massachusetts
...	...

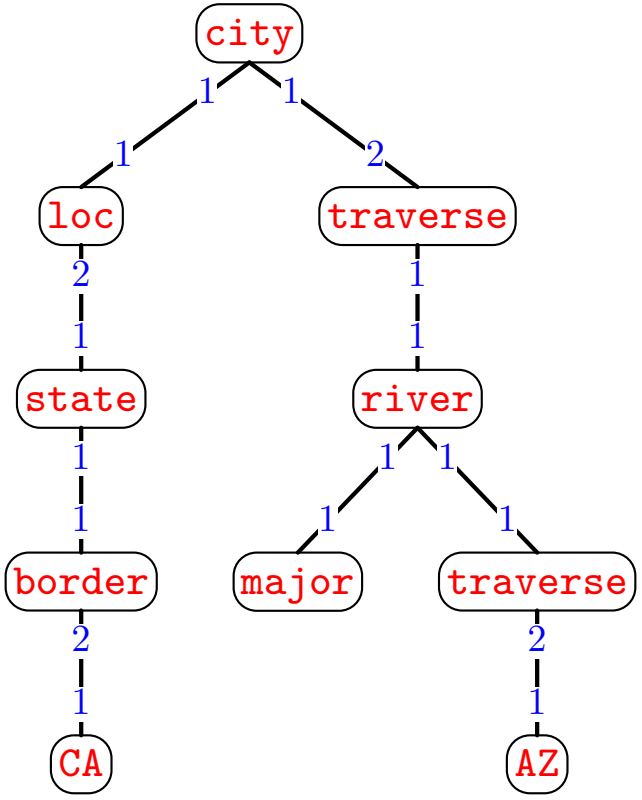
CA

California

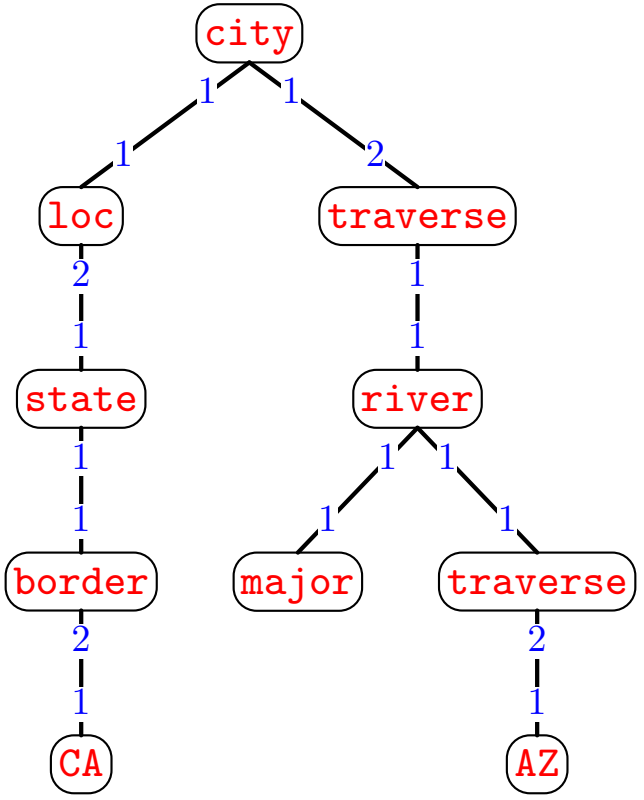
A DCS tree encodes a **constraint satisfaction problem** (CSP)

Computation: dynamic programming $\Rightarrow$ time $= O(\#$ nodes$)$

Properties of DCS Trees

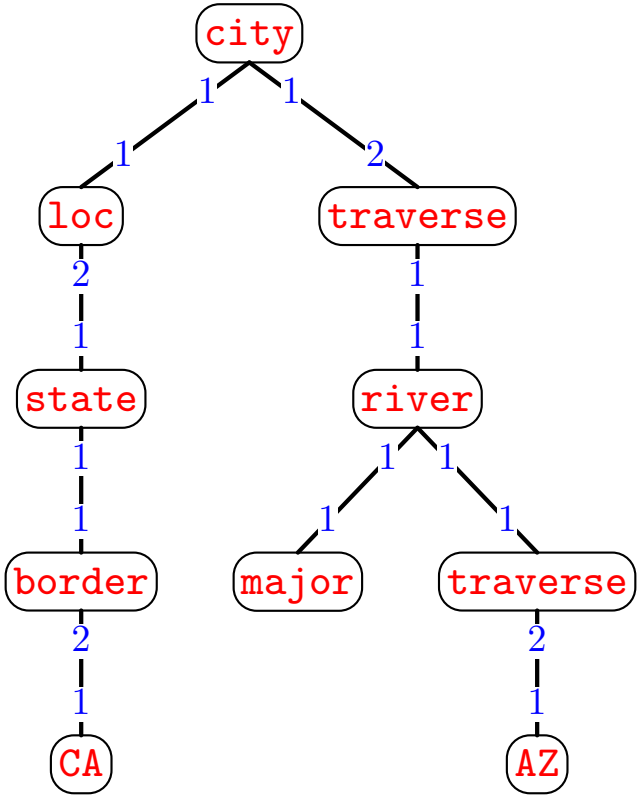


Properties of DCS Trees



Trees

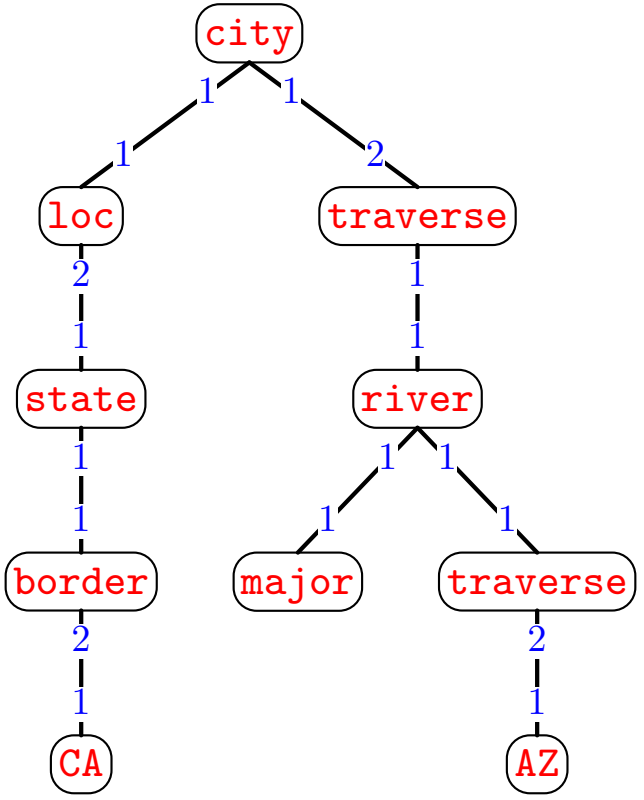
Properties of DCS Trees



Linguistics ..... Trees

syntactic locality

Properties of DCS Trees



Linguistics ..... Trees ..... Computation

syntactic locality

efficient interpretation

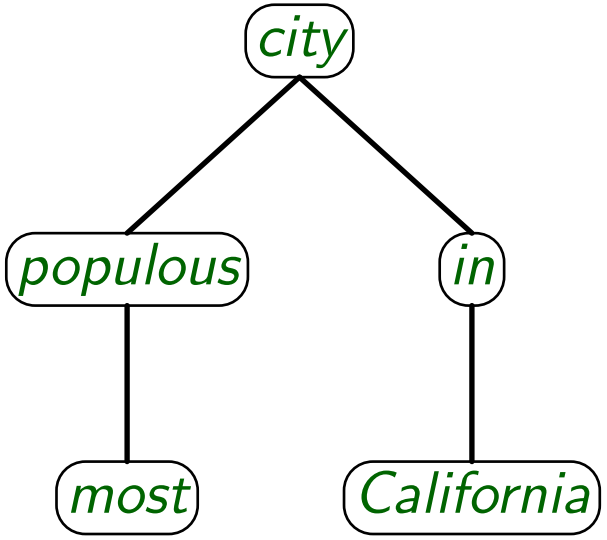
Divergence between Syntactic and Semantic Scope

most populous city in California

Divergence between Syntactic and Semantic Scope

most populous city in California

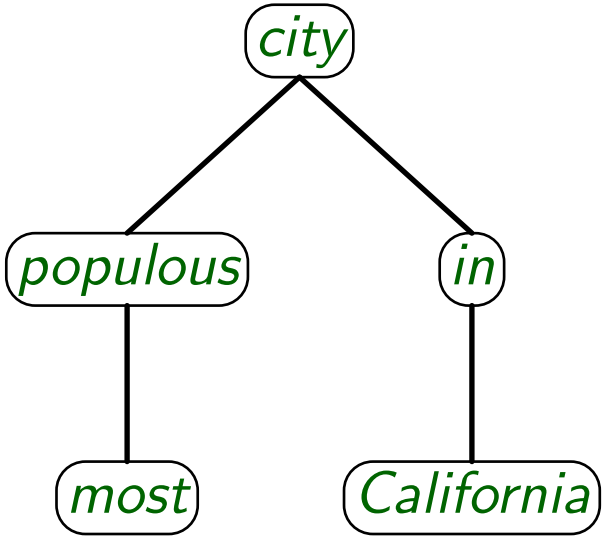
Syntax



Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



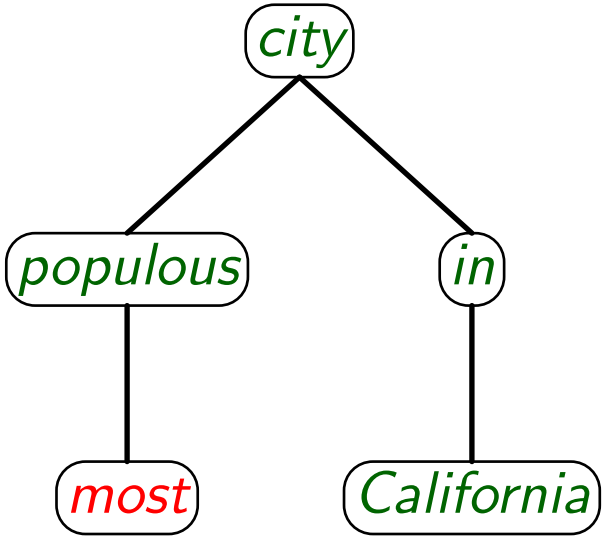
Semantics

$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



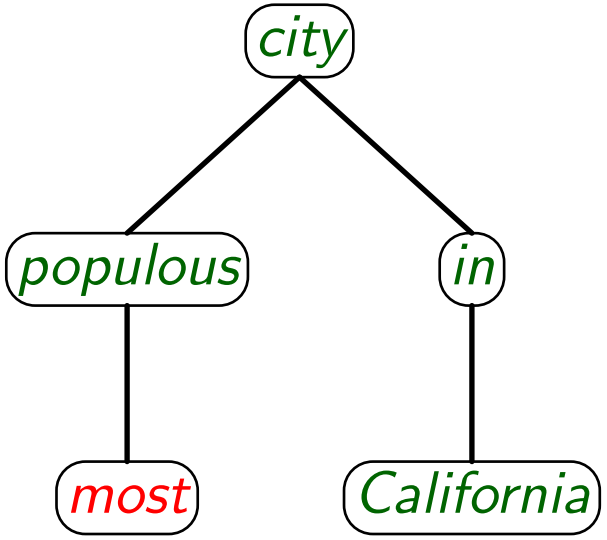
Semantics

$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



Semantics

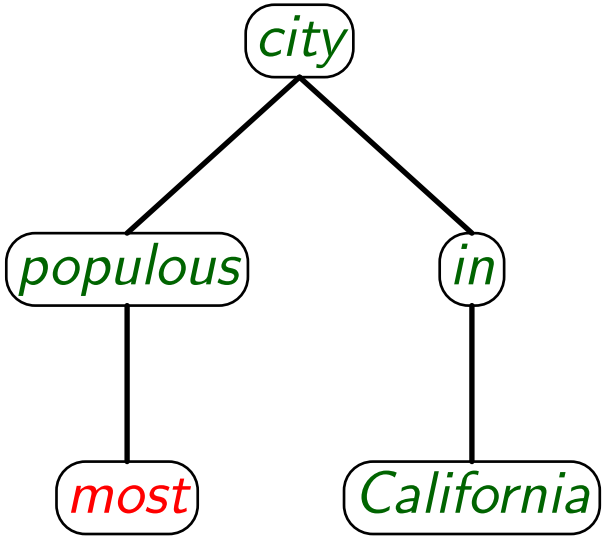
$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Problem: syntactic scope is lower than semantic scope

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



Semantics

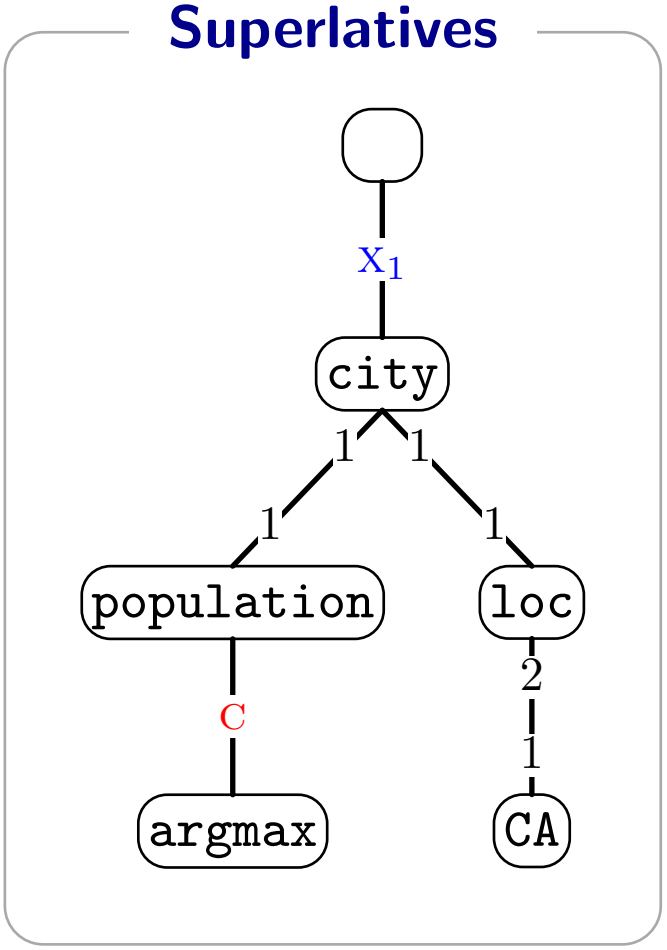
$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Problem: syntactic scope is lower than semantic scope

If DCS trees look like syntax, how do we get correct semantics?

Solution: Mark-Execute

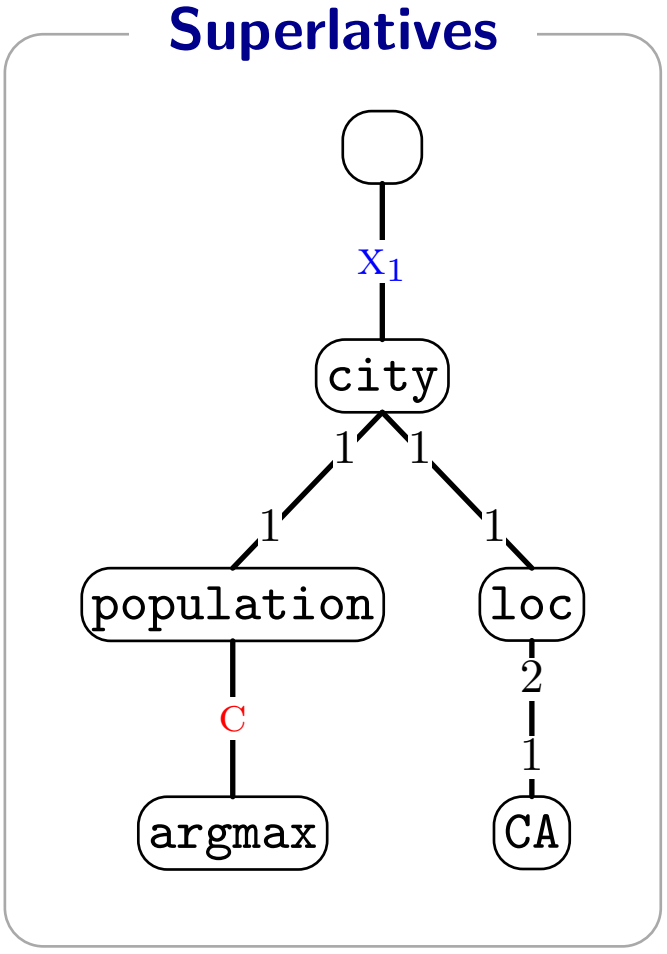
most populous city in California



Solution: Mark-Execute

most populous city in California

Mark at syntactic scope

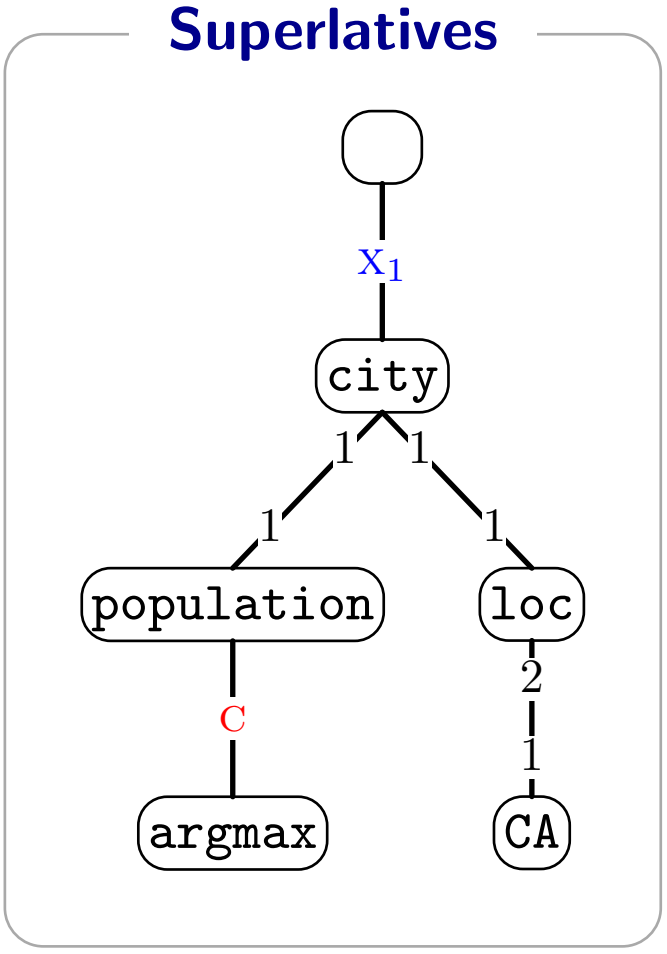


Solution: Mark-Execute

most populous city in California

Execute at semantic scope

Mark at syntactic scope

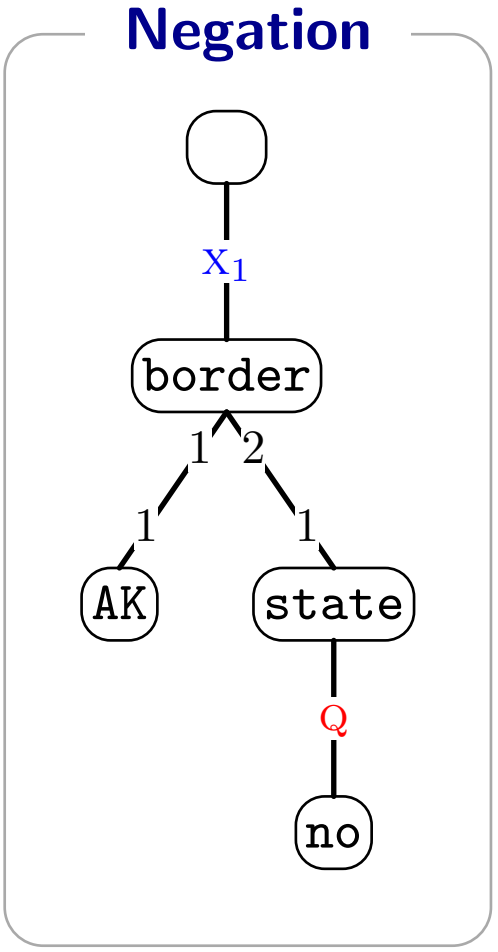


Solution: Mark-Execute

Alaska borders no states.

Execute at semantic scope

Mark at syntactic scope



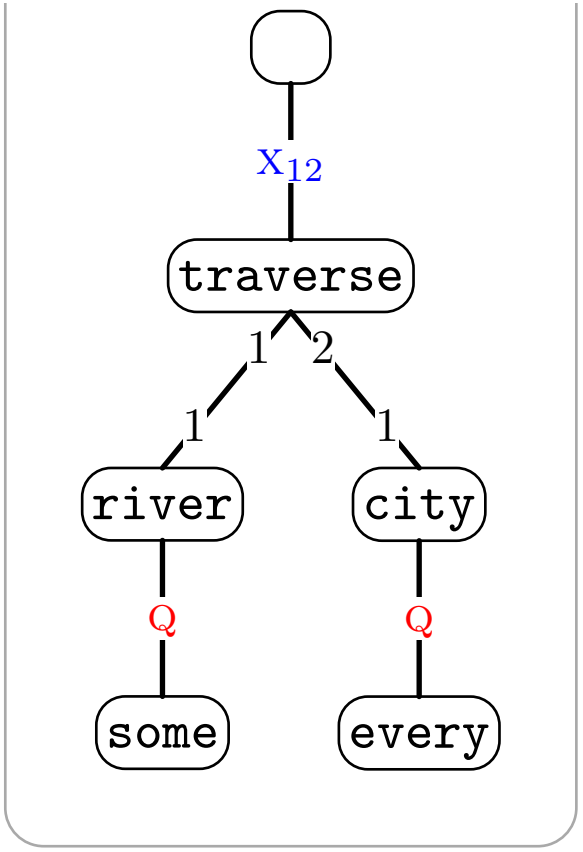
Solution: Mark-Execute

Some river traverses every city.

Execute at semantic scope

Mark at syntactic scope

Quantification (narrow)



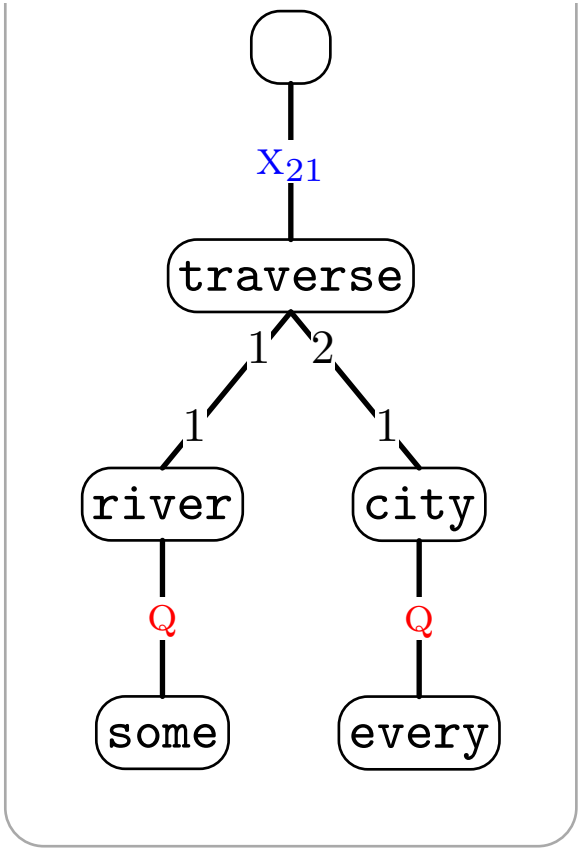
Solution: Mark-Execute

Some river traverses every city.

Execute at semantic scope

Mark at syntactic scope

Quantification (wide)



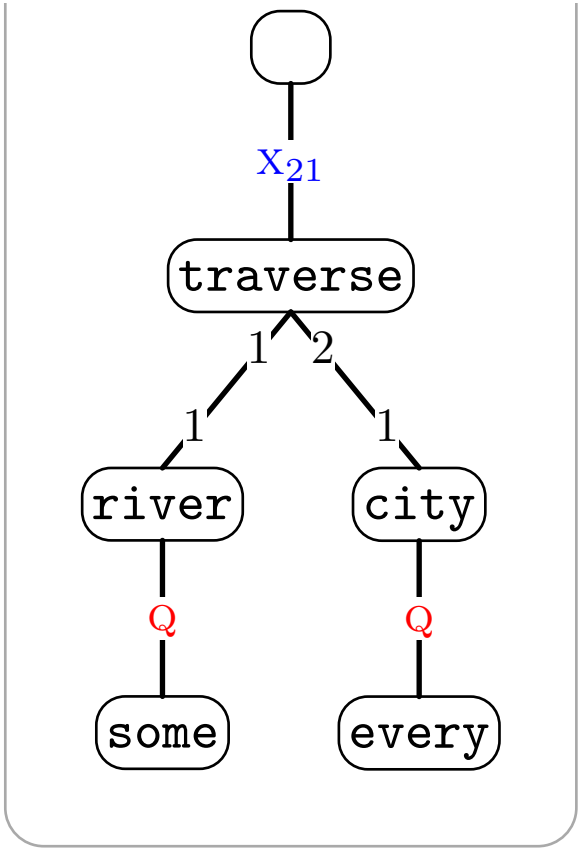
Solution: Mark-Execute

Some river traverses every city.

Execute at semantic scope

Mark at syntactic scope

Quantification (wide)



Analogy: Montague's quantifying in, Carpenter's scoping constructor

From Sentences to DCS Trees

Lexicon (very simple/crude)

no	$\Rightarrow$	`no`
state	$\Rightarrow$	`state`

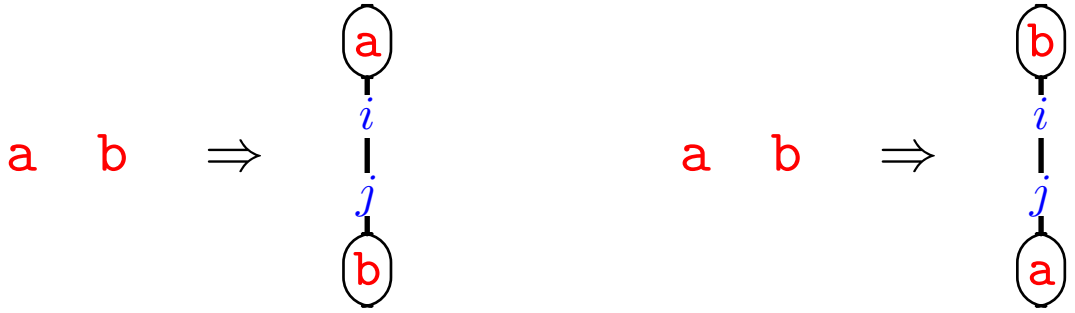
From Sentences to DCS Trees

Lexicon (very simple/crude)

no $\Rightarrow$ `no`

state $\Rightarrow$ `state`

Grammar (very simple/crude)



From Sentences to DCS Trees

Lexicon (very simple/crude)

no $\Rightarrow$ `no`

state $\Rightarrow$ `state`

Grammar (very simple/crude)

`a` `b` $\Rightarrow$

`a` `b` $\Rightarrow$

Words to Predicates (Lexical Semantics)

What *is* *the* *most* *populous* *city* *in* *CA* *?*

Words to Predicates (Lexical Semantics)

							CA	
What	*is*	*the*	*most*	*populous*	*city*	*in*	*CA*	*?*

Lexical Triggers:

1. String match *CA* $\Rightarrow$ `CA`

Words to Predicates (Lexical Semantics)

			`argmax`				`CA`	
What	*is*	*the*	*most*	*populous*	*city*	*in*	*CA*	*?*

Lexical Triggers:

1. String match *CA* $\Rightarrow$ `CA`
2. Function words (20 words) *most* $\Rightarrow$ `argmax`

Words to Predicates (Lexical Semantics)

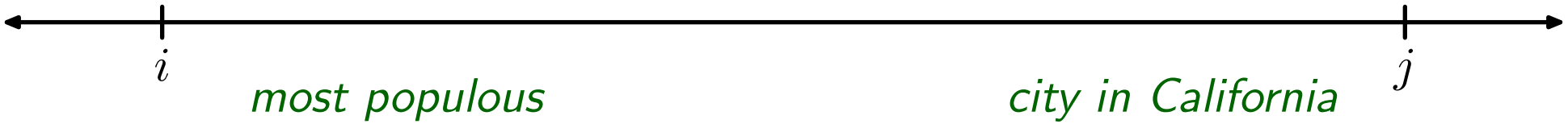
			`city`	`city`		
			`state`	`state`		
			`river`	`river`		
		`argmax`	`population`	`population`		`CA`	
What	*is*	*the*	*most*	*populous*	*city*	*in*	*CA*	*?*

Lexical Triggers:

1. String match *CA* $\Rightarrow$ `CA`
2. Function words (20 words) *most* $\Rightarrow$ `argmax`
3. Nouns/adjectives *city* $\Rightarrow$ `city state river population`

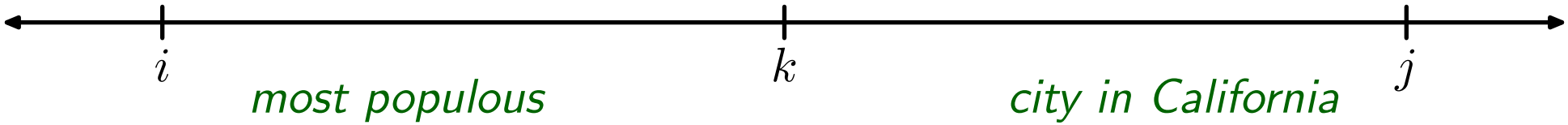
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



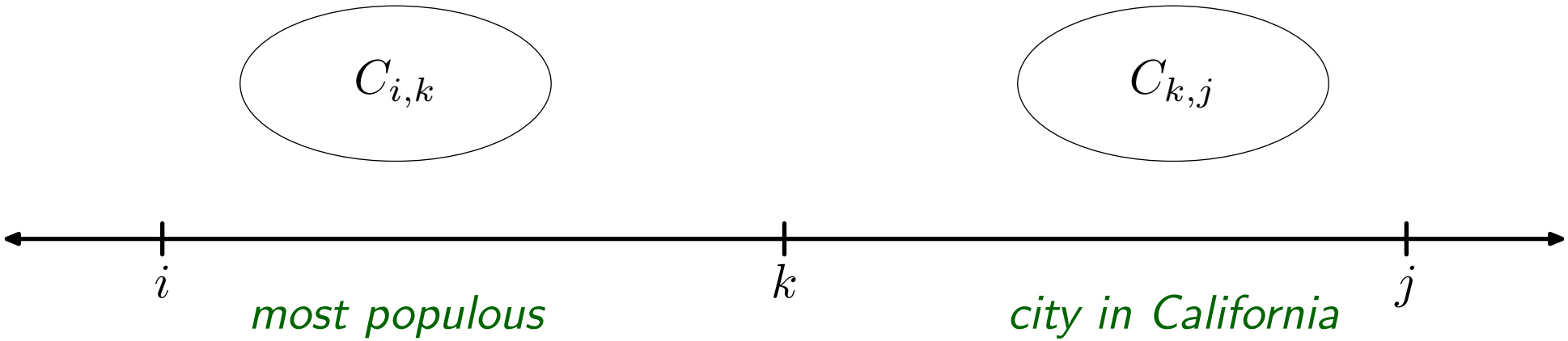
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



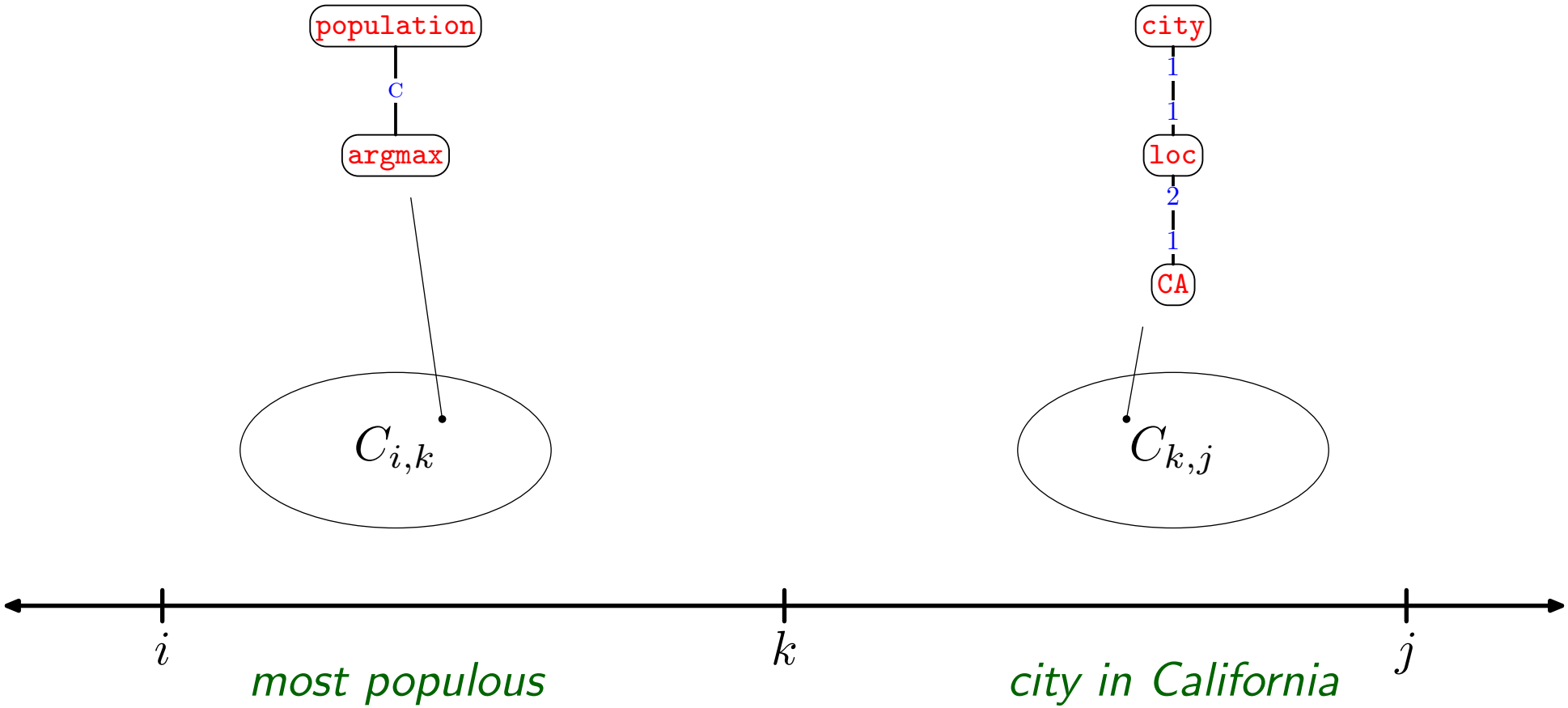
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



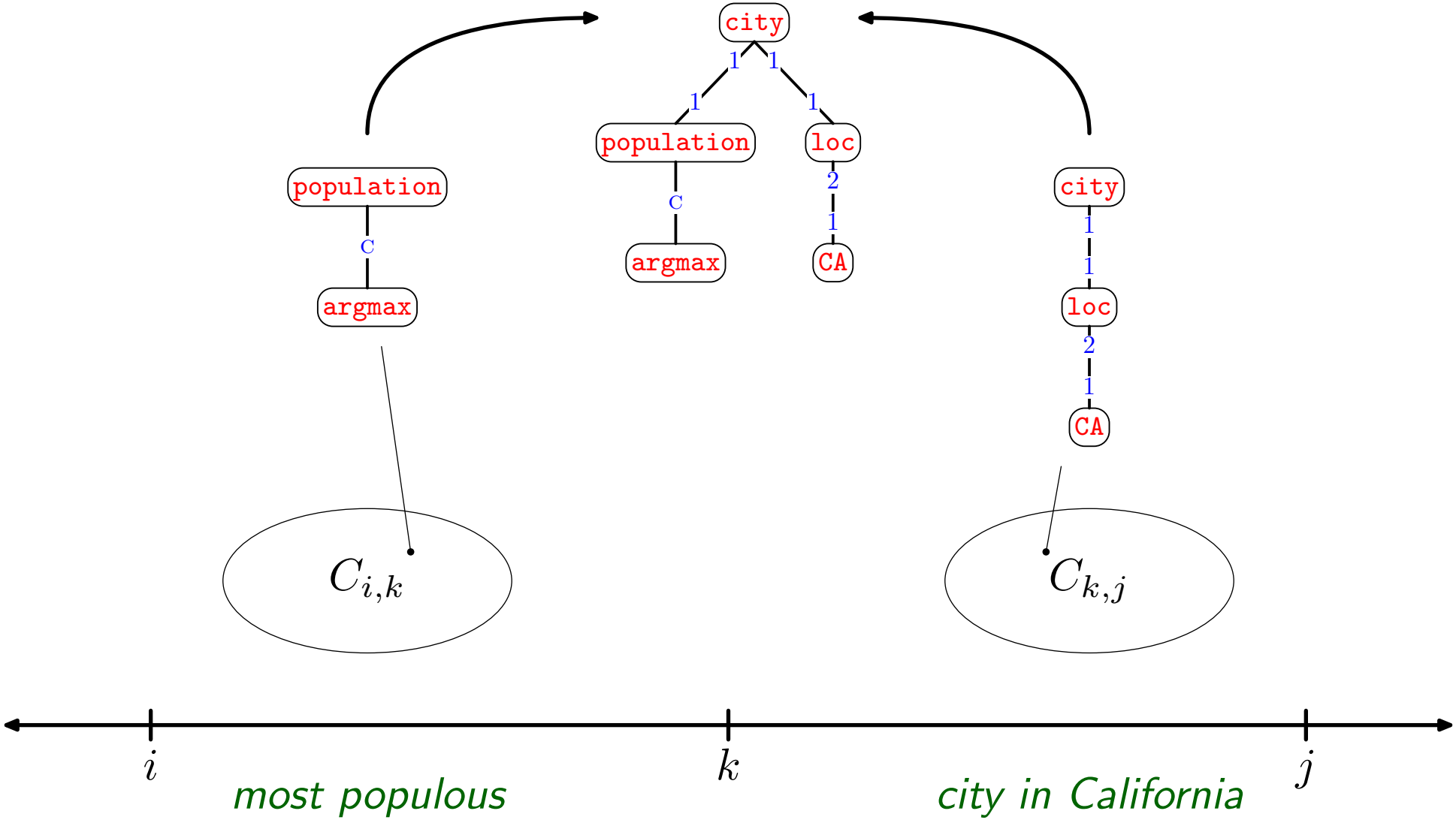
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



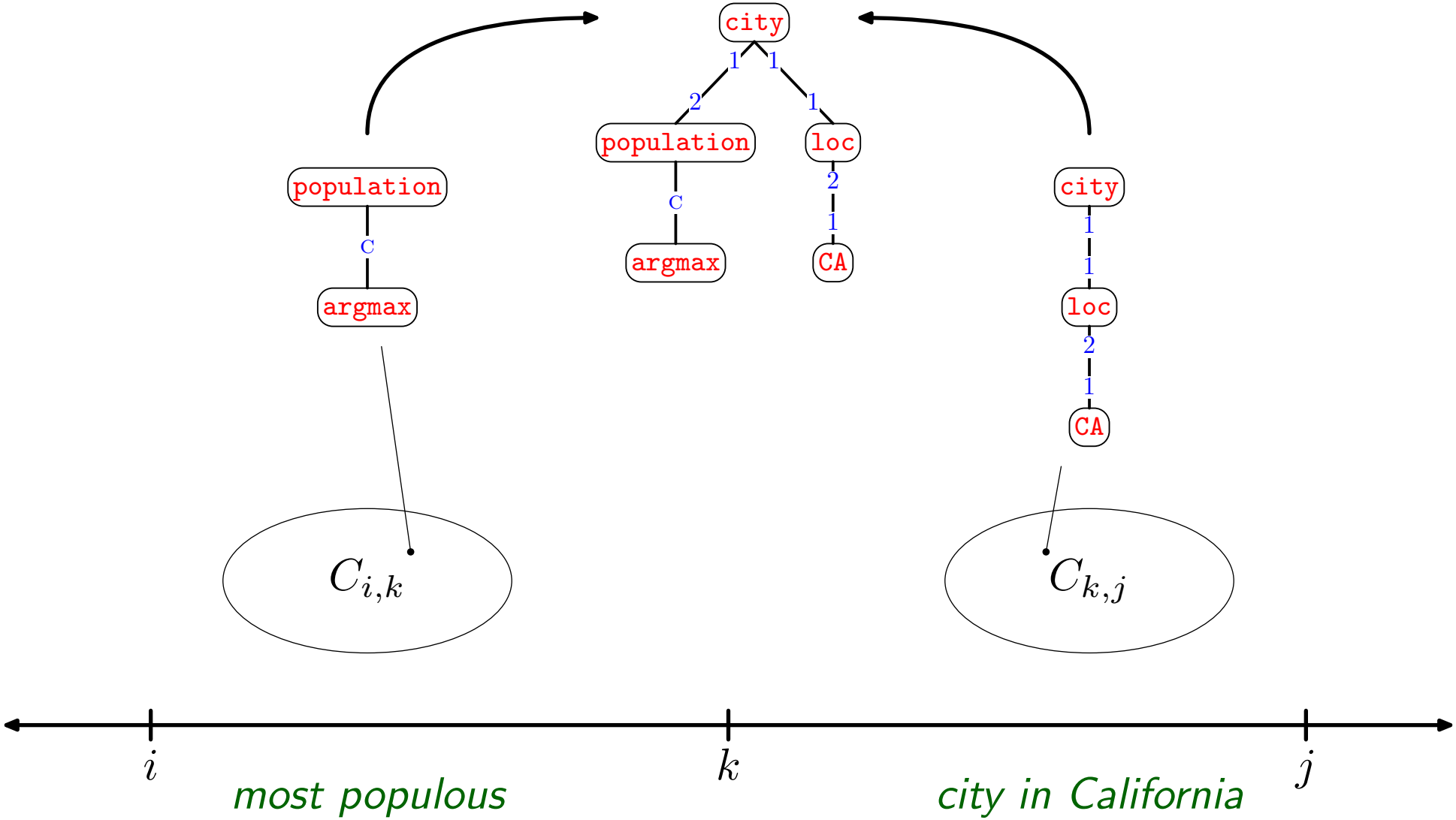
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



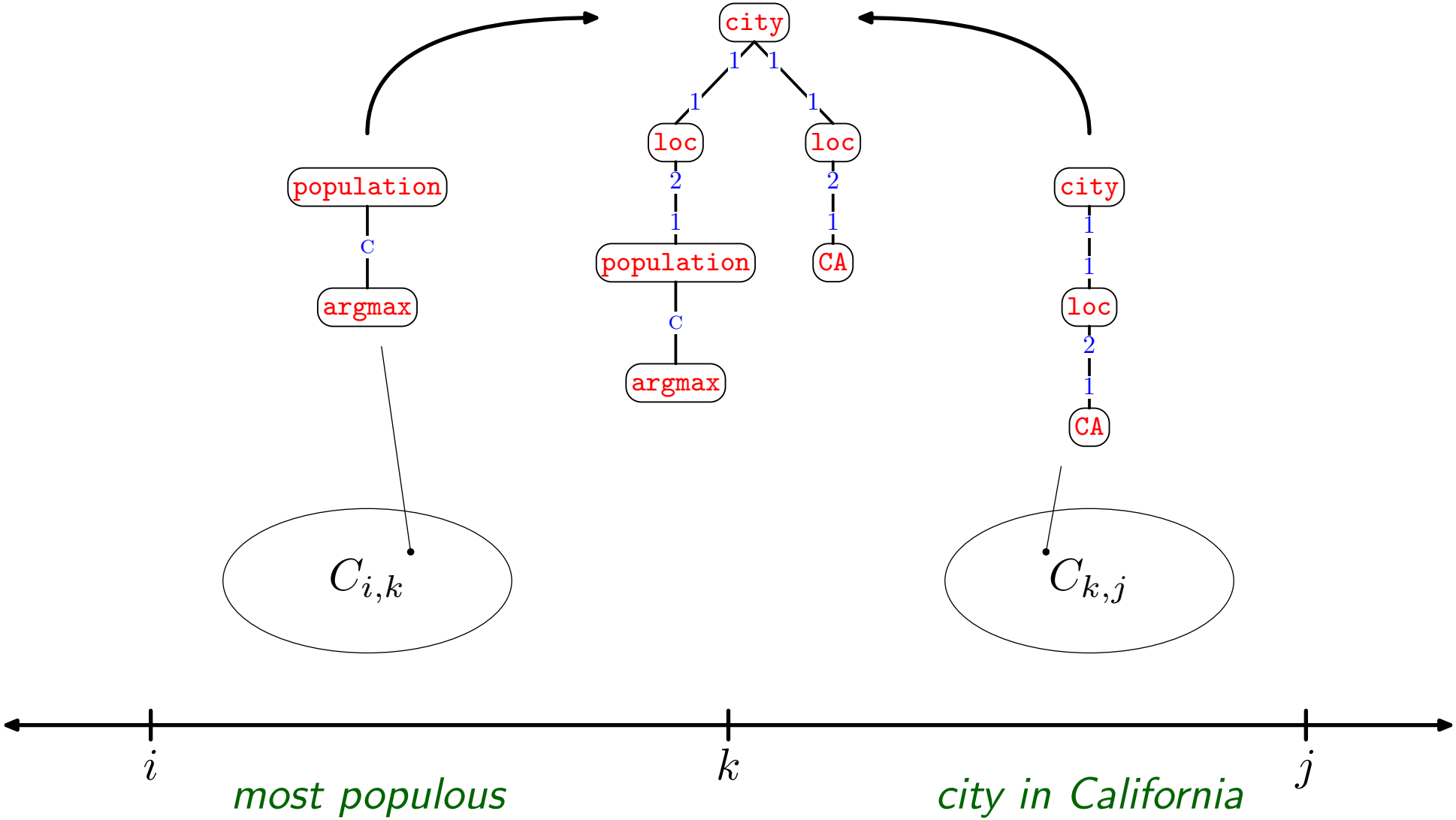
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



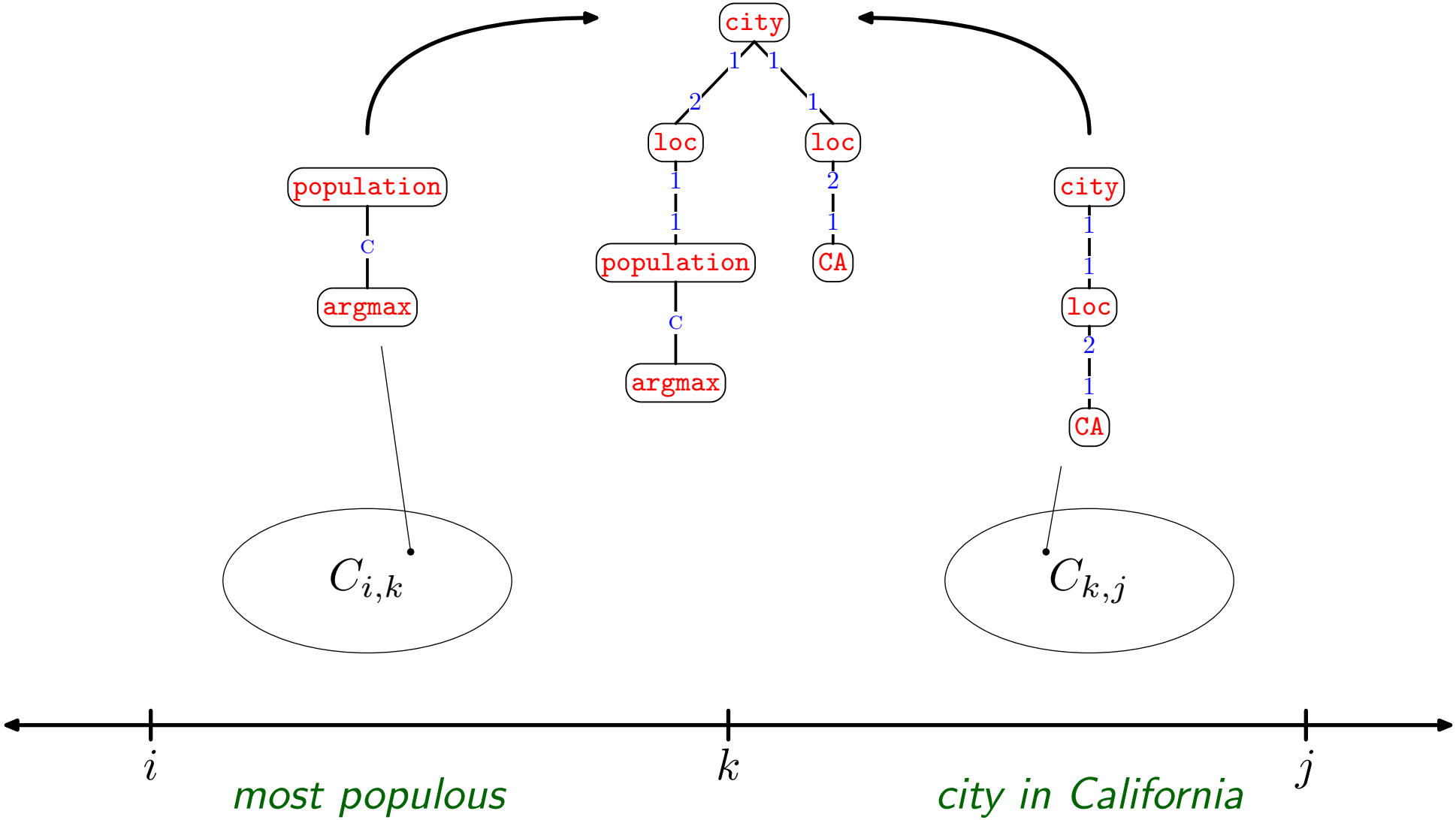
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



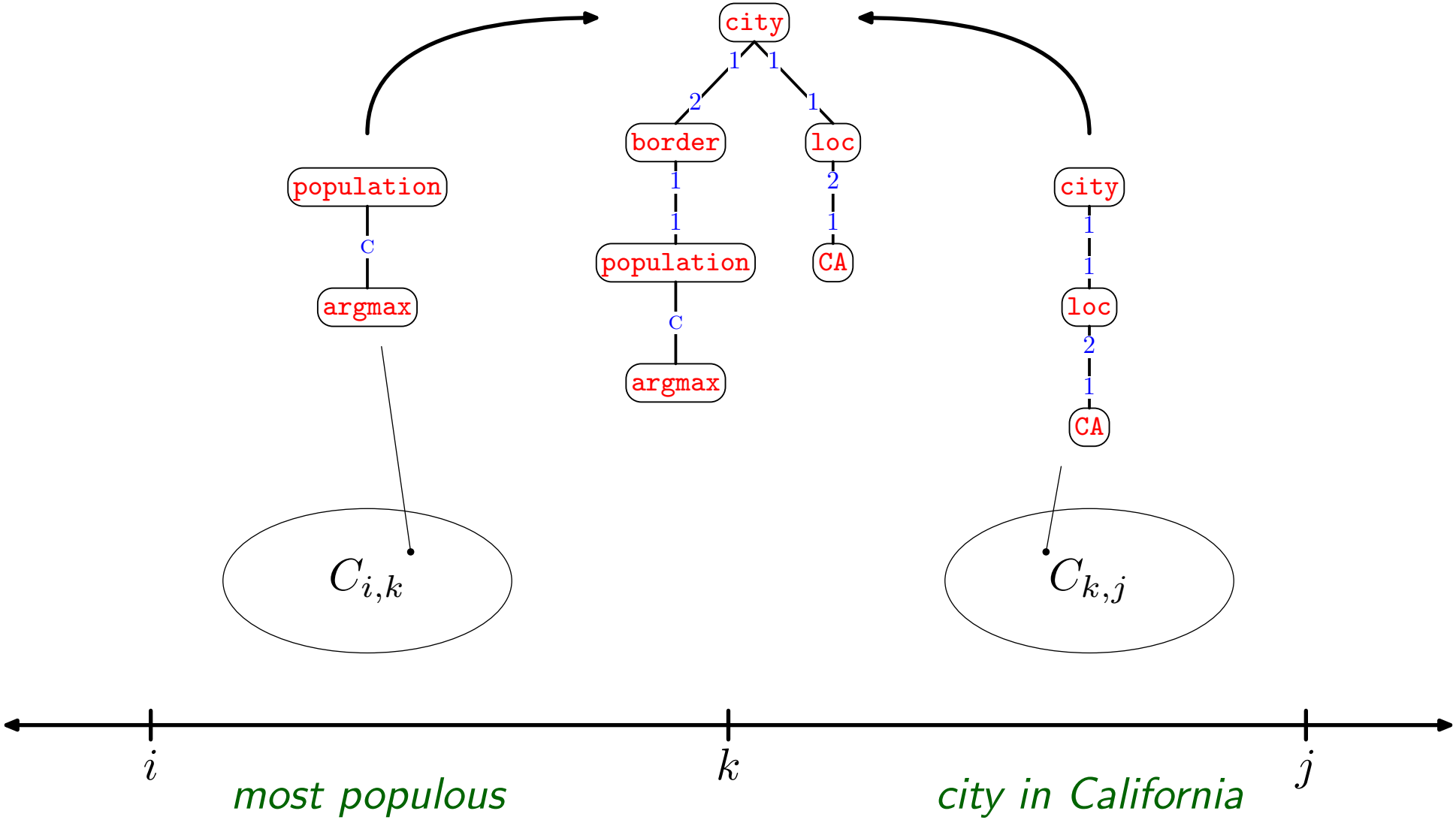
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



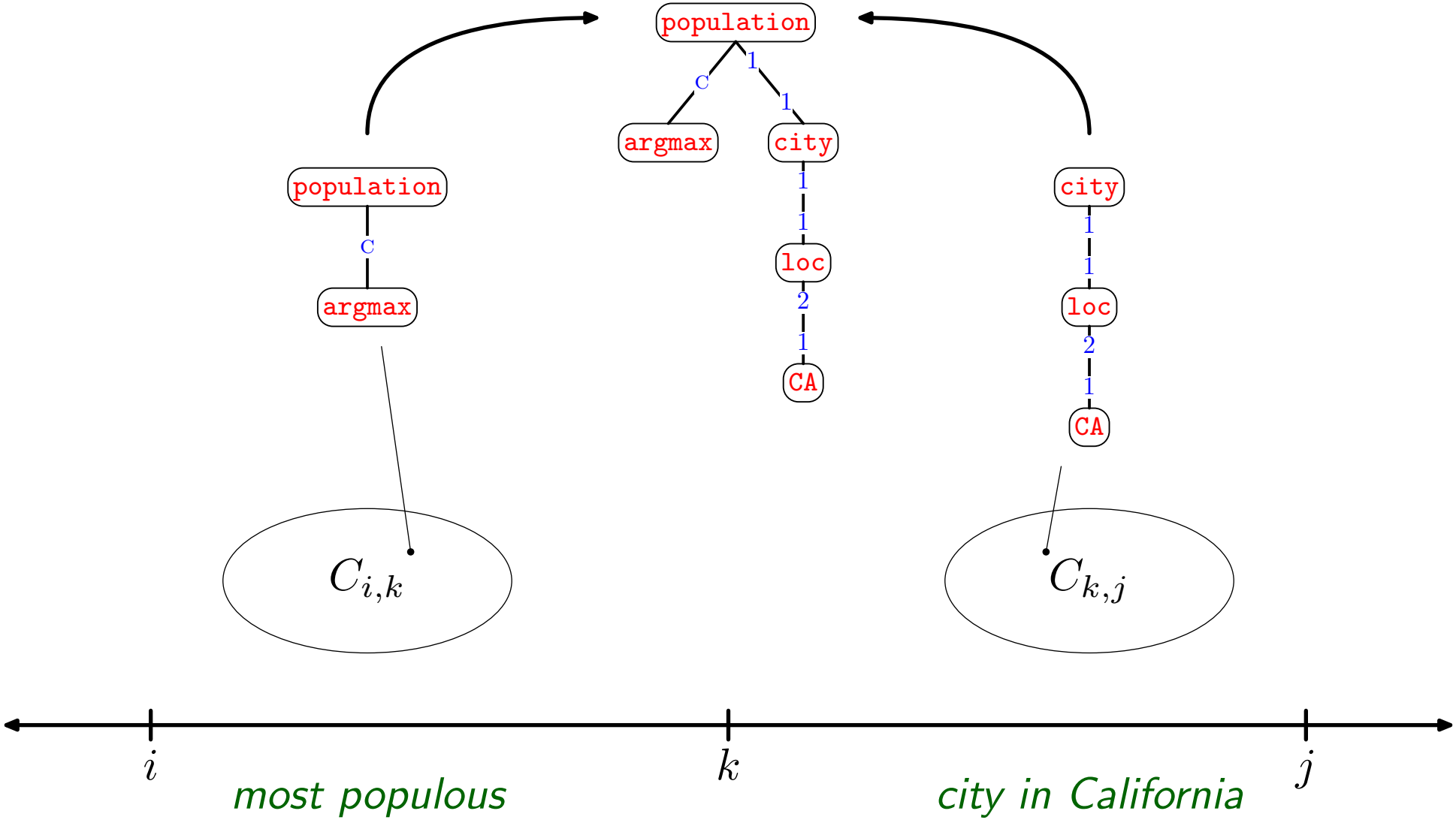
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$



Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = set of DCS trees for span $[i, j]$

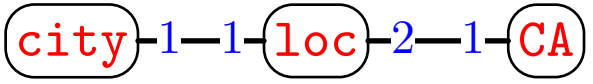


Comparison

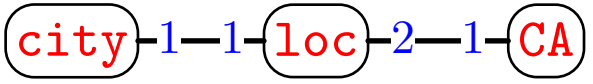
CCG

DCS

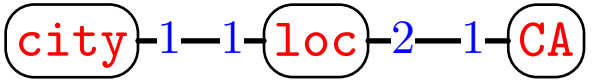
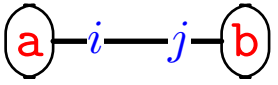
Comparison

	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	

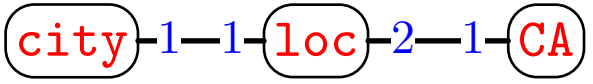
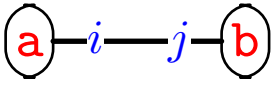
Comparison

	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	
Lexicon	categories + lambda calculus	predicates
major	$\text{N}/\text{N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	major

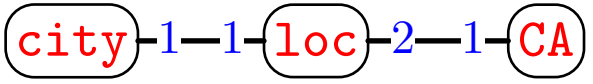
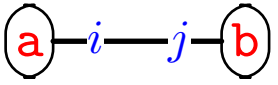
Comparison

	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	
Lexicon	categories + lambda calculus	predicates
major	$\text{N}/\text{N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	major
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \texttt{a} \quad X : \texttt{b} \quad \Rightarrow \quad Y : \texttt{a}(\texttt{b})$	

Comparison

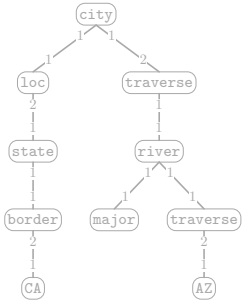
	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	
Lexicon	categories + lambda calculus	predicates
major	$\text{N}/\text{N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	major
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \texttt{a} \quad X : \texttt{b} \quad \Rightarrow \quad Y : \texttt{a}(\texttt{b})$	
Nature	tighter control	simple/permissive

Comparison

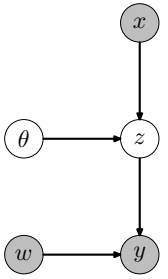
	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	
Lexicon	categories + lambda calculus	predicates
major	$\text{N}/\text{N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	major
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \texttt{a} \quad X : \texttt{b} \quad \Rightarrow \quad Y : \texttt{a}(\texttt{b})$	
Nature	tighter control	simple/permissive
Origin	linguistics	NLP

Outline

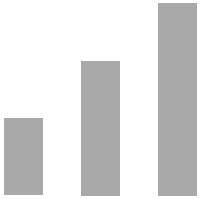
Representation



Learning



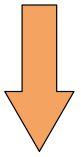
Experiments



Supervision

Detailed Supervision

What is the largest city in California?

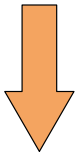


$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Supervision

Detailed Supervision

What is the largest city in California?



expert

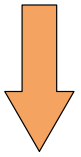
$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Supervision

Detailed Supervision

- doesn't scale up

What is the largest city in California?



expert

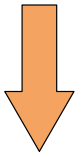
$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Supervision

Detailed Supervision

- doesn't scale up

What is the largest city in California?

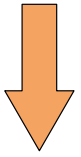


expert

$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Natural Supervision

What is the largest city in California?



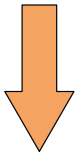
Los Angeles

Supervision

Detailed Supervision

- doesn't scale up

What is the largest city in California?

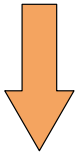


expert

$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Natural Supervision

What is the largest city in California?



non-expert

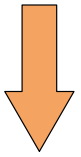
Los Angeles

Supervision

Detailed Supervision

- doesn't scale up

What is the largest city in California?



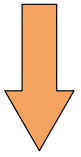
expert

$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Natural Supervision

- scales up

What is the largest city in California?



non-expert

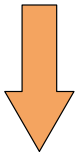
Los Angeles

Supervision

Detailed Supervision

- doesn't scale up
- representation-dependent

What is the largest city in California?



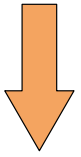
expert

$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Natural Supervision

- scales up

What is the largest city in California?



non-expert

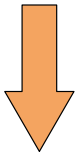
Los Angeles

Supervision

Detailed Supervision

- doesn't scale up
- representation-dependent

What is the largest city in California?



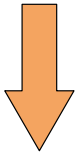
expert

$$\texttt{argmax}(\{c : \texttt{city}(c) \wedge \texttt{loc}(c, \texttt{CA})\}, \texttt{population})$$

Natural Supervision

- scales up
- representation-independent

What is the largest city in California?



non-expert

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\lambda x.\texttt{state}(x)$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\lambda x.\texttt{city}(x)$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\lambda x.\texttt{state}(x) \wedge \texttt{border}(x, \texttt{CA})$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

`population(CA)`

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\cdots$ LF LF LF LF LF [LF] LF LF LF LF LF LF LF LF LF LF [LF] $\cdots$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\cdots$ LF LF LF LF LF **LF** LF LF LF LF LF LF LF LF LF LF **LF** $\cdots$

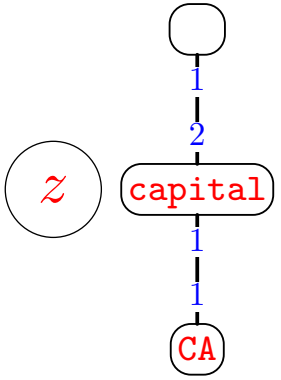
Los Angeles

Statistical: how to parametrize mapping from sentence to logical form?

What is the most populous city in California?

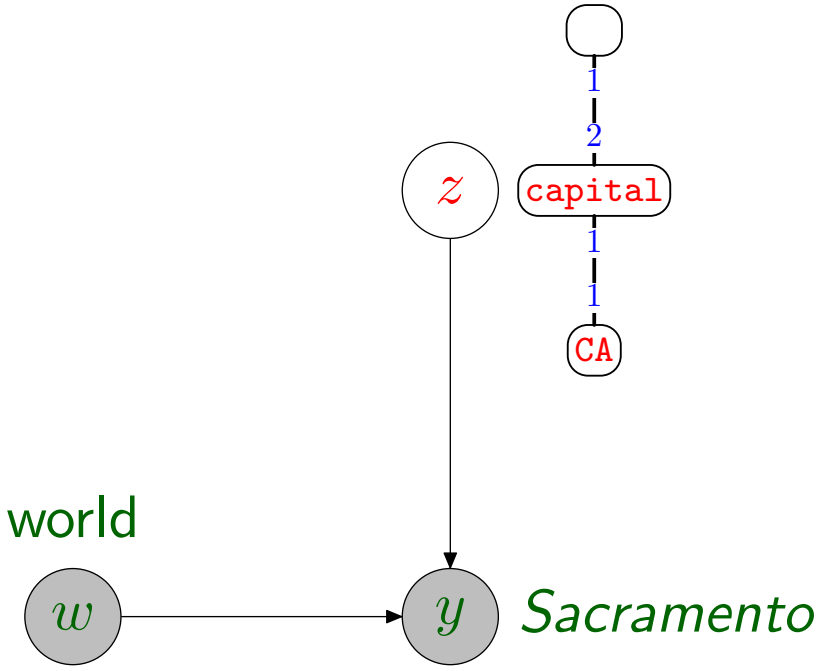
$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$

Graphical Model

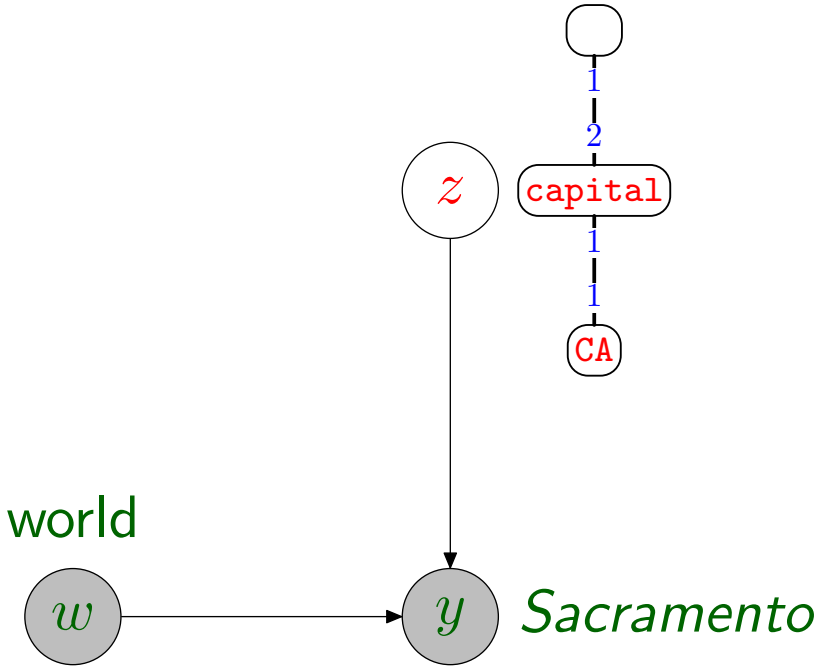


world

Graphical Model



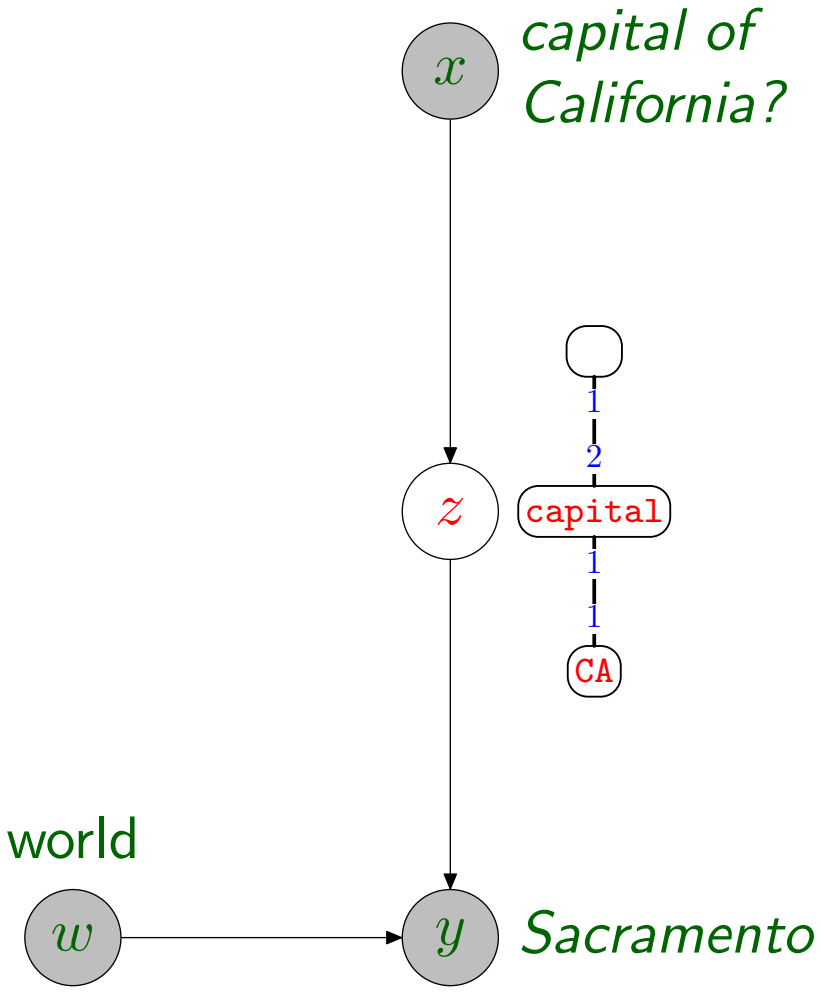
Graphical Model



Interpretation: $p(y \mid z, w)$

(deterministic)

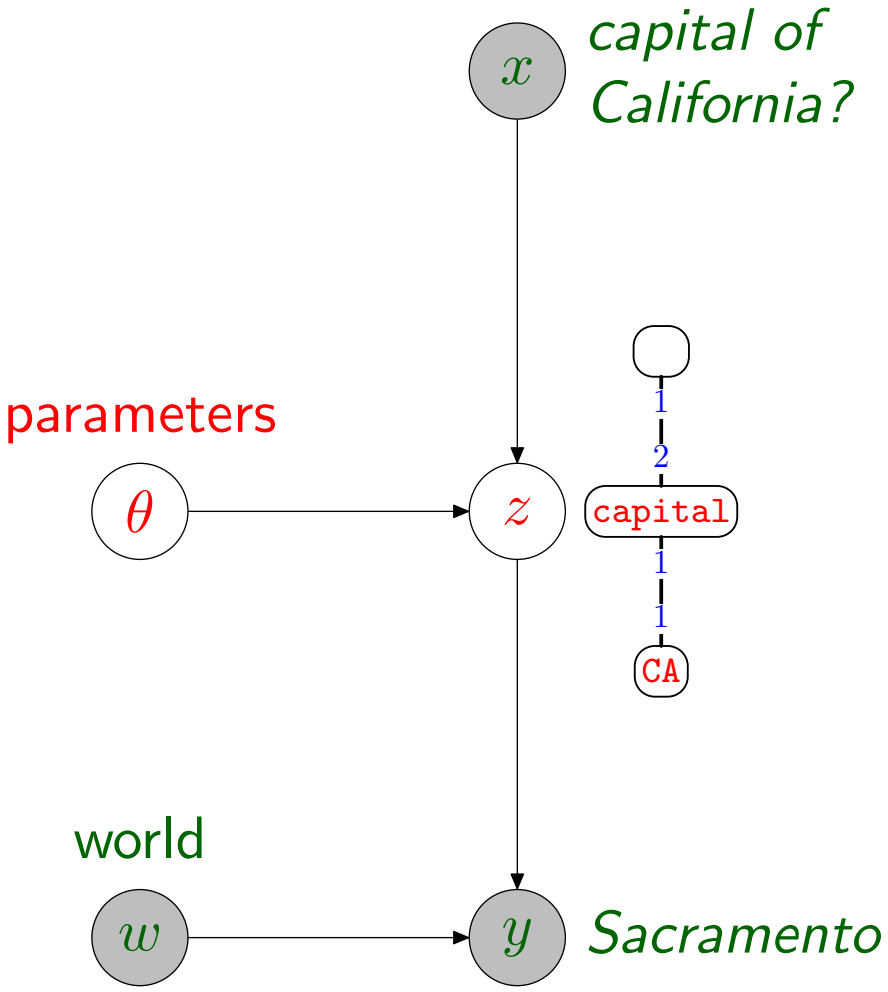
Graphical Model



Interpretation: $p(y \mid z, w)$

(deterministic)

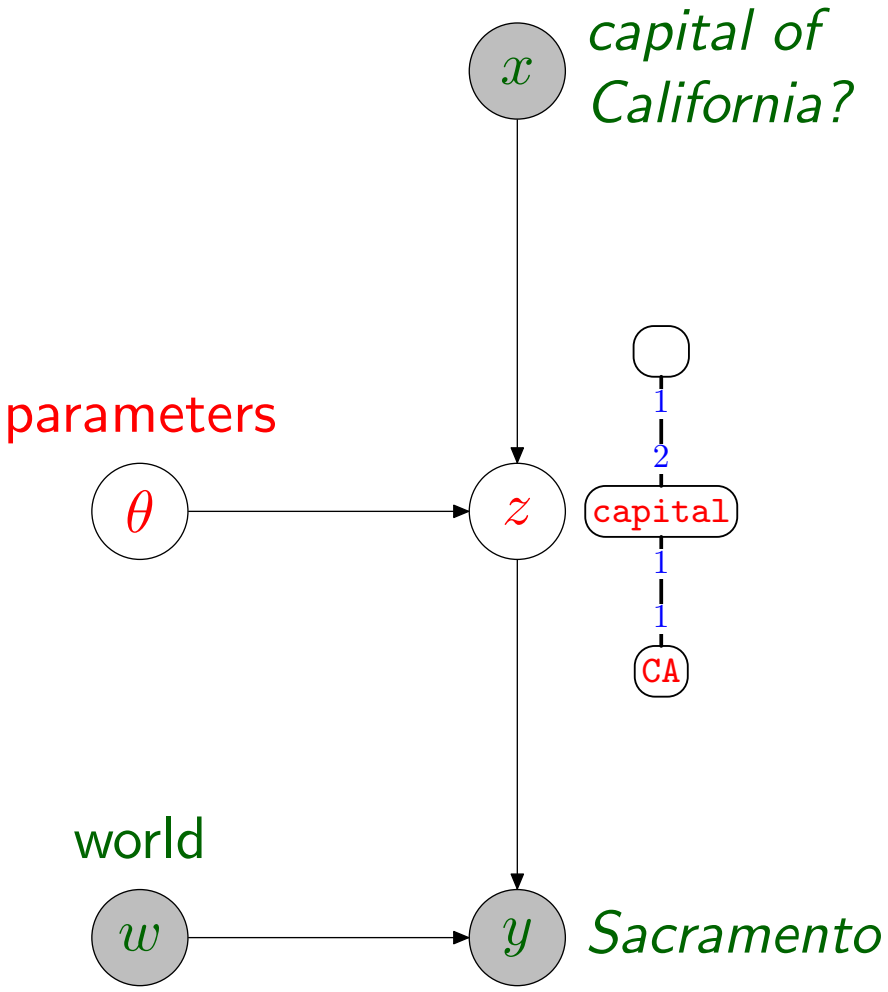
Graphical Model



Interpretation: $p(y \mid z, w)$

(deterministic)

Graphical Model



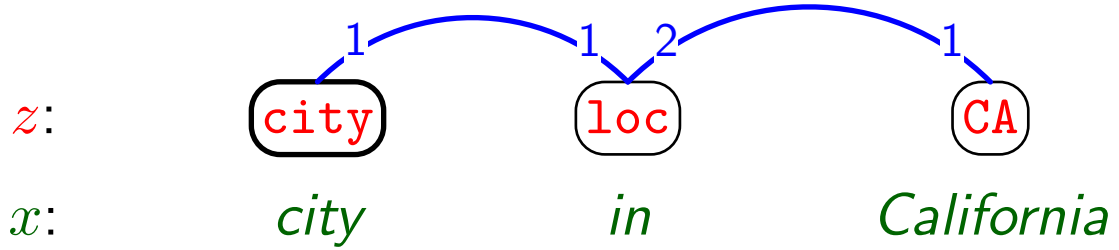
Semantic Parsing: $p(z \mid x, \theta)$

(probabilistic)

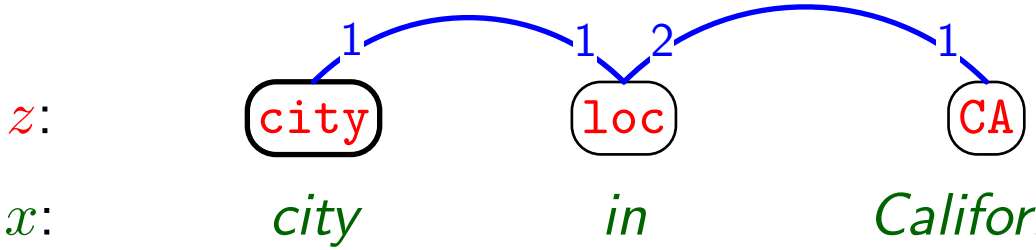
Interpretation: $p(y \mid z, w)$

(deterministic)

Semantic Parsing Log-linear Model

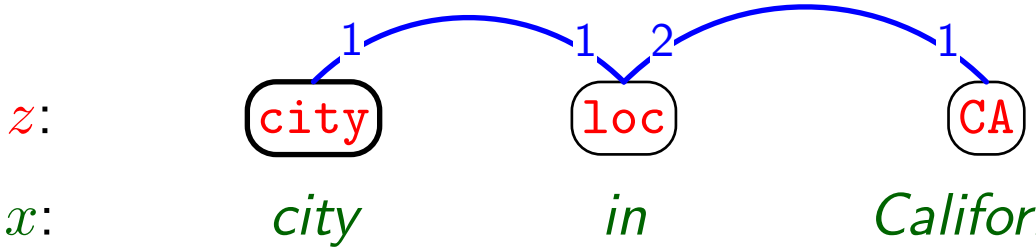


Semantic Parsing Log-linear Model



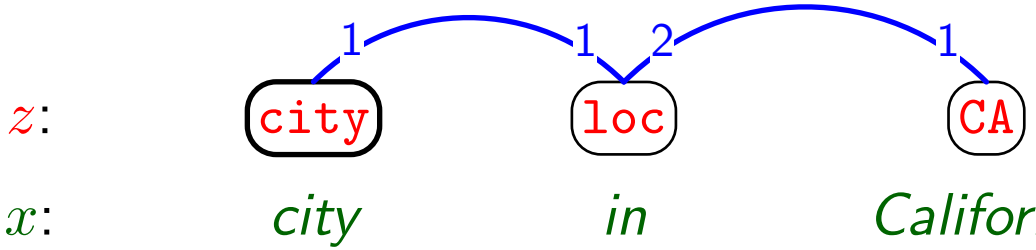
$$\text{features}(x, z) = \begin{pmatrix} \\ \\ \\ \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



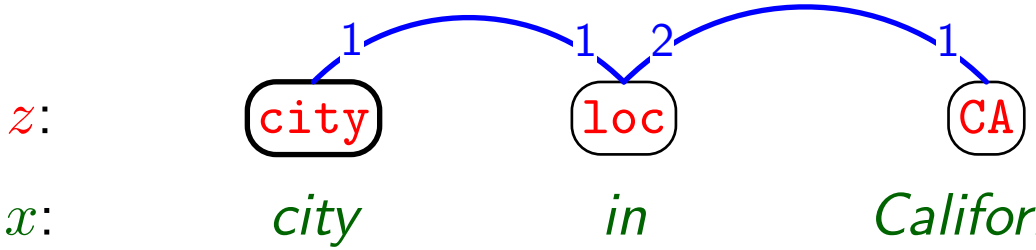
$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} : 1 \\ \\ \\ \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



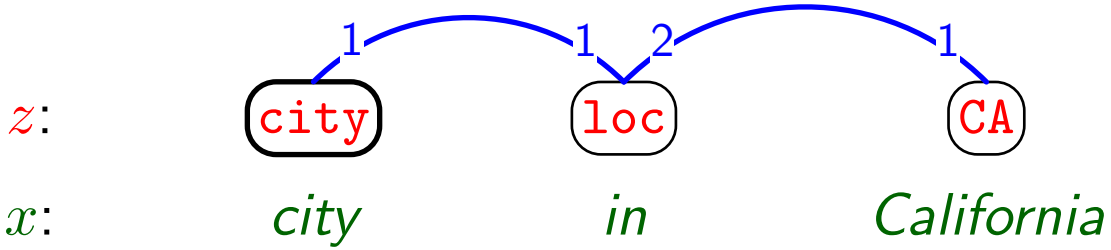
$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} & : 1 \\ \texttt{city}\text{-}1\text{-}1\text{-}\texttt{loc} & : 1 \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} & : 1 \\ \texttt{city}\text{-}1\text{-}1\text{-}\texttt{loc} & : 1 \\ \cdots & \end{pmatrix} \in \mathbb{R}^d$$

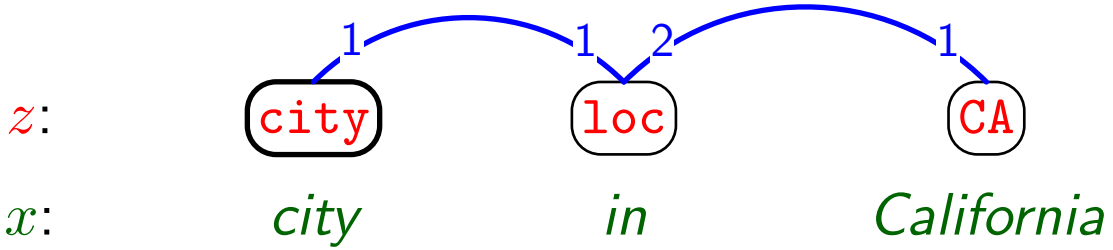
Semantic Parsing Log-linear Model



$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} & : 1 \\ \texttt{city}\text{-}1\text{-}1\text{-}\texttt{loc} & : 1 \\ \cdots & \end{pmatrix} \in \mathbb{R}^d$$

$$\text{score}(x, z) = \text{features}(x, z) \cdot \theta$$

Semantic Parsing Log-linear Model



$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} & : 1 \\ \texttt{city}\text{-}1\text{-}1\text{-}\texttt{loc} & : 1 \\ \cdots & \end{pmatrix} \in \mathbb{R}^d$$

$$\text{score}(x, z) = \text{features}(x, z) \cdot \theta$$

$$p(z \mid x, \theta) = \frac{e^{\text{score}(x,z)}}{\sum_{z' \in \mathcal{Z}(x)} e^{\text{score}(x,z')}}$$

Learning

Objective Function:

$$p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

Learning

Objective Function:

$$\max_{\theta} \quad p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0, 0, \ldots, 0)$

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0, 0, \ldots, 0)$ → enumerate/score DCS trees →

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0, 0, \ldots, 0)$

k-best list

`tree1` ✗
`tree2` ✗
`tree3` ✓
`tree4` ✗
`tree5` ✗

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0.2, -1.3, \ldots, 0.7)$

enumerate/score DCS trees →

← numerical optimization (L-BFGS)

k-best list

`tree1` ✗
`tree2` ✗
`tree3` ✓
`tree4` ✗
`tree5` ✗

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0.2, -1.3, \ldots, 0.7)$

enumerate/score DCS trees →

← numerical optimization (L-BFGS)

k-best list

`tree3` ✔
`tree8` ✔
`tree6` ✘
`tree2` ✘
`tree4` ✘

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0.3, -1.4, \ldots, 0.6)$

enumerate/score DCS trees →

← numerical optimization (L-BFGS)

k-best list

`tree3` ✔
`tree8` ✔
`tree6` ✘
`tree2` ✘
`tree4` ✘

Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w)\, p(z \mid x, \theta)$$

Interpretation **Semantic parsing**

EM-like Algorithm:

parameters θ

$(0.3, -1.4, \ldots, 0.6)$

enumerate/score DCS trees →

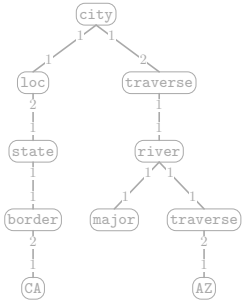
← numerical optimization (L-BFGS)

k-best list

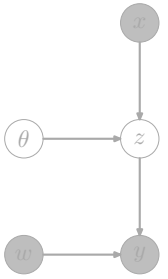
`tree3` ✔
`tree8` ✔
`tree2` ✘
`tree4` ✘
`tree9` ✘

Outline

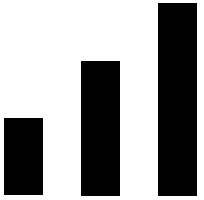
Representation



Learning



Experiments



US Geography Benchmark

Standard semantic parsing benchmark since 1990s

- 600 training examples, 280 test examples

US Geography Benchmark

Standard semantic parsing benchmark since 1990s

600 training examples, 280 test examples

What is the highest point in Florida?

How many states have a city called Rochester?

What is the longest river that runs through a state that borders Tennessee?

Of the states washed by the Mississippi river which has the lowest point?

...

US Geography Benchmark

Standard semantic parsing benchmark since 1990s

600 training examples, 280 test examples

What is the highest point in Florida?

$\Rightarrow$ `answer(A,highest(A,(place(A),loc(A,B),const(B,stateid(florida)))))`

How many states have a city called Rochester?

$\Rightarrow$ `answer(A,count(B,(state(B),loc(C,B),const(C,cityid(rochester,_))),A))`

What is the longest river that runs through a state that borders Tennessee?

$\Rightarrow$ `answer(A,longest(A,(river(A),traverse(A,B),state(B),next_to(B,C),const(C,stateid(tennessee)))))`

Of the states washed by the Mississippi river which has the lowest point?

$\Rightarrow$ `answer(A,lowest(B,(state(A),traverse(C,A),const(C,riverid(mississippi)),loc(B,A),place(B))))`

...

Supervision in past work: question + program

US Geography Benchmark

Standard semantic parsing benchmark since 1990s

600 training examples, 280 test examples

What is the highest point in Florida?
⇒ *Walton County*

How many states have a city called Rochester?
⇒ *2*

What is the longest river that runs through a state that borders Tennessee?
⇒ *Missouri*

Of the states washed by the Mississippi river which has the lowest point?
⇒ *Louisiana*

...

Supervision in past work: question + program

Supervision in this work: question + answer

Input to Learning Algorithm

Training data (600 examples)

What is the highest point in Florida?	⇒	*Walton County*
How many states have a city called Rochester?	⇒	*2*
What is the longest river that runs through a state that borders Tennessee?	⇒	*Missouri*
Of the states washed by the Mississippi river which has the lowest point?	⇒	*Louisiana*
...		...

Input to Learning Algorithm

Training data (600 examples)

What is the highest point in Florida?	⇒	*Walton County*
How many states have a city called Rochester?	⇒	*2*
What is the longest river that runs through a state that borders Tennessee?	⇒	*Missouri*
Of the states washed by the Mississippi river which has the lowest point?	⇒	*Louisiana*
...		...

Lexicon (20 general, 22 specific)

`no`	⇒	*no*
`argmax`	⇒	*most*
`city`	⇒	*city*
`state`	⇒	*state*
`mountain`	⇒	*mountain*
...		...

Input to Learning Algorithm

Training data (600 examples)

What is the highest point in Florida?	⇒	*Walton County*
How many states have a city called Rochester?	⇒	*2*
What is the longest river that runs through a state that borders Tennessee?	⇒	*Missouri*
Of the states washed by the Mississippi river which has the lowest point?	⇒	*Louisiana*
...		...

Lexicon (20 general, 22 specific)

`no`	⇒	*no*
`argmax`	⇒	*most*
`city`	⇒	*city*
`state`	⇒	*state*
`mountain`	⇒	*mountain*
...		...

World/Database

`city`

San Francisco
Chicago
Boston
...

`state`

Alabama
Alaska
Arizona
...

`loc`

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
...	...

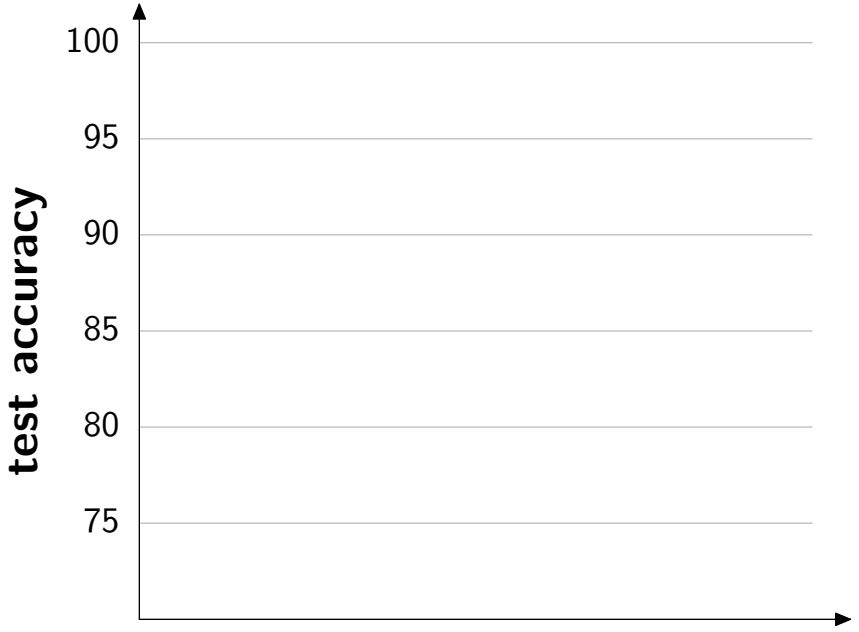
`border`

Washington	Oregon
Washington	Idaho
Oregon	Washington
...	...

... ...

Experiment 1

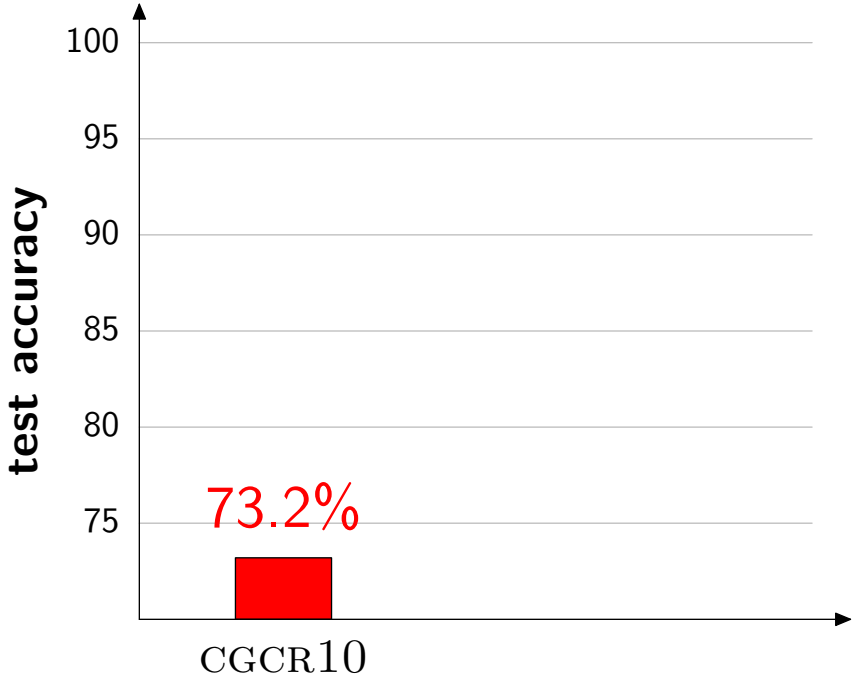
On GEO, 250 training examples, 250 test examples



Experiment 1

On GEO, 250 training examples, 250 test examples

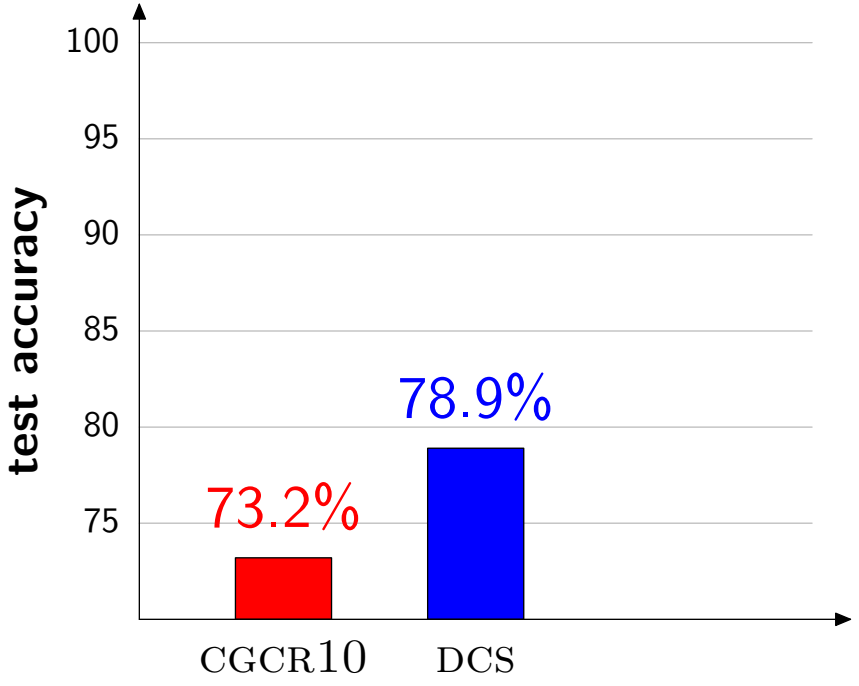
System	Description	Lexicon (gen./spec.)	Logical forms
CGCR10	FunQL [Clarke et al., 2010]	✔ ✔	✗



Experiment 1

On GEO, 250 training examples, 250 test examples

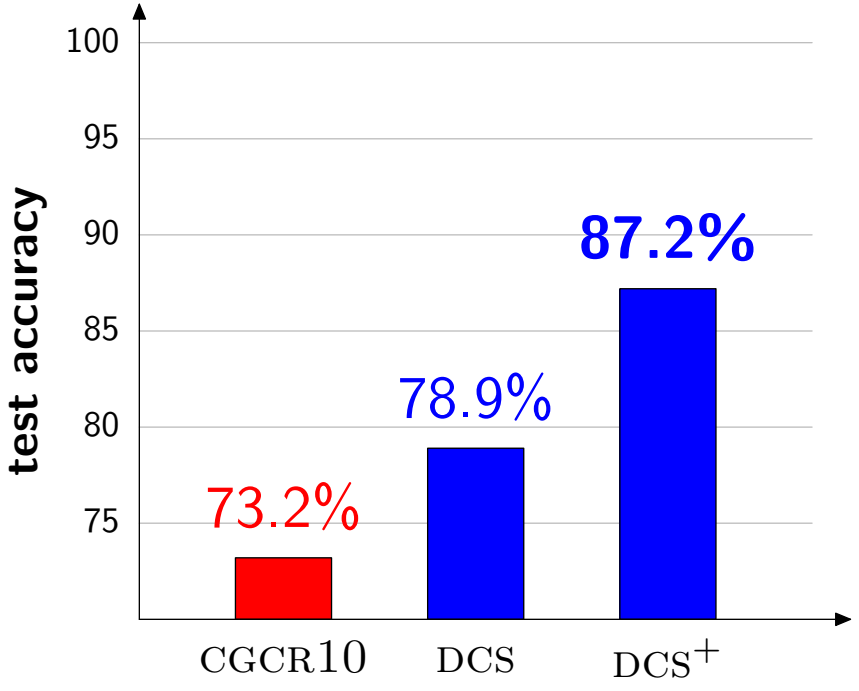
System	Description	Lexicon (gen./spec.)	Logical forms
CGCR10	FunQL [Clarke et al., 2010]	✓ ✓	✗
LJK11	DCS [Liang et al., 2011]	✓ ✗	✗



Experiment 1

On GEO, 250 training examples, 250 test examples

System	Description	Lexicon (gen./spec.)	Logical forms
CGCR10	FunQL [Clarke et al., 2010]	✓ ✓	✗
LJK11	DCS [Liang et al., 2011]	✓ ✗	✗
LJK11$^+$	DCS [Liang et al., 2011]	✓ ✓	✗



Experiment 2

On GEO, 600 training examples, 280 test examples

Experiment 2

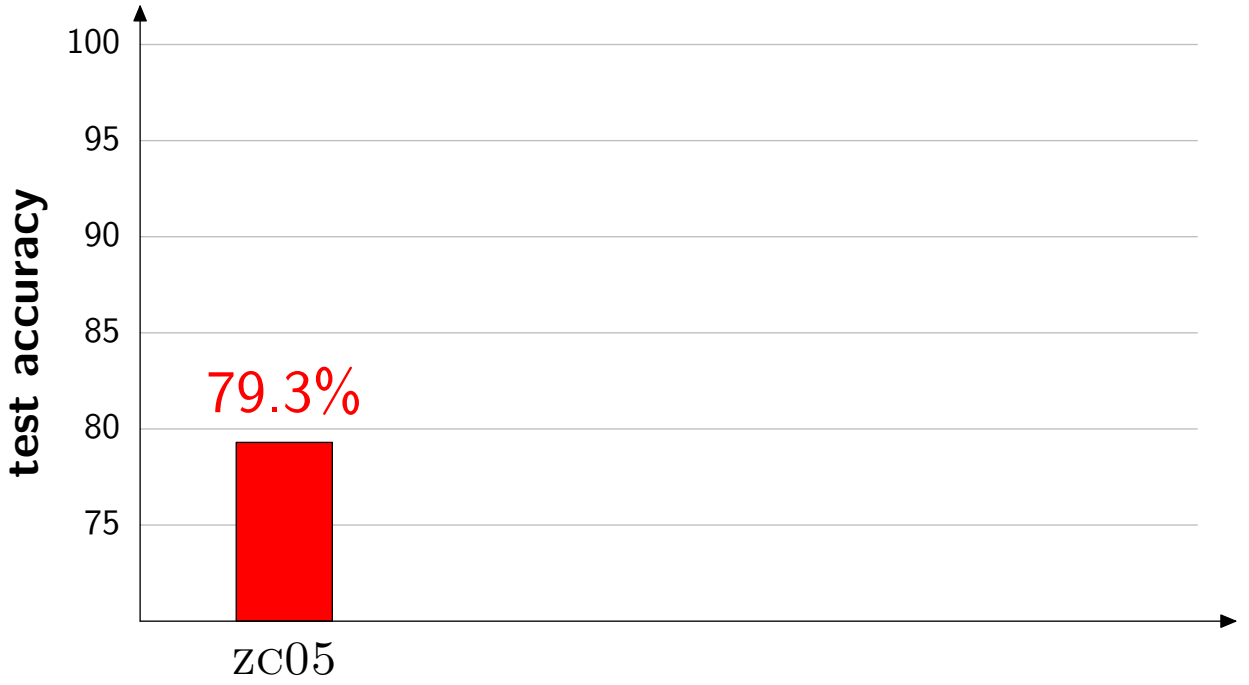
On GEO, 600 training examples, 280 test examples

System	Description	Lexicon	Logical forms

Experiment 2

On GEO, 600 training examples, 280 test examples

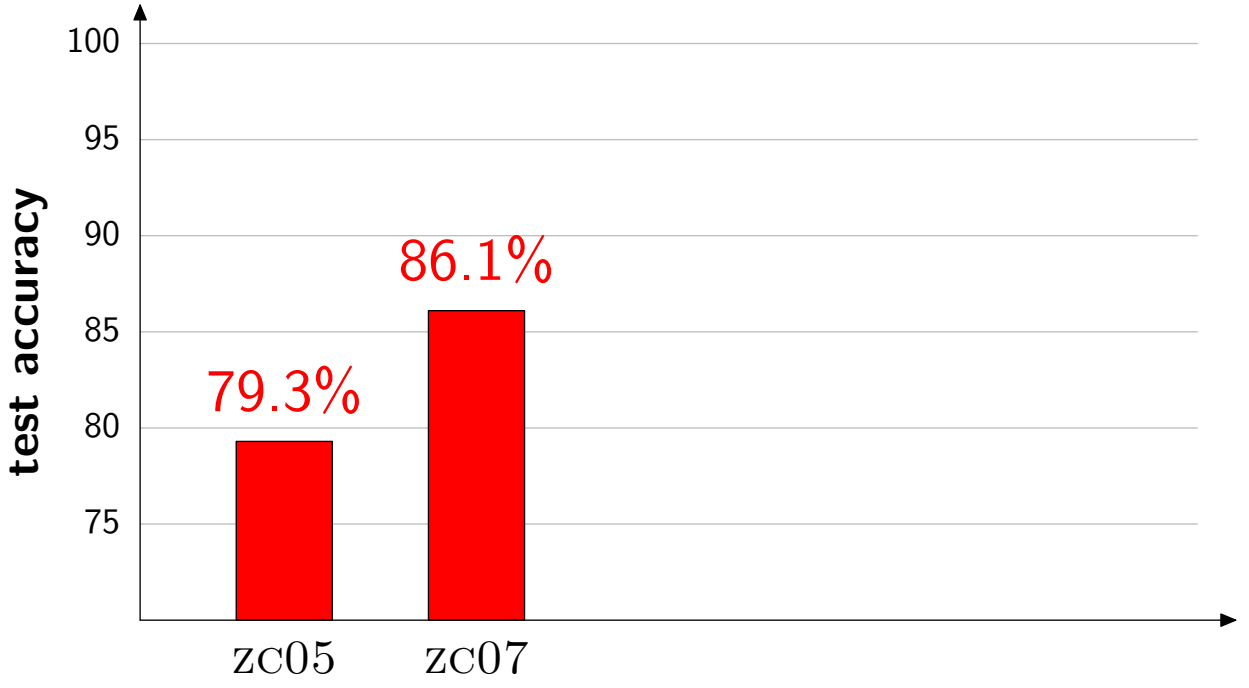
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

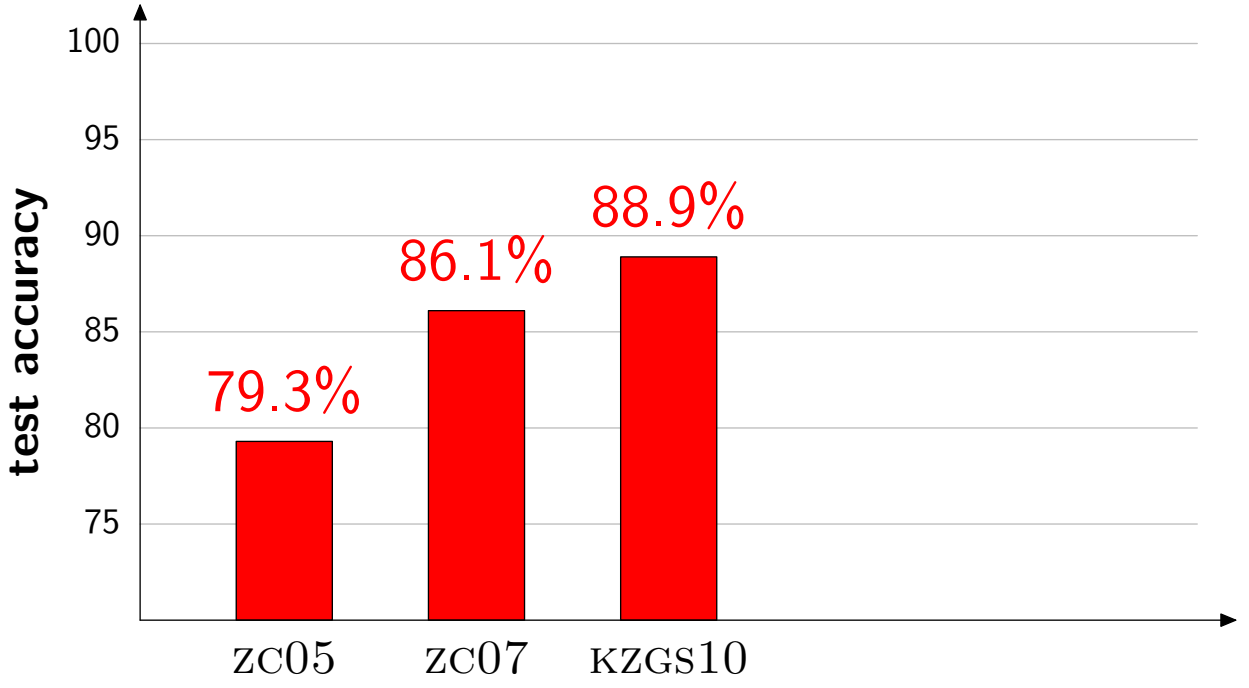
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

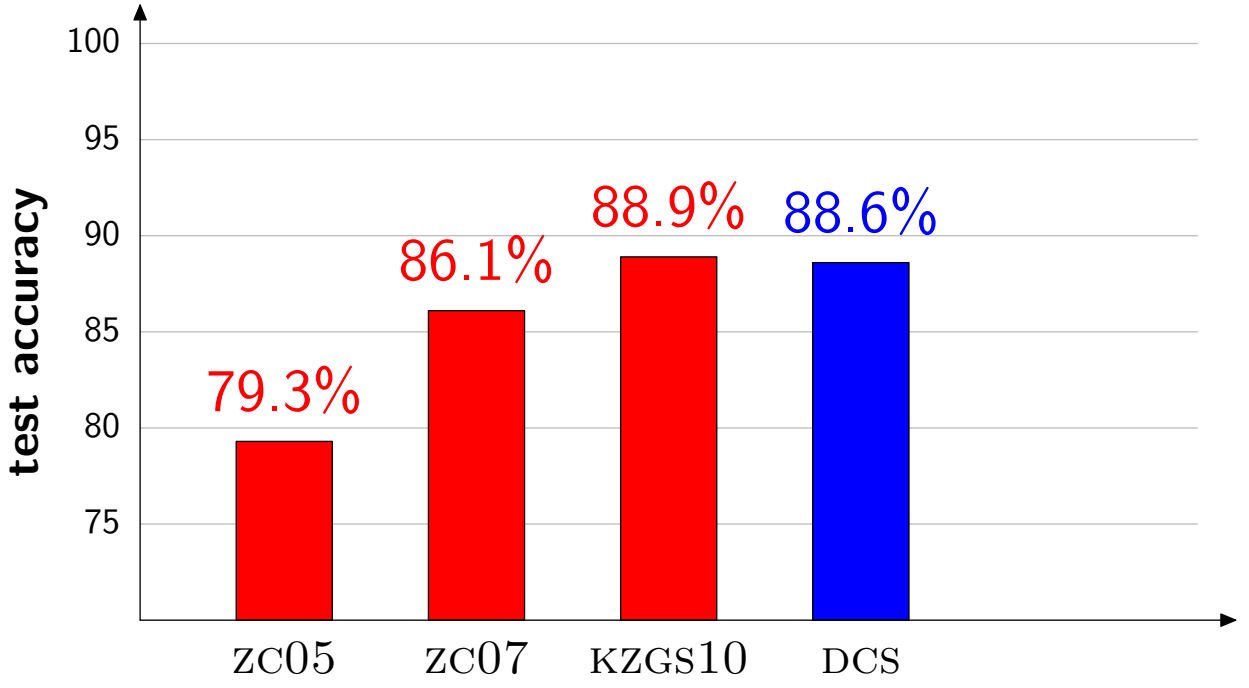
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓
KZGS10	CCG w/unification [Kwiatkowski et al., 2010]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

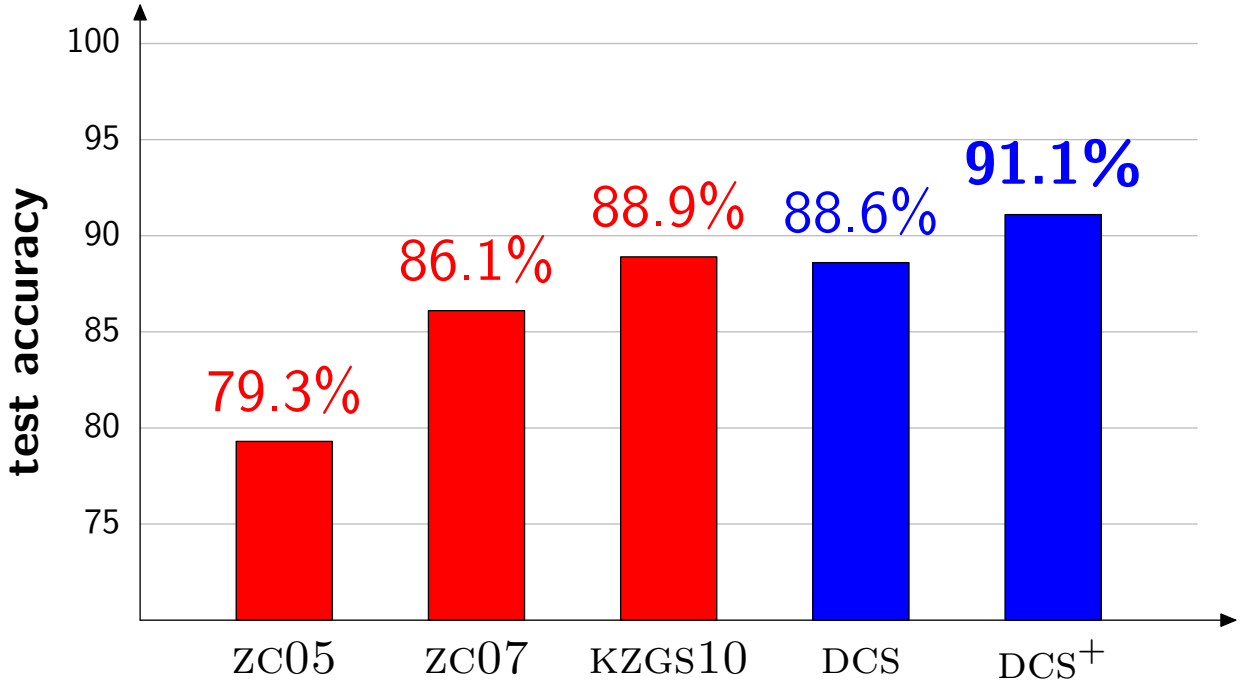
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓
KZGS10	CCG w/unification [Kwiatkowski et al., 2010]	✗ ✗	✓
LJK11	DCS [Liang et al., 2011]	✓ ✗	✗



Experiment 2

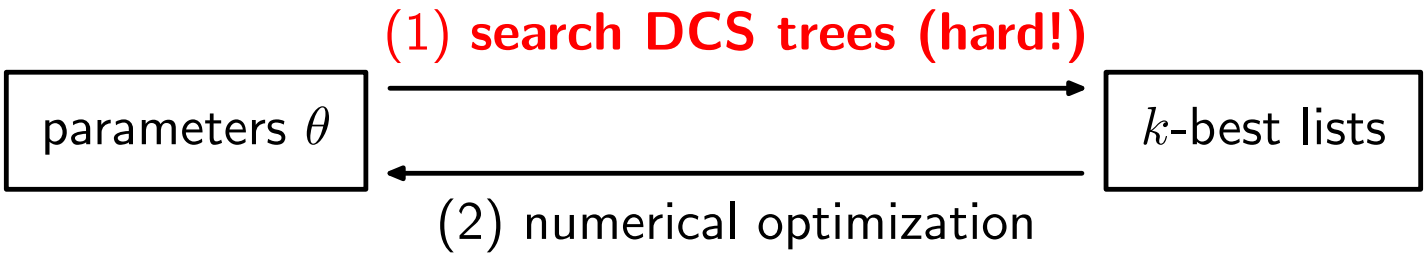
On GEO, 600 training examples, 280 test examples

System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓
KZGS10	CCG w/unification [Kwiatkowski et al., 2010]	✗ ✗	✓
LJK11	DCS [Liang et al., 2011]	✓ ✗	✗
LJK11$^+$	DCS [Liang et al., 2011]	✓ ✓	✗

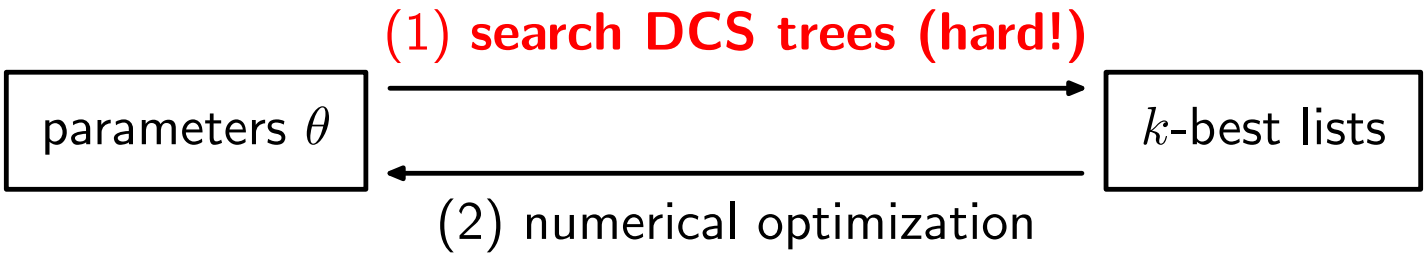


Some Intuition on Learning

Some Intuition on Learning

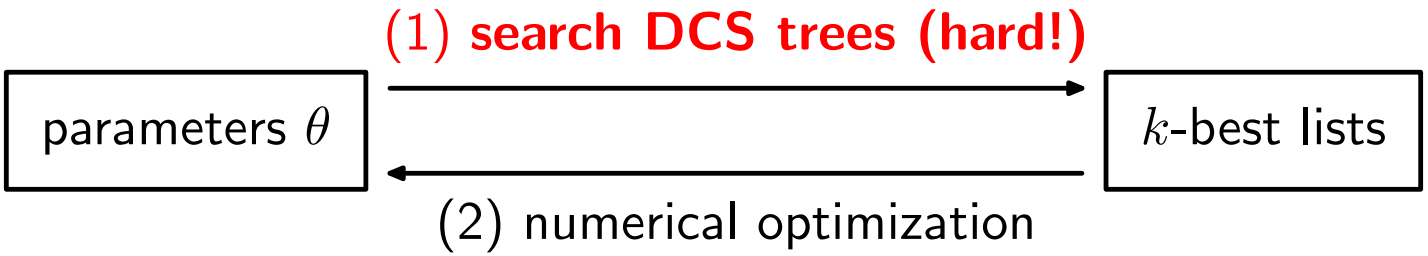


Some Intuition on Learning

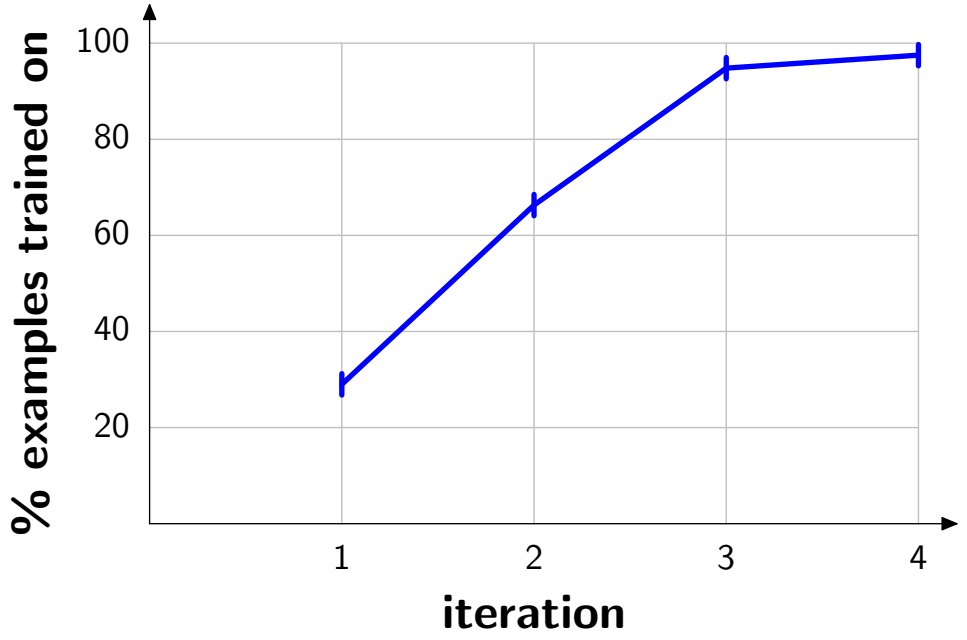


If no DCS tree on k-best list is correct, skip example in (2)

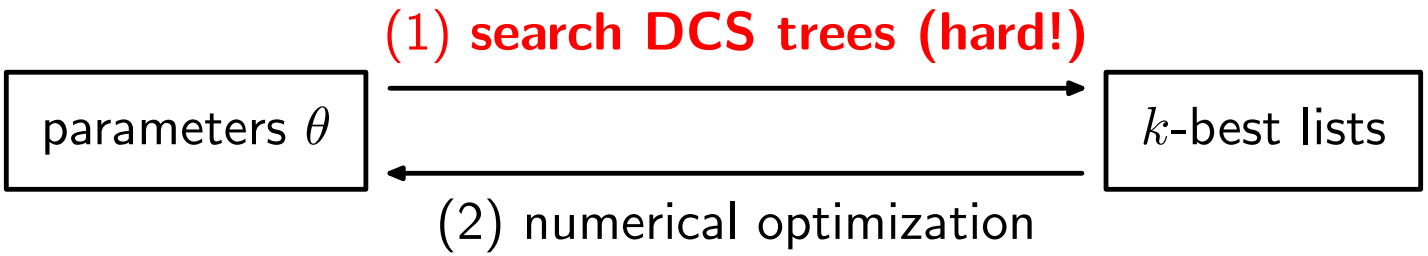
Some Intuition on Learning



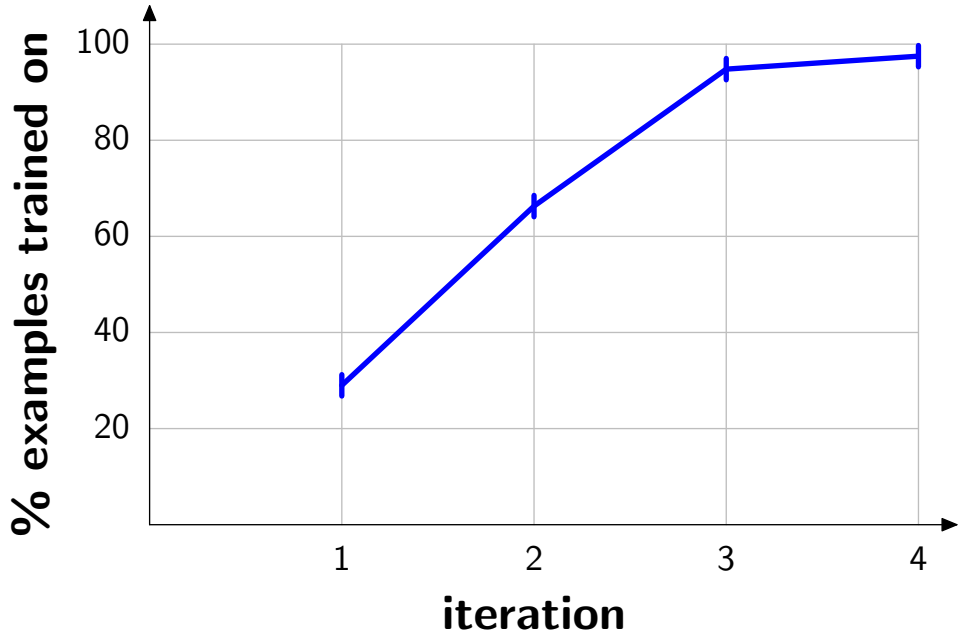
If no DCS tree on k-best list is correct, skip example in (2)



Some Intuition on Learning



If no DCS tree on k-best list is correct, skip example in (2)



Effect: automatic curriculum learning, learning improves search

Current Limitations

Unknown facts: *How far is Los Angeles from Boston?*

Database has no distance information

Current Limitations

Unknown facts: *How far is Los Angeles from Boston?*

Database has no distance information

Unknown concepts: *What states are landlocked?*

Need to induce database view for $\texttt{landlocked}(x) = \neg\texttt{border}(x, \texttt{ocean})$

Current Limitations

Unknown facts: *How far is Los Angeles from Boston?*

Database has no distance information

Unknown concepts: *What states are landlocked?*

Need to induce database view for $\texttt{landlocked}(x) = \neg\texttt{border}(x, \texttt{ocean})$

Unknown words: *What is the largest settlement in California?*

Training examples do not contain the word *settlement*

Summary


sentence
Semantic Parser
logical form
Interpretation
denotation

Summary

Learning from Weak Supervision

- Model logical form as latent variable
- Semantic formalisms: CCG, DCS

Summary

Learning from Weak Supervision

- Model logical form as latent variable
- Semantic formalisms: CCG, DCS

Strategy:

- Lexicon/grammar generates set of candidate logical forms
- Learned feature weights capture linguistic generalizations

Summary

Learning from Weak Supervision

- Model logical form as latent variable
- Semantic formalisms: CCG, DCS

Strategy:

- Lexicon/grammar generates set of candidate logical forms
- Learned feature weights capture linguistic generalizations

Challenges:

- Open-ended lexical generalization
- Large search space
- Discourse, pragmatics