Learning Compositional Semantics

CS224U: Natural Language Understanding

Feb. 9, 2012

Percy Liang

Google/Stanford

Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

$$\downarrow$$

$$\neg \exists x.\mathtt{state}(x) \wedge \mathtt{border}(\mathtt{AK}, x)$$

Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

$$\Downarrow$$

$$\neg\exists x.\mathtt{state}(x) \wedge \mathtt{border}(\mathtt{AK}, x)$$

We assumed the following were given:

Lexicon:	*no*	$\Rightarrow$	$\text{DT} : \lambda P.\lambda Q.\neg\exists x.P(x) \wedge Q(x)$
	states	$\Rightarrow$	$\text{N} : \lambda x.\mathtt{state}(x)$

Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

$$\neg\exists x.\texttt{state}(x) \wedge \texttt{border}(\texttt{AK}, x)$$

We assumed the following were given:

Lexicon:	*no*	$\Rightarrow$	$\text{DT} : \lambda P.\lambda Q.\neg\exists x.P(x) \wedge Q(x)$
	states	$\Rightarrow$	$\text{N} : \lambda x.\texttt{state}(x)$
Grammar:	$\text{DT} : f \quad \text{N} : a$	$\Rightarrow$	$\text{NP} : f(a)$

Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

$$\neg \exists x.\mathtt{state}(x) \wedge \mathtt{border}(\mathtt{AK}, x)$$

We assumed the following were given:

Lexicon:	*no*	$\Rightarrow$	$\mathrm{DT} : \lambda P.\lambda Q.\neg \exists x.P(x) \wedge Q(x)$
	states	$\Rightarrow$	$\mathrm{N} : \lambda x.\mathtt{state}(x)$
Grammar:	$\mathrm{DT} : f \quad \mathrm{N} : a$	$\Rightarrow$	$\mathrm{NP} : f(a)$

Questions:

But where do they come from?
What if a sentence generates multiple logical forms?
What if a sentences is slightly ungrammatical?

Outline

Today: building real semantic parsers!

$$\text{sentence} \longrightarrow \boxed{\text{Semantic Parser}} \longrightarrow \text{logical form}$$

Outline

Today: building real semantic parsers!

sentence $\longrightarrow$ $\boxed{\text{Semantic Parser}}$ $\longrightarrow$ logical form

Strategy: break up complex mapping into two parts

Outline

Today: building real semantic parsers!

sentence $\longrightarrow$ $\boxed{\text{Semantic Parser}}$ $\longrightarrow$ logical form

Strategy: break up complex mapping into two parts

Representation (Lexicon/Grammar):

- Should be simple, require minimal human effort
- Generates set of candidate logical forms
  Allow overgeneration: *state* $\quad\Rightarrow\quad \mathrm{N} : \lambda x.\mathtt{river}(x)$

Outline

Today: building real semantic parsers!

sentence $\longrightarrow$ $\boxed{\text{Semantic Parser}}$ $\longrightarrow$ logical form

Strategy: break up complex mapping into two parts

Representation (Lexicon/Grammar):

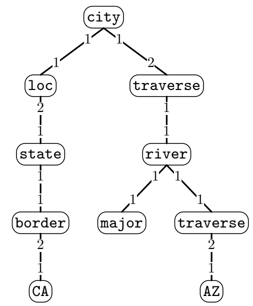
- Should be simple, require minimal human effort
- Generates set of candidate logical forms
  Allow overgeneration: *state* $\Rightarrow$ $\mathrm{N} : \lambda x.\texttt{river}(x)$

Learning:

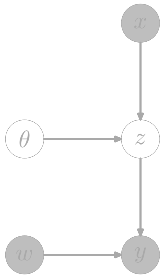
- Score/rank candidates based on features
- Optimize feature weights discriminatively to minimize training error

Outline

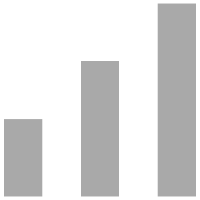
Representation



Learning



Experiments



Semantic Formalisms

sentence $\rightarrow$ $\boxed{\text{Semantic Parser}}$ $\rightarrow$ logical form

Semantic Formalisms

sentence → $\boxed{\text{Semantic Parser}}$ → logical form → $\boxed{\text{Interpretation}}$ → denotation

Semantic Formalisms

sentence $\rightarrow$ $\boxed{\text{Semantic Parser}}$ $\rightarrow$ logical form $\rightarrow$ $\boxed{\text{Interpretation}}$ $\rightarrow$ denotation

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Semantic Formalisms

sentence → Semantic Parser → logical form → Interpretation → denotation

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation (mapping from world to true/false)

Semantic Formalisms

sentence → Semantic Parser → logical form → Interpretation → denotation

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation (mapping from world to true/false)

Compositional: meaning (logical form) of phrase computed from combining meaning of sub-phrases

Semantic Formalisms

sentence → **Semantic Parser** → logical form → **Interpretation** → denotation

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation (mapping from world to true/false)

Compositional: meaning (logical form) of phrase computed from combining meaning of sub-phrases

Semantic Formalisms:

- Combinatory Categorial Grammar (CCG)
- Dependency-Based Compositional Semantics (DCS)

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

$$\mathrm{NP} \quad \mathrm{VP} \quad \Rightarrow \quad \mathrm{S}$$

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

$$\mathrm{NP} \quad \mathrm{VP} \quad \Rightarrow \quad \mathrm{S} \qquad\qquad \mathrm{NP} \quad \mathrm{S}\backslash\mathrm{NP} \quad \Rightarrow \quad \mathrm{S}$$

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

$$\mathrm{NP} \quad \mathrm{VP} \quad \Rightarrow \quad \mathrm{S} \qquad\qquad \mathrm{NP} \quad \mathrm{S}\backslash\mathrm{NP} \quad \Rightarrow \quad \mathrm{S}$$

$$\mathrm{V} \quad \mathrm{NP} \quad \Rightarrow \quad \mathrm{VP}$$

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

$$\mathrm{NP} \quad \mathrm{VP} \quad \Rightarrow \quad \mathrm{S} \qquad\qquad \mathrm{NP} \quad \mathrm{S}\backslash\mathrm{NP} \quad \Rightarrow \quad \mathrm{S}$$

$$\mathrm{V} \quad \mathrm{NP} \quad \Rightarrow \quad \mathrm{VP} \qquad\qquad (\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} \quad \mathrm{NP} \quad \Rightarrow \quad \mathrm{S}\backslash\mathrm{NP}$$

Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

$$\mathrm{NP} \quad \mathrm{VP} \quad \Rightarrow \quad \mathrm{S} \qquad\qquad \mathrm{NP} \quad \mathrm{S}\backslash\mathrm{NP} \quad \Rightarrow \quad \mathrm{S}$$

$$\mathrm{V} \quad \mathrm{NP} \quad \Rightarrow \quad \mathrm{VP} \qquad\qquad (\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} \quad \mathrm{NP} \quad \Rightarrow \quad \mathrm{S}\backslash\mathrm{NP}$$

In general:

Base categories: $\mathrm{S}, \mathrm{NP}, \mathrm{N}$

Derived categories: if X, Y are categories, then X/Y and $X\backslash Y$ are too

Combinatory Categorial Grammar (CCG)

Lexicon:

Alice	$\mathrm{NP} : \texttt{alice}$
Bob	$\mathrm{NP} : \texttt{bob}$
saw	$(\mathrm{S} \backslash \mathrm{NP}) / \mathrm{NP} : \lambda y . \lambda x . \texttt{saw}(x, y)$

Combinatory Categorial Grammar (CCG)

Lexicon:

Alice	$\mathrm{NP} : \texttt{alice}$
Bob	$\mathrm{NP} : \texttt{bob}$
saw	$(\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} : \lambda y.\lambda x.\texttt{saw}(x, y)$

Grammar (template):

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

Backward application (<) $\quad X : a \quad Y\backslash X : f \quad \Rightarrow \quad Y : f(a)$

Combinatory Categorial Grammar (CCG)

Lexicon:

Alice	$\mathrm{NP} : \texttt{alice}$
Bob	$\mathrm{NP} : \texttt{bob}$
saw	$(\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} : \lambda y.\lambda x.\texttt{saw}(x, y)$

Grammar (template):

Forward application ($>$)	$Y/X : f \quad X : a$	$\Rightarrow$	$Y : f(a)$
Backward application ($<$)	$X : a \quad Y\backslash X : f$	$\Rightarrow$	$Y : f(a)$

Derivation:

$$\mathrm{S} : \texttt{saw}(\texttt{alice}, \texttt{bob})$$

$<$

$\mathrm{NP} : \texttt{alice}$ $\qquad$ $\mathrm{S}\backslash\mathrm{NP} : \lambda x.\texttt{saw}(x, \texttt{bob})$

Alice $\qquad$ $>$

$(\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} : \lambda y.\lambda x.\texttt{saw}(x, y)$ $\qquad$ $\mathrm{NP} : \texttt{bob}$

saw $\qquad$ *Bob*

Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition $(\mathbf{B}>)$ $\quad Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z))$

Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition $(\mathbf{B}>)$ $\quad Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z))$

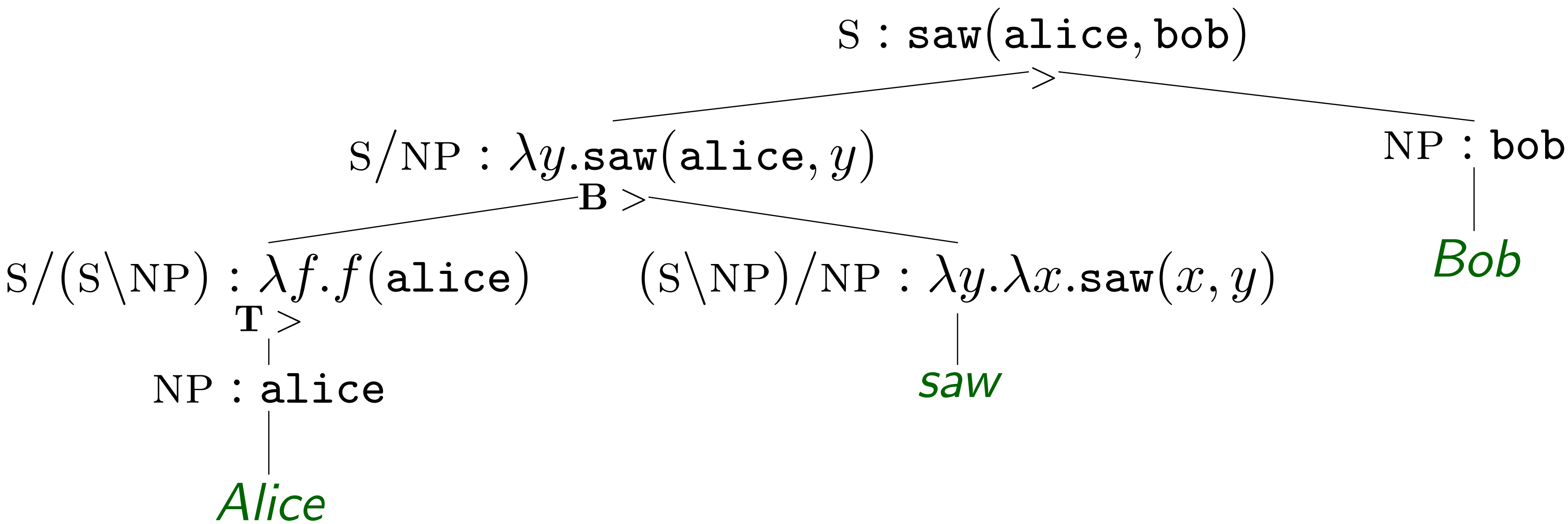
Type raising $(\mathbf{T}>)$ $\quad X : a \quad \Rightarrow \quad Y/(Y\backslash X) : \lambda f.f(a)$

Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition $(\mathbf{B}>)$ $\quad Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z))$

Type raising $(\mathbf{T}>)$ $\quad X : a \quad \Rightarrow \quad Y/(Y\backslash X) : \lambda f.f(a)$

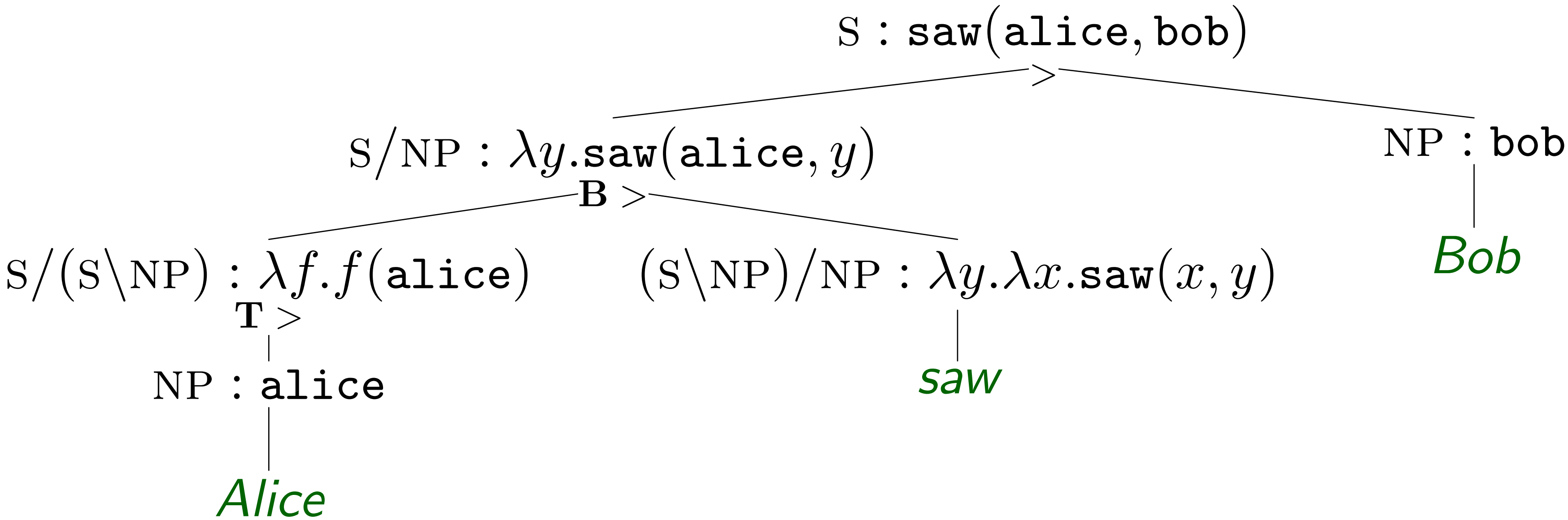


Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition ($\mathbf{B}>$) $\quad Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z))$

Type raising ($\mathbf{T}>$) $\quad X : a \quad \Rightarrow \quad Y/(Y\backslash X) : \lambda f.f(a)$



Composition creates non-traditional bracketing useful for right-node raising:

$\mathrm{S} : \mathtt{saw}(\mathtt{alice}, \mathtt{bob}) \wedge \mathtt{heard}(\mathtt{carol}, \mathtt{bob})$

[[Alice saw] and [Carol heard]] Bob.

CCG Meets Real Data

Non-contentful words:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$

Show me flights to Boston

CCG Meets Real Data

Non-contentful words:

$\lambda x.\mathtt{flight}(x) \wedge \mathtt{to}(x, \mathtt{boston})$

Show me flights to Boston

Solution: identity functions: *show me* $\Rightarrow$ $\mathrm{N}/\mathrm{N} : \lambda f.f$

CCG Meets Real Data

Non-contentful words:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$

Show me flights to Boston

Solution: identity functions: *show me* $\Rightarrow$ $\mathrm{N/N} : \lambda f.f$

Missing content:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$

Boston flights

CCG Meets Real Data

Non-contentful words:

$\lambda x.\mathtt{flight}(x) \wedge \mathtt{to}(x, \mathtt{boston})$

Show me flights to Boston

Solution: identity functions: *show me* $\Rightarrow$ $\mathrm{N}/\mathrm{N} : \lambda f.f$

Missing content:

$\lambda x.\mathtt{flight}(x) \wedge \mathtt{to}(x, \mathtt{boston})$

Boston flights

Solution: type-raising: $\mathrm{NP} : x$ $\Rightarrow$ $\mathrm{NP}/\mathrm{N} : \lambda f.\lambda a.f(a) \wedge \mathtt{to}(a, x)$

CCG Meets Real Data

Non-contentful words:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$
Show me flights to Boston

Solution: identity functions: *show me* $\quad \Rightarrow \quad \mathrm{N/N} : \lambda f.f$

Missing content:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$
Boston flights

Solution: type-raising: $\mathrm{NP} : x \quad \Rightarrow \quad \mathrm{NP/N} : \lambda f.\lambda a.f(a) \wedge \texttt{to}(a, x)$

Non-standard ordering:

$\lambda x.\texttt{flight}(x) \wedge \texttt{oneway}(x)$
flights one-way

CCG Meets Real Data

Non-contentful words:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$

Show me flights to Boston

Solution: identity functions: *show me* $\Rightarrow$ $\mathrm{N}/\mathrm{N} : \lambda f.f$

Missing content:

$\lambda x.\texttt{flight}(x) \wedge \texttt{to}(x, \texttt{boston})$

Boston flights

Solution: type-raising: $\mathrm{NP} : x \Rightarrow \mathrm{NP}/\mathrm{N} : \lambda f.\lambda a.f(a) \wedge \texttt{to}(a, x)$

Non-standard ordering:

$\lambda x.\texttt{flight}(x) \wedge \texttt{oneway}(x)$

flights one-way

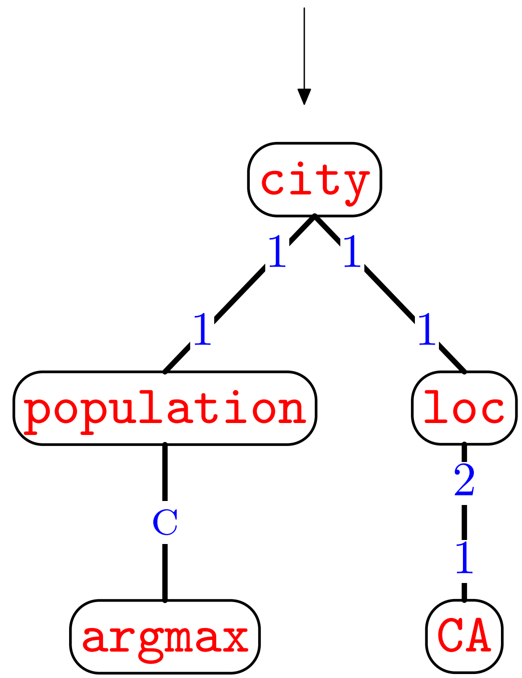
Solution: disharmonic combinators: $X : a \quad Y/X : f \Rightarrow Y : f(a)$

Dependency-Based Compositional Semantics (DCS)

What is the most populous city in California?

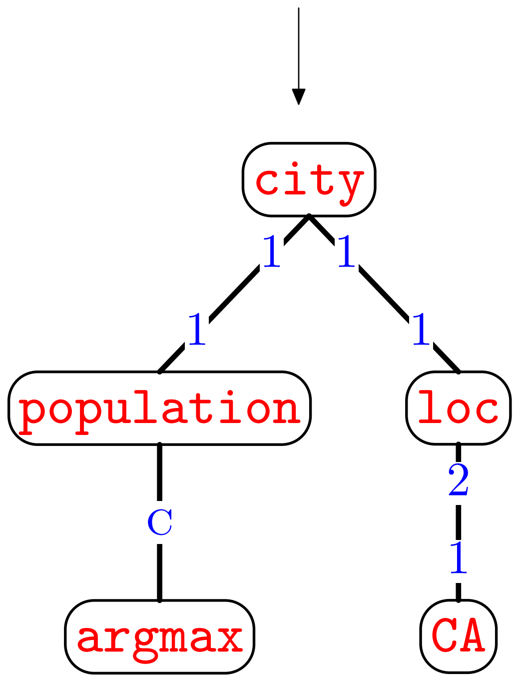
Dependency-Based Compositional Semantics (DCS)

What is the most populous city in California?



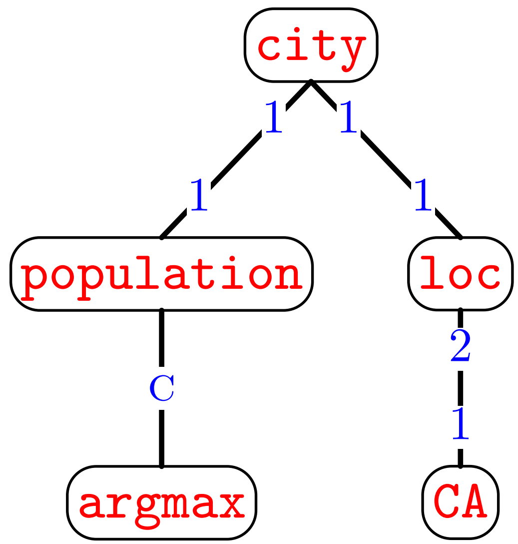
How to interpret the logical form?

What is the most populous city in California?



How to interpret the logical form?

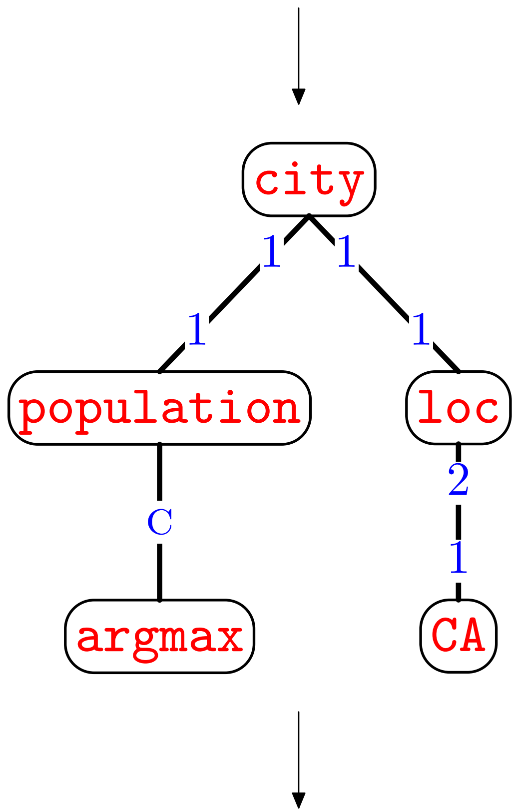
What is the most populous city in California?



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
...	...

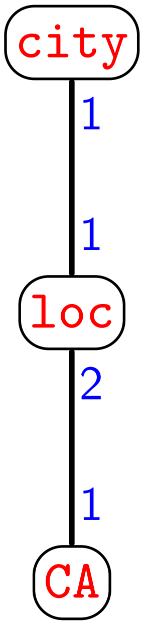
border

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

... ...

Basic DCS Trees

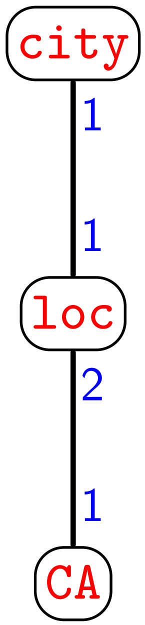
DCS tree



Database

Basic DCS Trees

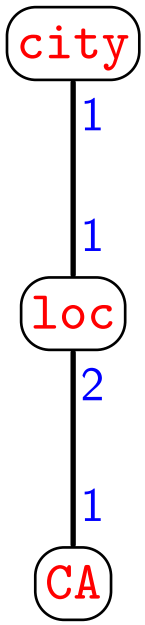
DCS tree **Constraints** **Database**



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

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

Database

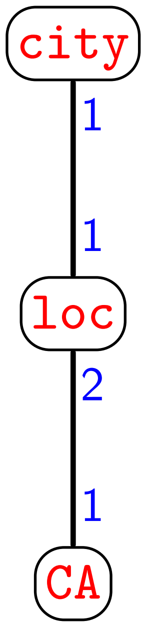
`city`

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
$\cdots$

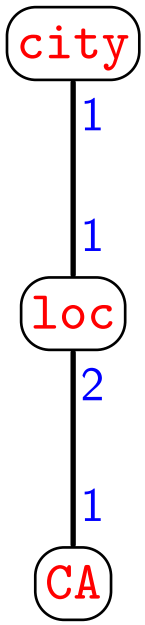
`loc`

Mount Shasta	California
San Francisco	California
Boston	Massachusetts
$\cdots$	$\cdots$

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

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$\ell \in \texttt{loc}$

$s \in \texttt{CA}$

Database

`city`

San Francisco
Chicago
Boston
...

`loc`

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

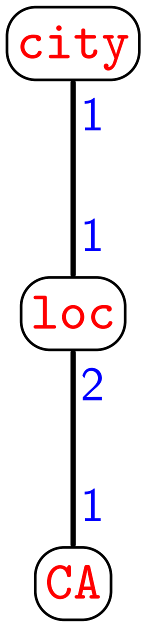
`CA`

California

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

Basic DCS Trees

DCS tree



Constraints

$c \in \texttt{city}$

$c_1 = \ell_1$

$\ell \in \texttt{loc}$

$s \in \texttt{CA}$

Database

`city`

San Francisco
Chicago
Boston
...

`loc`

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

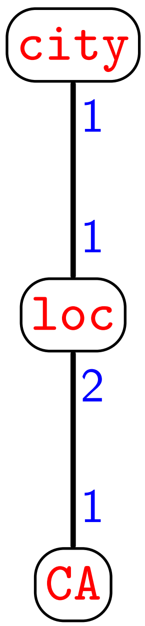
`CA`

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
...	...

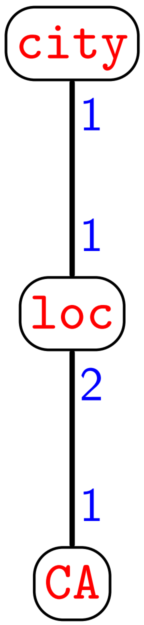
`CA`

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
...	...

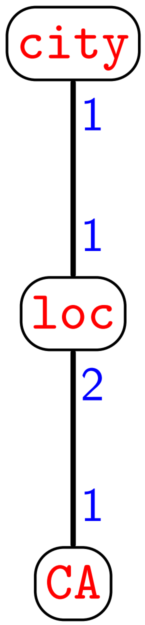
`CA`

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
...	...

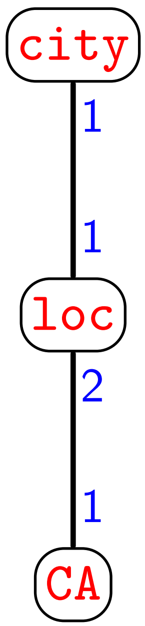
`CA`

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
...	...

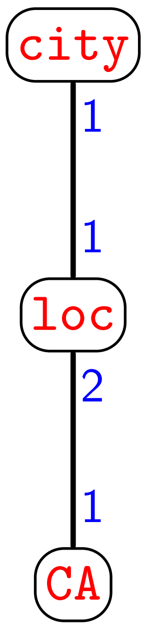
`CA`

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
...	...

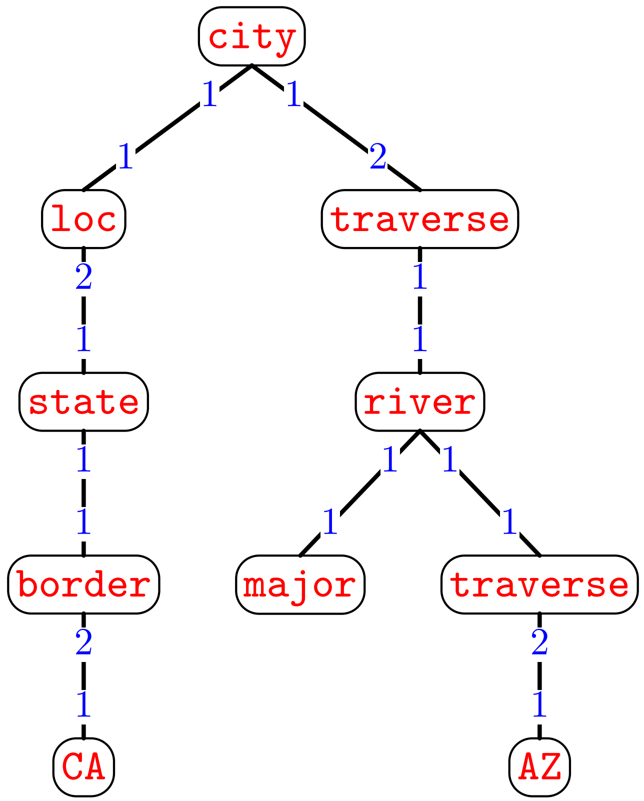
`CA`

California

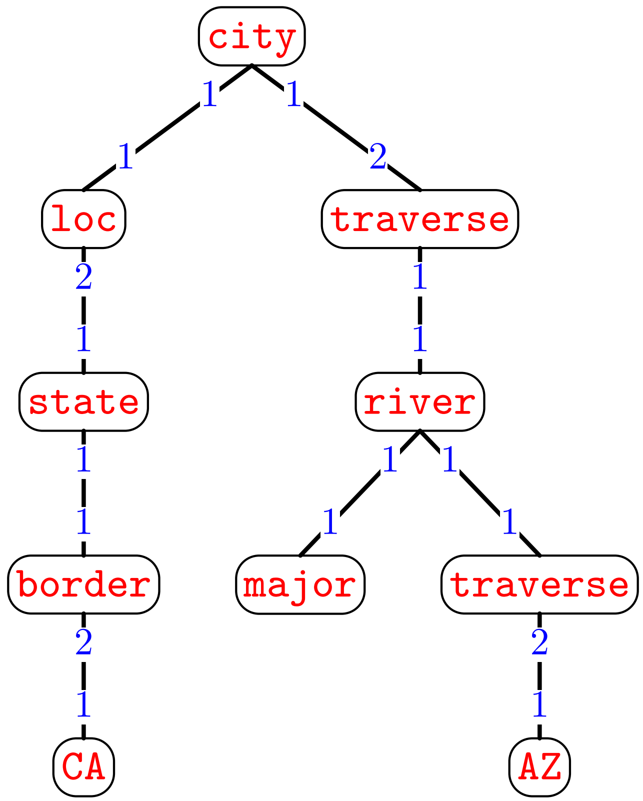
A DCS tree encodes a **constraint satisfaction problem** (CSP)

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

Properties of DCS Trees

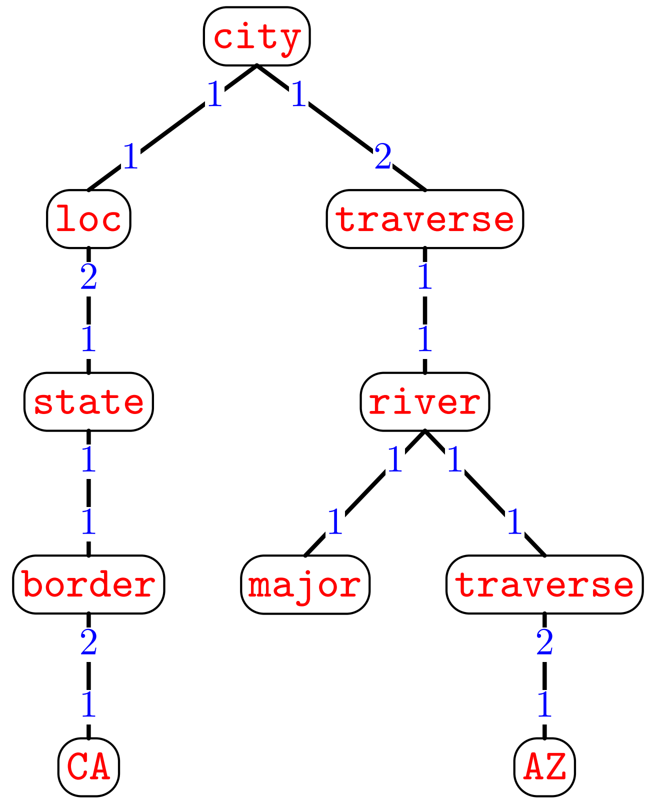


Properties of DCS Trees



Trees

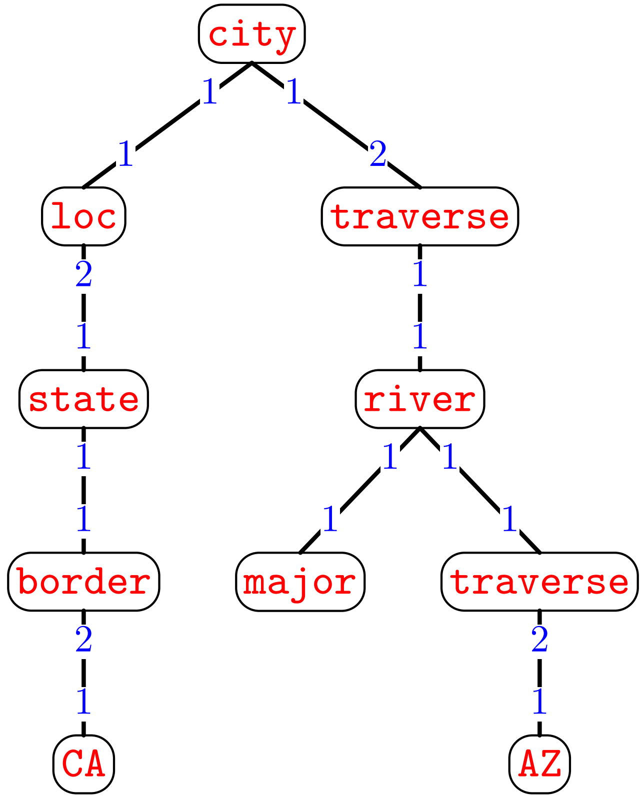
Properties of DCS Trees



Linguistics ············· Trees

syntactic locality

Properties of DCS Trees



Linguistics ·············· Trees ·············· Computation

syntactic locality efficient interpretation

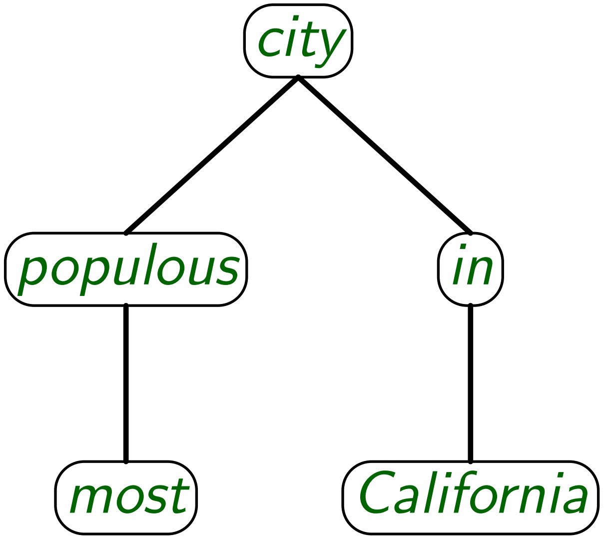
Divergence between Syntactic and Semantic Scope

most populous city in California

Divergence between Syntactic and Semantic Scope

most populous city in California

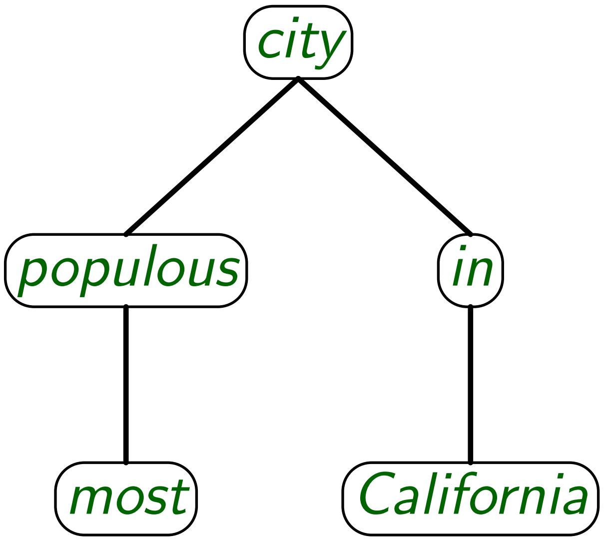
Syntax



Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



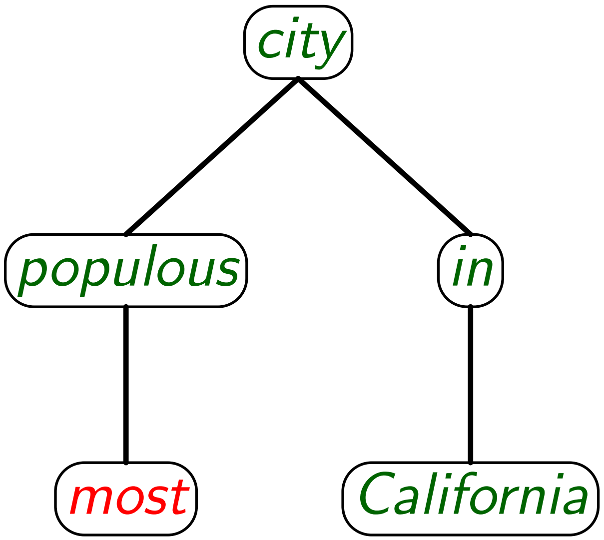
Semantics

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

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



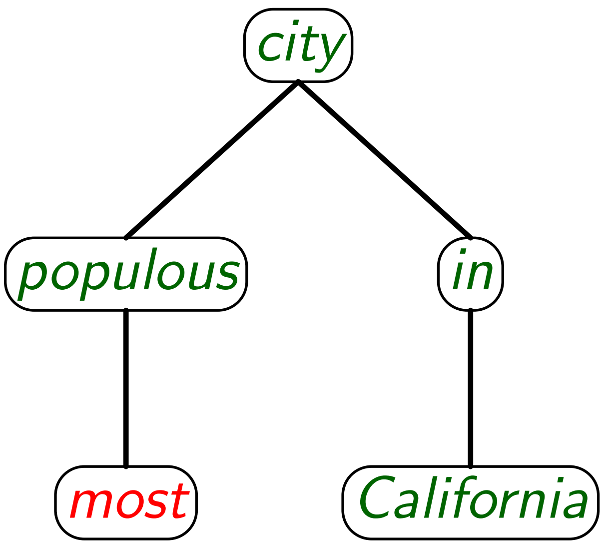
Semantics

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

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



Semantics

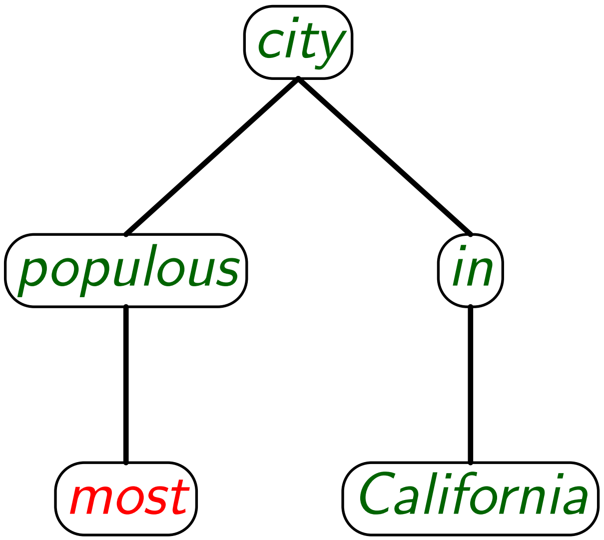
$$\texttt{argmax}(\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA}), \lambda x.\texttt{population}(x))$$

Problem: syntactic scope is lower than semantic scope

Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax



Semantics

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

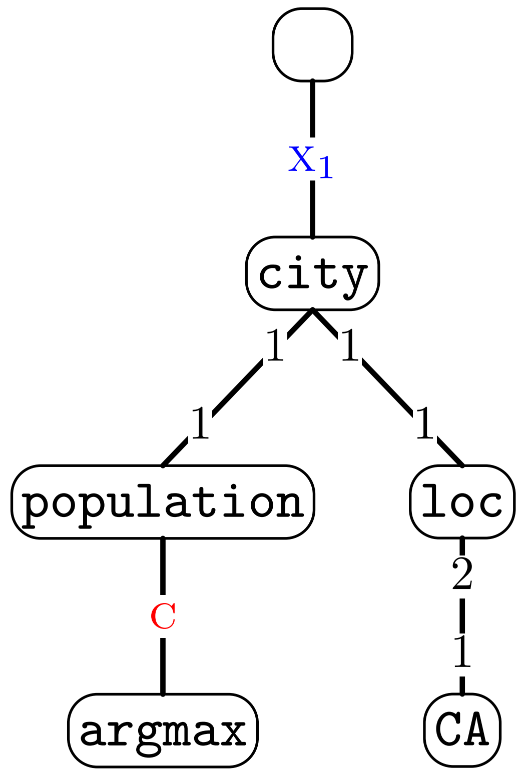
Problem: syntactic scope is lower than semantic scope

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

Solution: Mark-Execute

most populous city in California

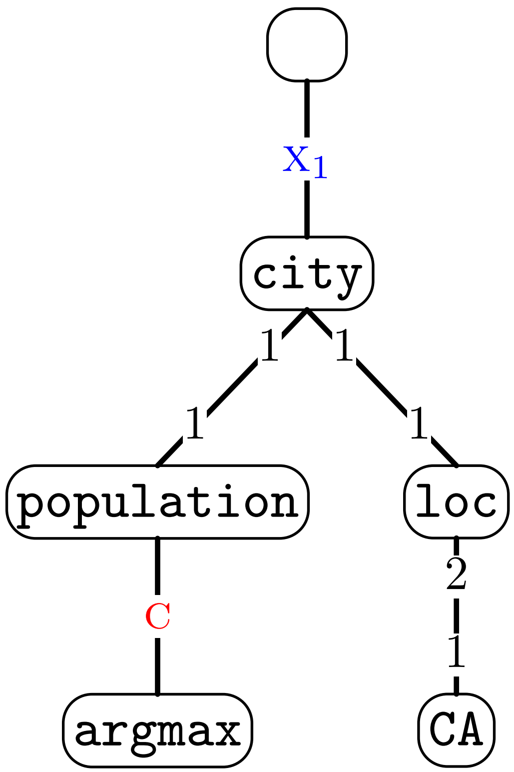
Superlatives



Solution: Mark-Execute

most populous city in California

Superlatives



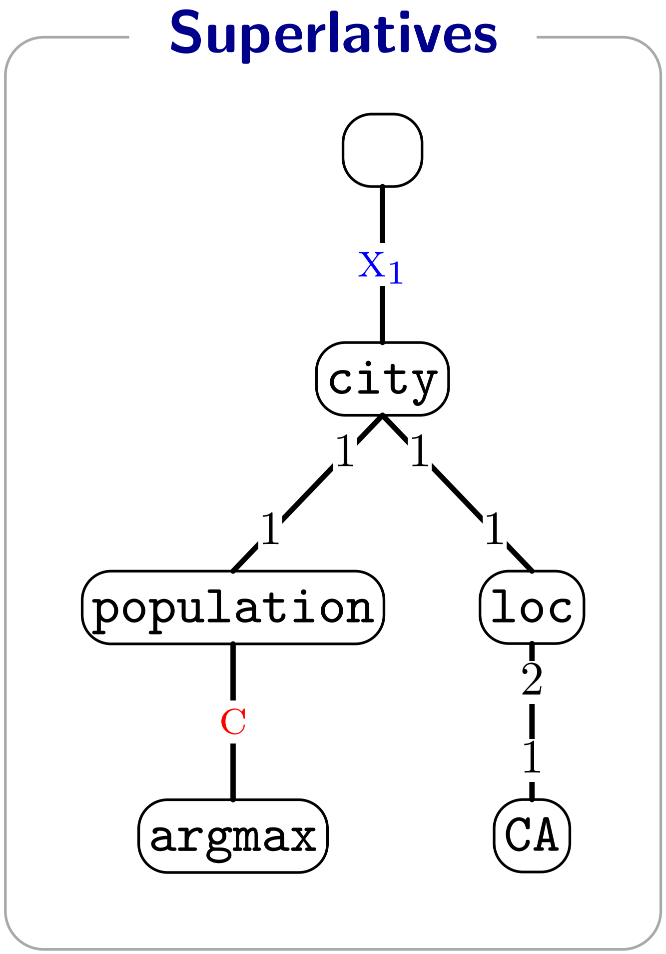
Mark at syntactic scope

Solution: Mark-Execute

most populous city in California

Execute at semantic scope

Mark at syntactic scope

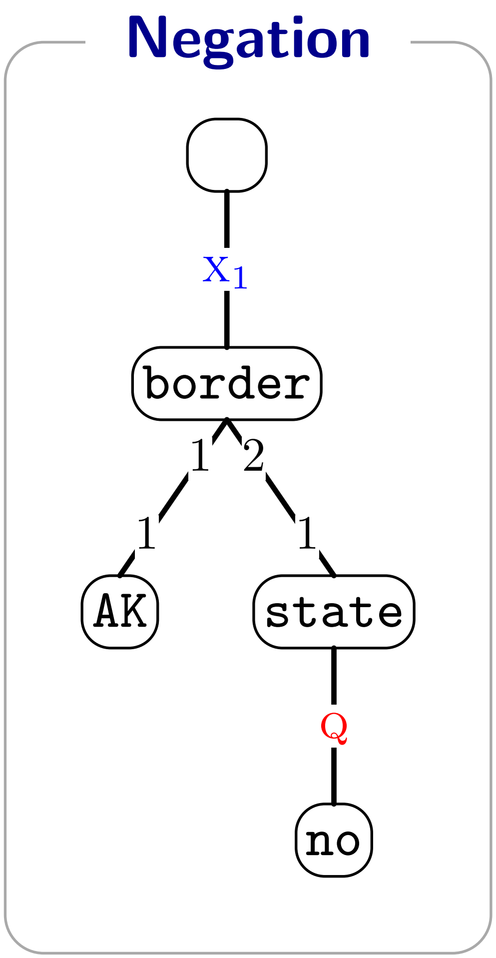


Solution: Mark-Execute

Alaska borders no states.

Execute at semantic scope

Mark at syntactic scope



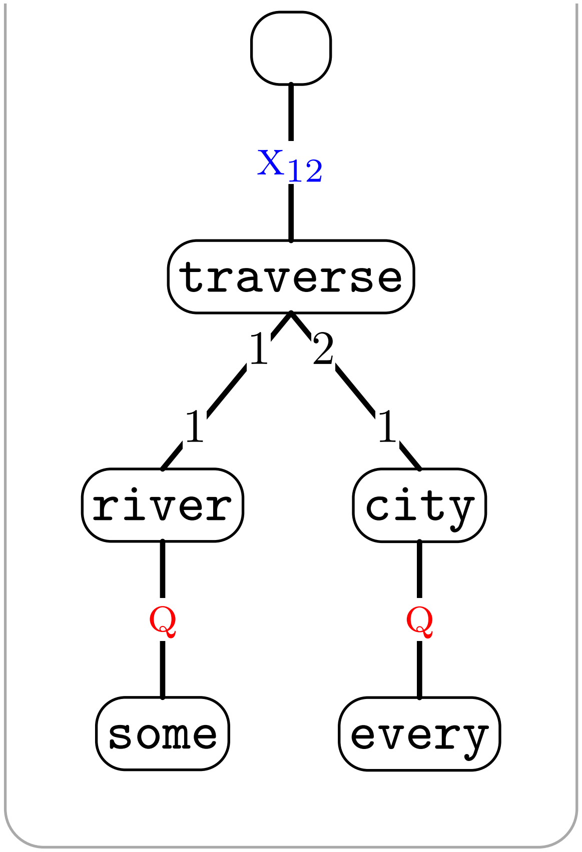
Solution: Mark-Execute

Some river traverses every city.

Quantification (narrow)

Execute at semantic scope

Mark at syntactic scope



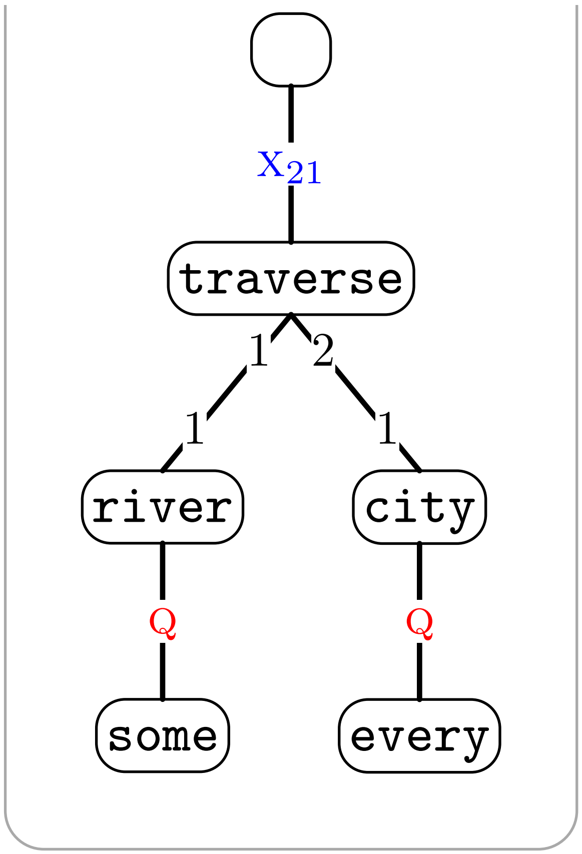
Solution: Mark-Execute

Some river traverses every city.

Quantification (wide)

Execute at semantic scope

Mark at syntactic scope



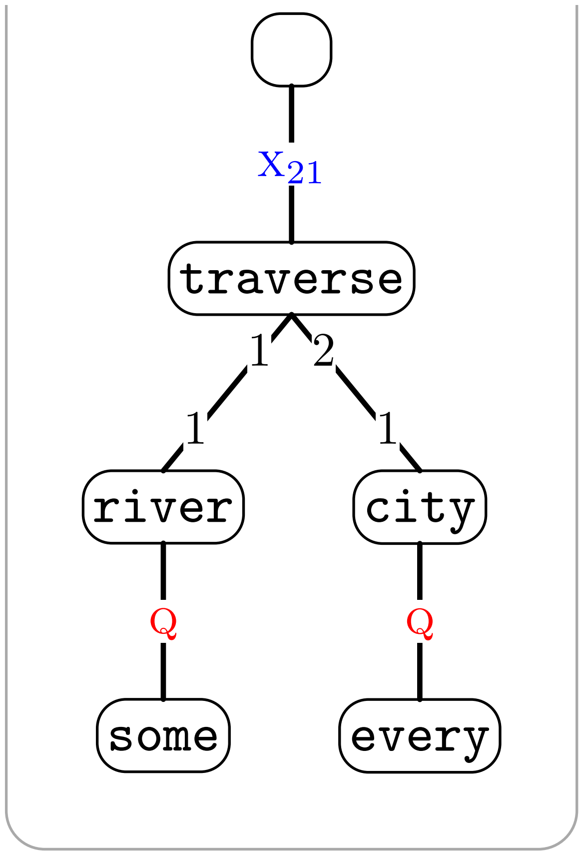
Solution: Mark-Execute

Some river traverses every city.

Quantification (wide)

Execute at semantic scope

Mark at syntactic scope



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

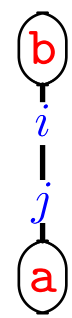
Lexicon (very simple/crude)

no $\Rightarrow$ `no`

state $\Rightarrow$ `state`

Grammar (very simple/crude)

`a` `b` $\Rightarrow$ 

`a` `b` $\Rightarrow$ 

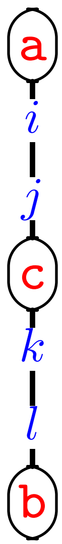
From Sentences to DCS Trees

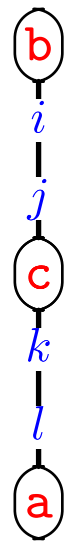
Lexicon (very simple/crude)

no ⇒ `no`

state ⇒ `state`

Grammar (very simple/crude)

`a` `b` ⇒ 

`a` `b` ⇒ 

Words to Predicates (Lexical Semantics)

What is the most populous city in CA ?

Words to Predicates (Lexical Semantics)

$\overset{\texttt{CA}}{}$

What is the most populous city in $\overset{\texttt{CA}}{\textit{CA}}$ *?*

Lexical Triggers:

1. String match $\quad$ *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:

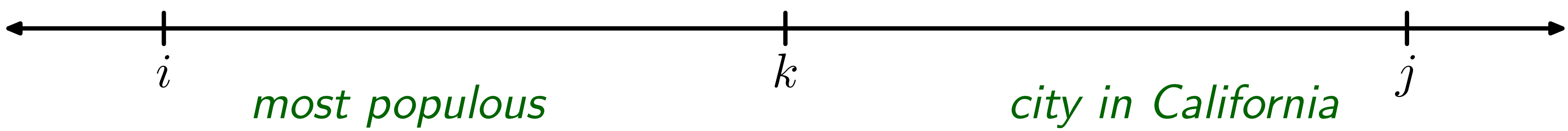
1. String match *CA* $\Rightarrow$ `CA`
2. Function words (20 words) *most* $\Rightarrow$ `argmax`
3. Nouns/adjectives *city* $\Rightarrow$ `city state river population`

Predicates to DCS Trees (Compositional Semantics)

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

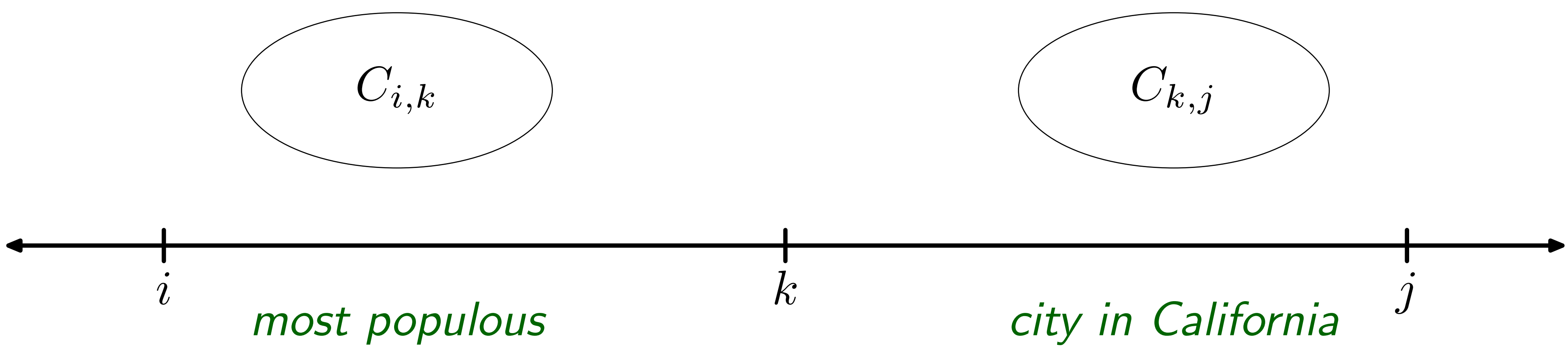
Predicates to DCS Trees (Compositional Semantics)

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



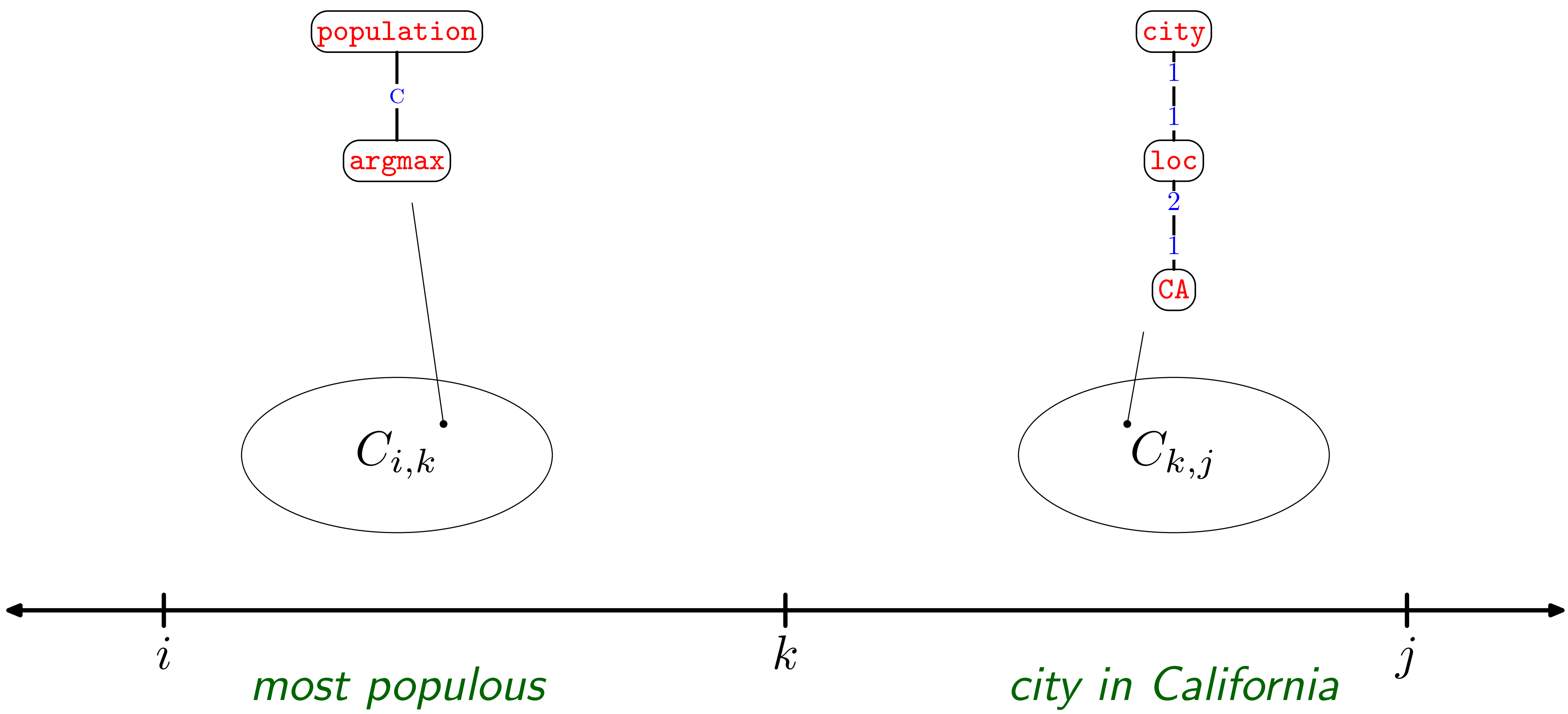
Predicates to DCS Trees (Compositional Semantics)

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



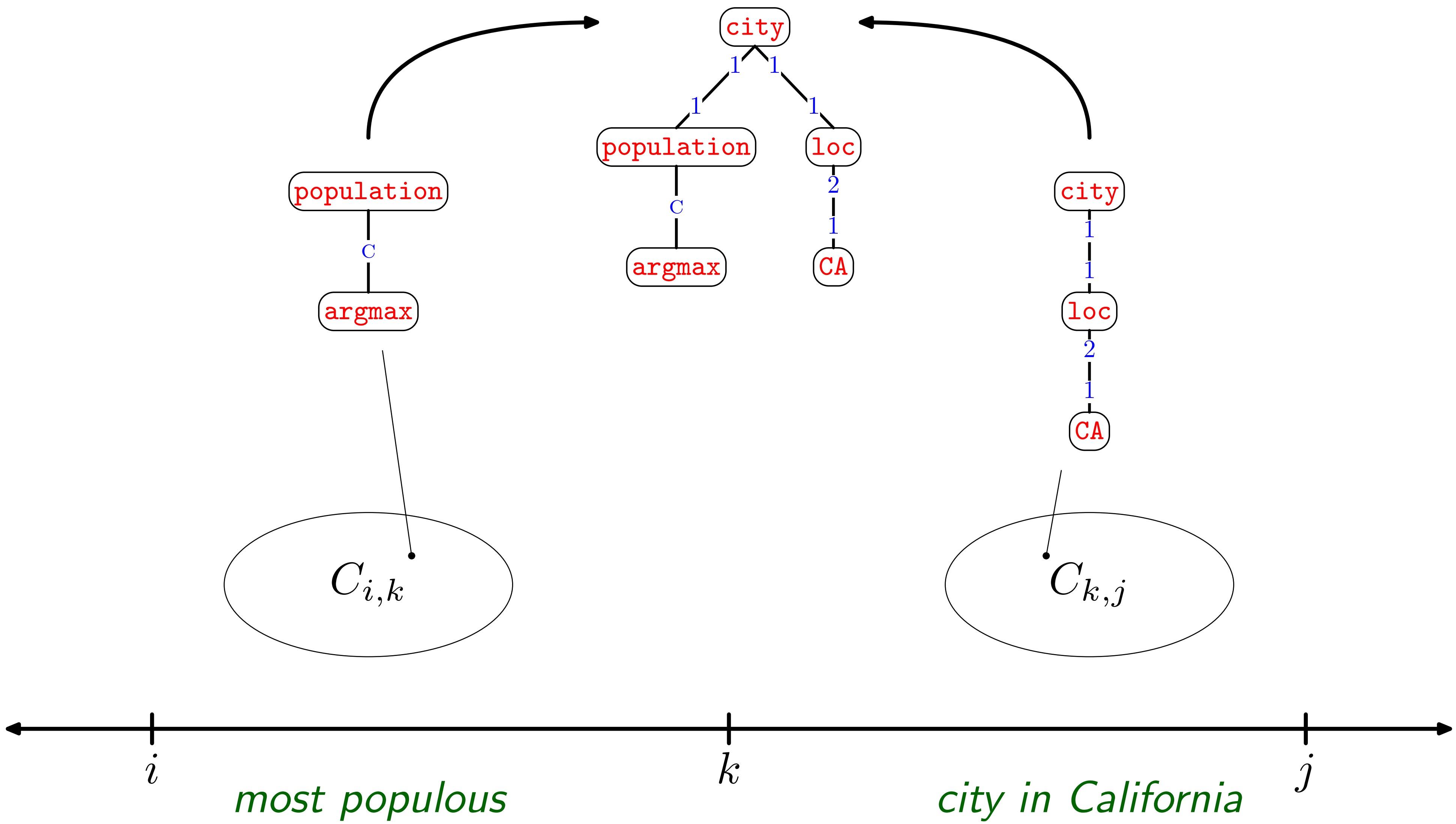
Predicates to DCS Trees (Compositional Semantics)

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



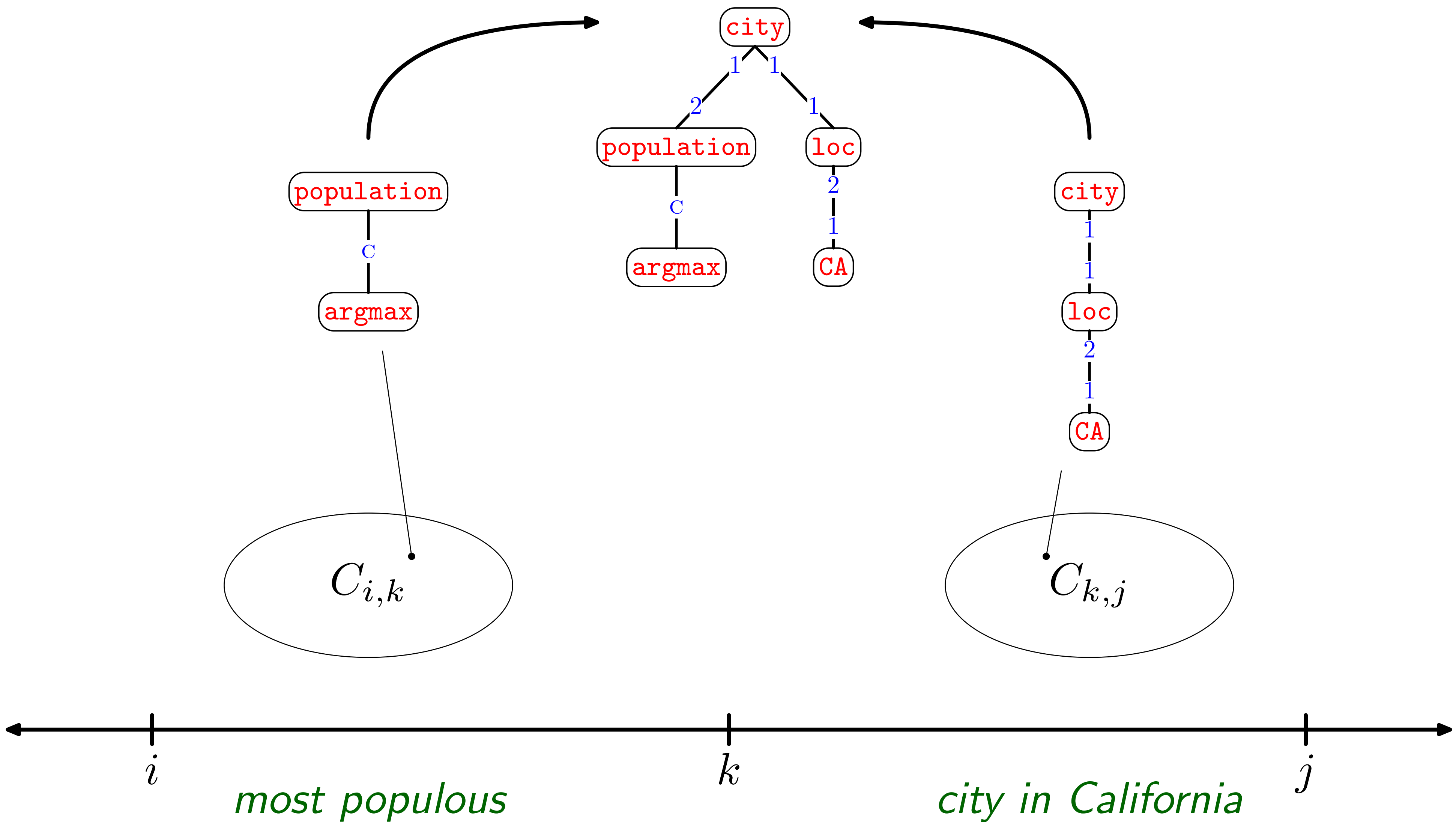
Predicates to DCS Trees (Compositional Semantics)

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



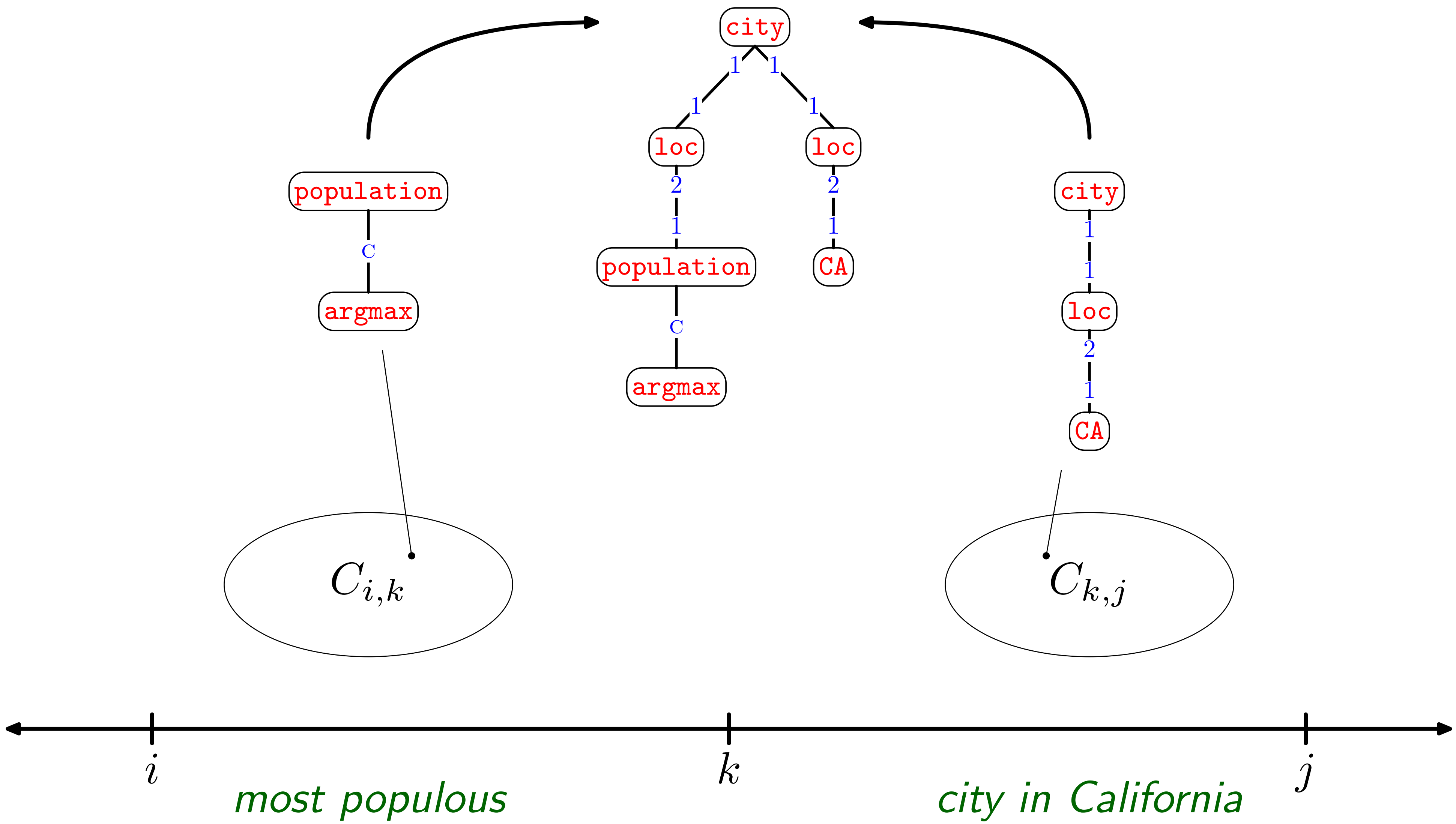
Predicates to DCS Trees (Compositional Semantics)

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



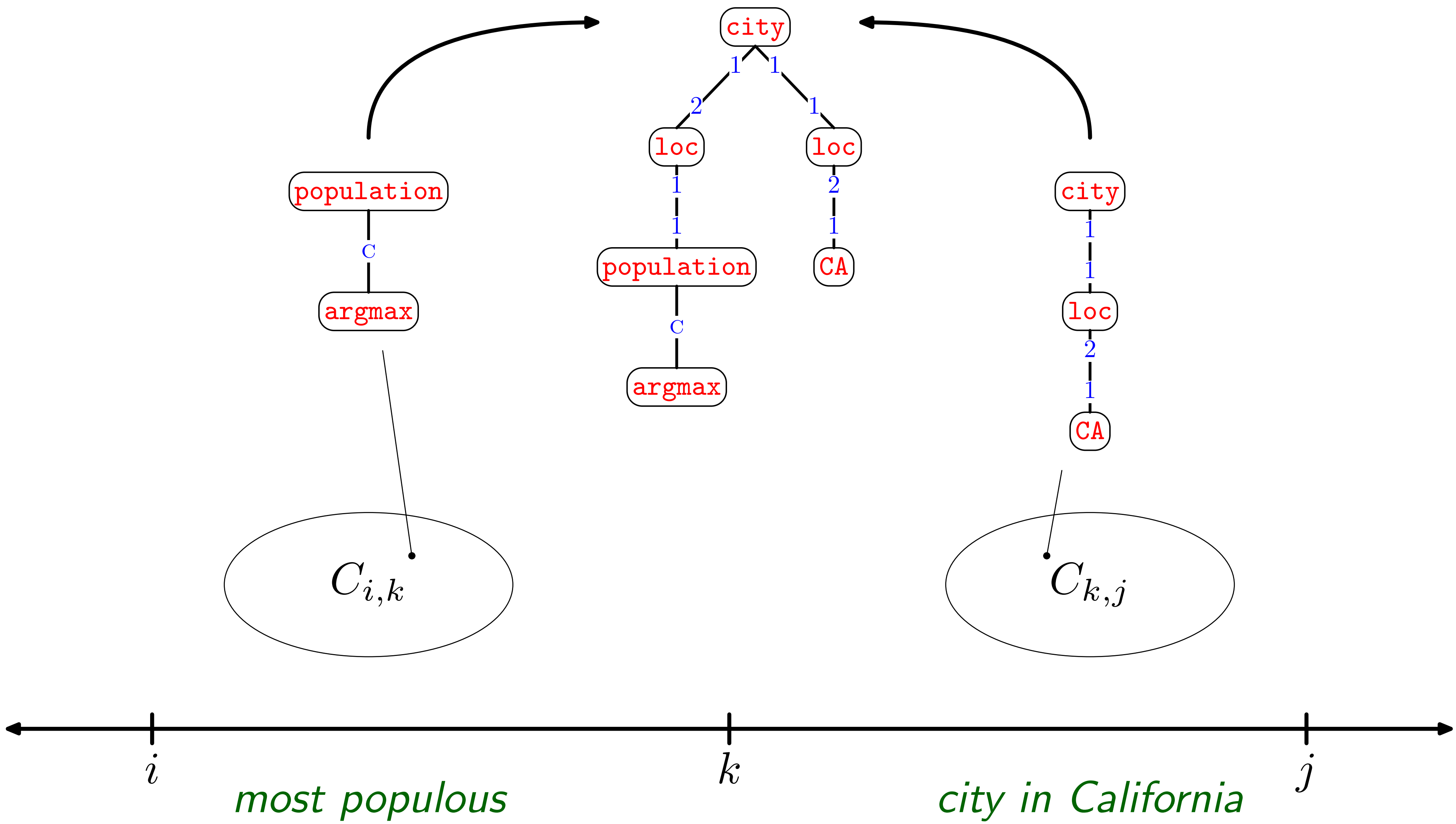
Predicates to DCS Trees (Compositional Semantics)

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



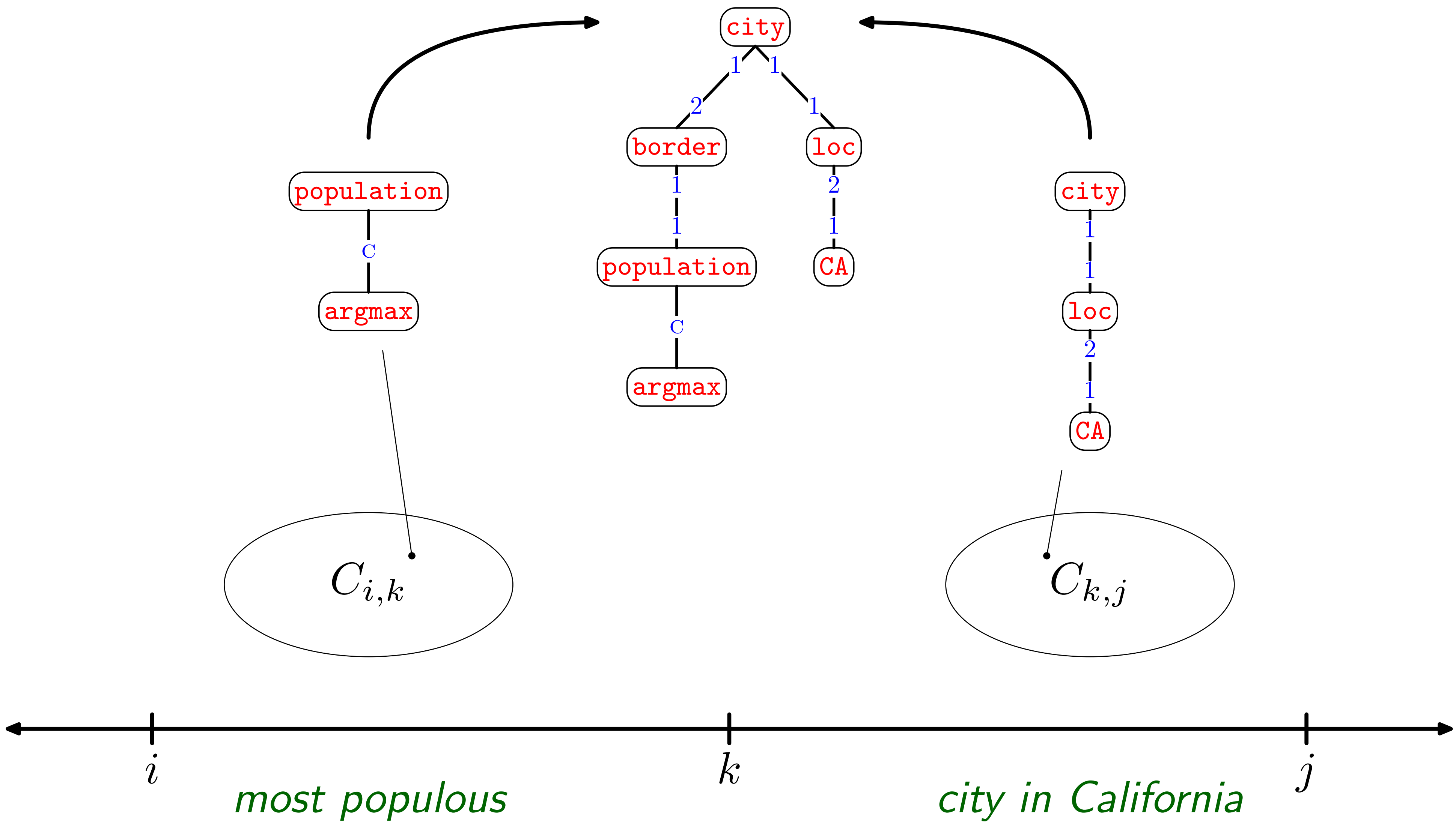
Predicates to DCS Trees (Compositional Semantics)

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



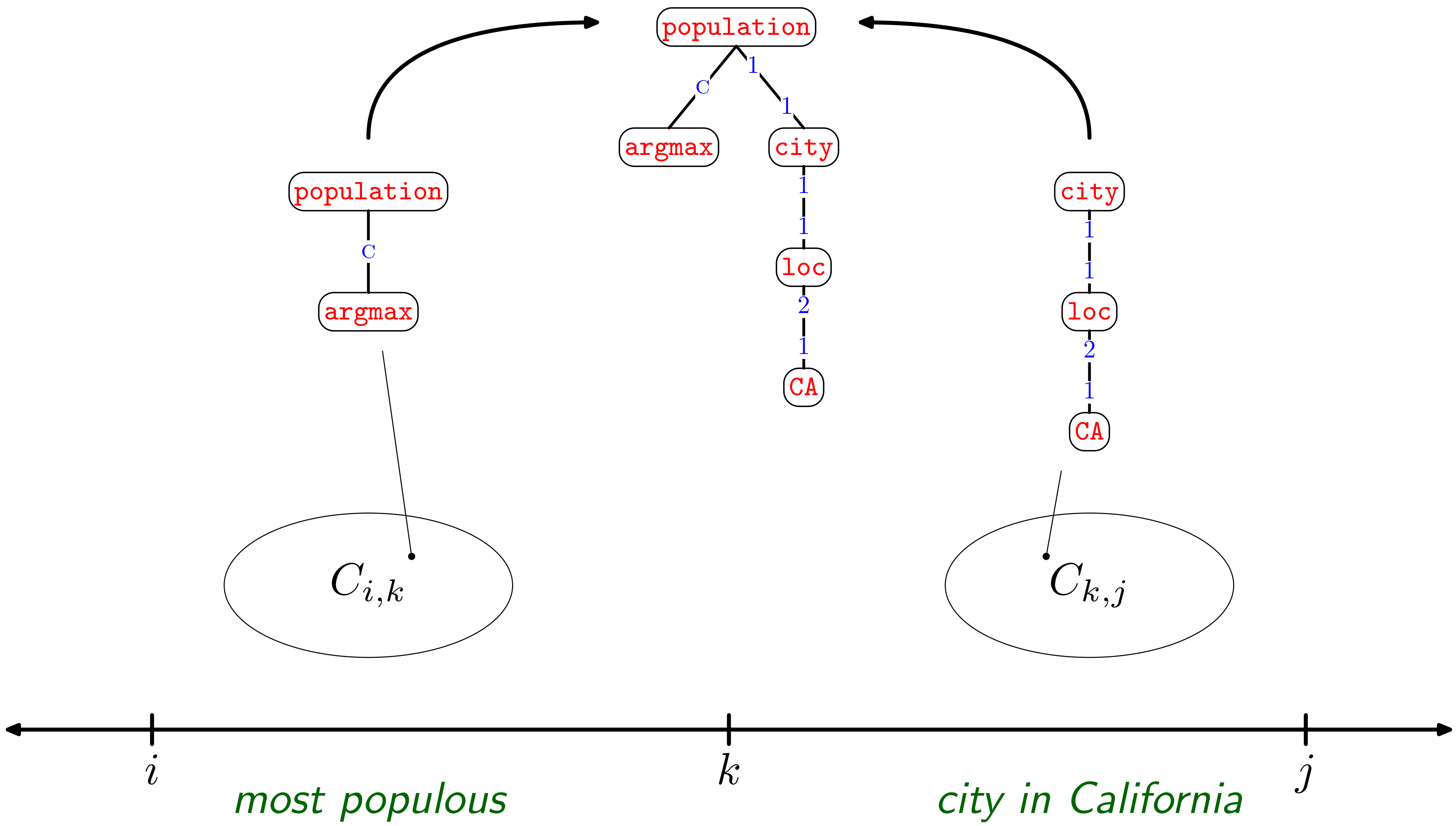
Predicates to DCS Trees (Compositional Semantics)

$C_{i,j}$ = 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
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	$\boxed{\texttt{city}}\text{-}1\text{—}1\text{-}\boxed{\texttt{loc}}\text{-}2\text{—}1\text{-}\boxed{\texttt{CA}}$

Comparison

	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	$\boxed{\texttt{city}}\overset{1}{-}\overset{1}{-}\boxed{\texttt{loc}}\overset{2}{-}\overset{1}{-}\boxed{\texttt{CA}}$
Lexicon	categories + lambda calculus	predicates
major	$\mathrm{N}/\mathrm{N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	$\texttt{major}$

Comparison

	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\mathtt{city}(x) \wedge \mathtt{loc}(x, \mathtt{CA})$	$\boxed{\mathtt{city}}\!-\!1\!-\!1\!-\!\boxed{\mathtt{loc}}\!-\!2\!-\!1\!-\!\boxed{\mathtt{CA}}$
Lexicon	categories + lambda calculus	predicates
major	$\mathrm{N/N} : \lambda f.\lambda x.f(x) \wedge \mathtt{major}(x)$	major
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \mathtt{a} \quad X : \mathtt{b} \quad \Rightarrow \quad Y : \mathtt{a}(\mathtt{b})$	$\boxed{\mathtt{a}}\!-\!i\!-\!j\!-\!\boxed{\mathtt{b}}$

Comparison

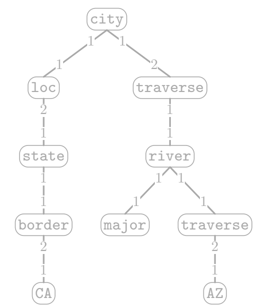
	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	$\texttt{city} - 1 — 1 - \texttt{loc} - 2 — 1 - \texttt{CA}$
Lexicon	categories + lambda calculus	predicates
major	$\mathrm{N/N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	$\texttt{major}$
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \texttt{a} \quad X : \texttt{b} \quad \Rightarrow \quad Y : \texttt{a}(\texttt{b})$	$\texttt{a} - i — j - \texttt{b}$
Nature	tighter control	simple/permissive

Comparison

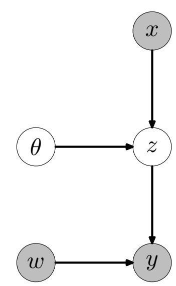
	CCG	**DCS**
Logical form	lambda calculus formulae	DCS trees
	$\lambda x.\texttt{city}(x) \wedge \texttt{loc}(x, \texttt{CA})$	$\boxed{\texttt{city}} \overset{1}{-}\overset{1}{-} \boxed{\texttt{loc}} \overset{2}{-}\overset{1}{-} \boxed{\texttt{CA}}$
Lexicon	categories + lambda calculus	predicates
major	$\mathrm{N/N} : \lambda f.\lambda x.f(x) \wedge \texttt{major}(x)$	`major`
Grammar	combinator rules	$\approx$ dependency parsing
	$Y/X : \texttt{a} \quad X : \texttt{b} \quad \Rightarrow \quad Y : \texttt{a}(\texttt{b})$	$\texttt{a} \overset{i}{-}\overset{j}{-} \texttt{b}$
Nature	tighter control	simple/permissive
Origin	linguistics	NLP

Outline

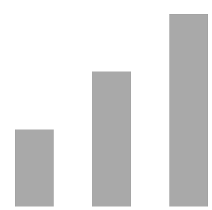
Representation



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

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

Supervision

Detailed Supervision

- doesn't scale up

What is the largest city in California?

$\Downarrow$ **expert**

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

Natural Supervision

What is the largest city in California?

$\Downarrow$

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?

$\Downarrow$ **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?

$\Downarrow$ **non-expert**

Los Angeles

Supervision

Detailed Supervision

- doesn't scale up
- representation-dependent

What is the largest city in California?

$\Downarrow$ **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?

$\Downarrow$ **non-expert**

Los Angeles

Supervision

Detailed Supervision

- doesn't scale up
- representation-dependent

What is the largest city in California?

$\Big\downarrow$ **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?

$\Big\downarrow$ **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 $\boxed{\text{LF}}$ LF LF LF LF LF LF LF LF LF LF $\boxed{\text{LF}}$ $\cdots$

Los Angeles

Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

$\cdots$ LF LF LF LF LF $\boxed{\text{LF}}$ LF LF LF LF LF LF LF LF LF LF $\boxed{\text{LF}}$ $\cdots$

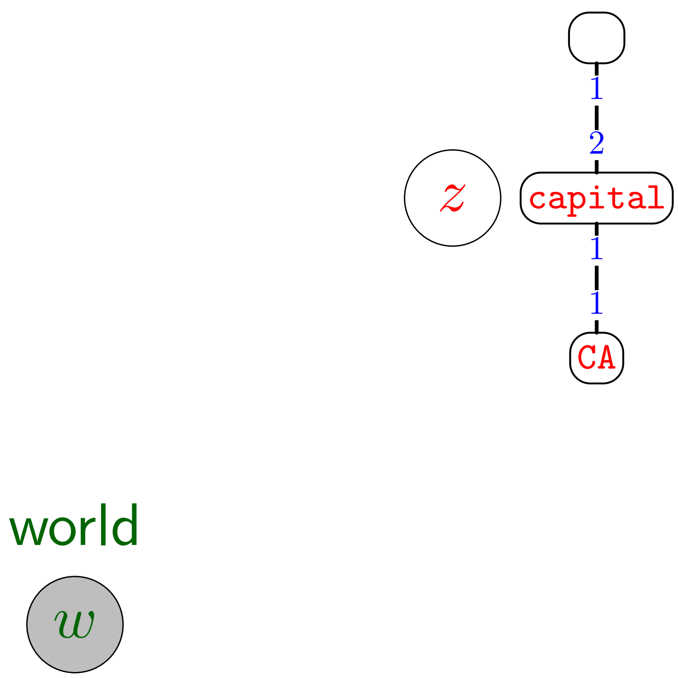
Los Angeles

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

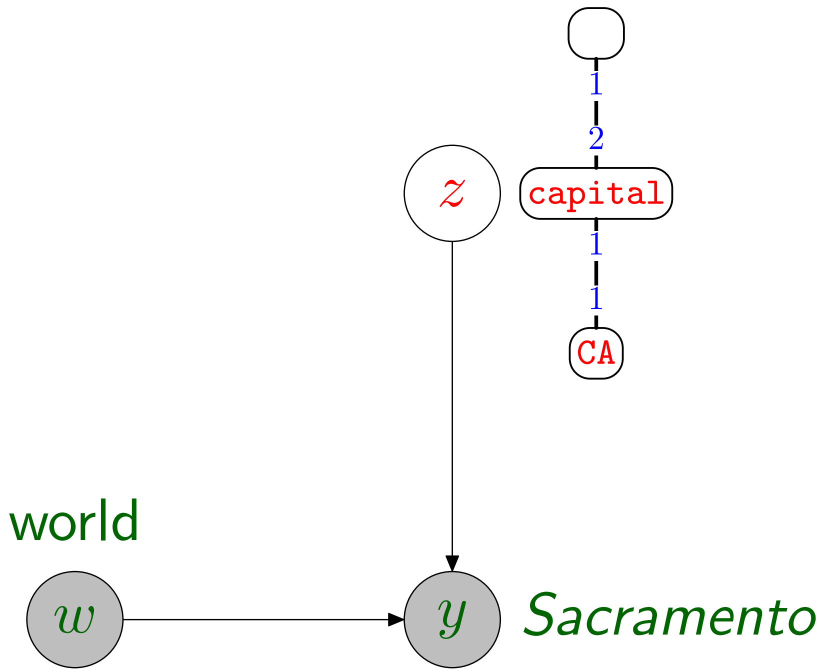
What is the most populous city in California?

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

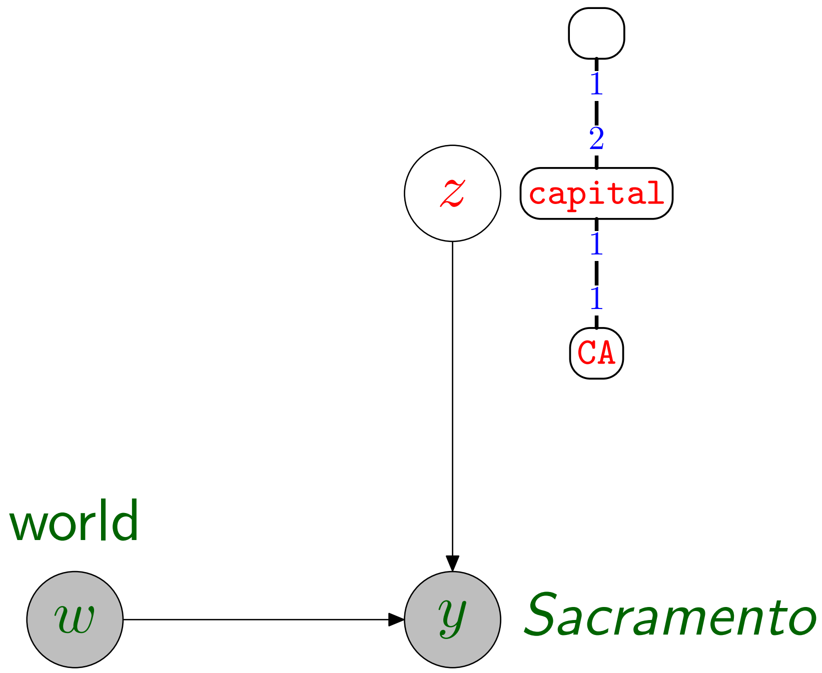
Graphical Model



Graphical Model



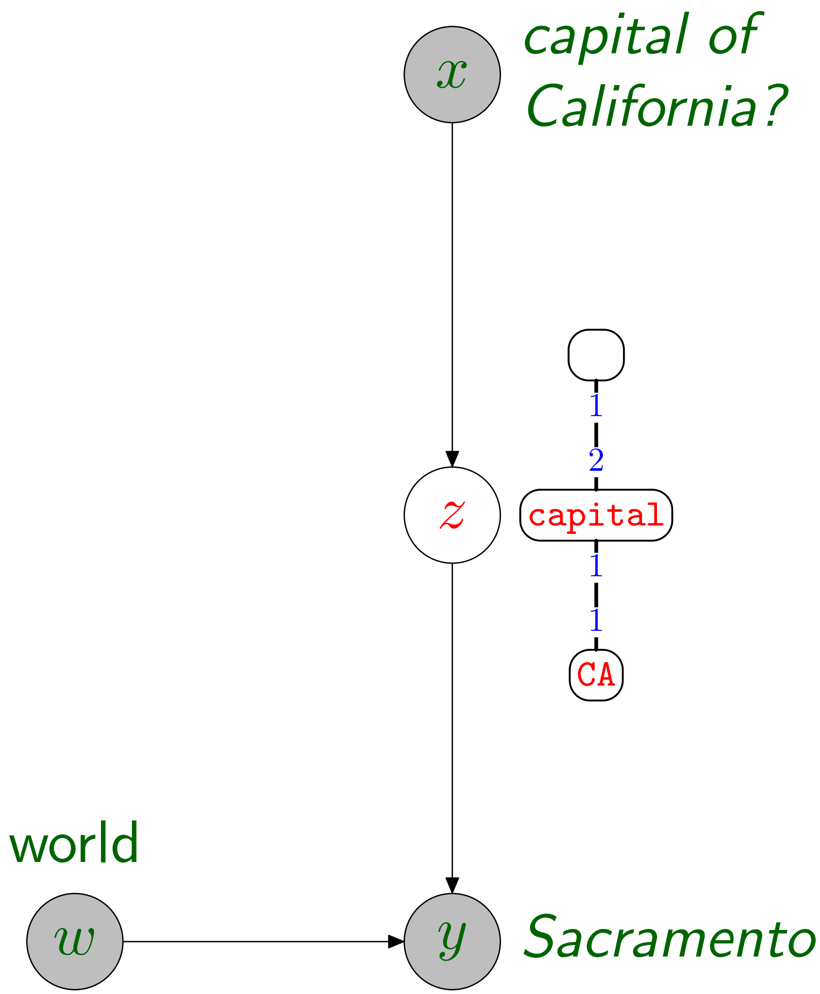
Graphical Model



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

(deterministic)

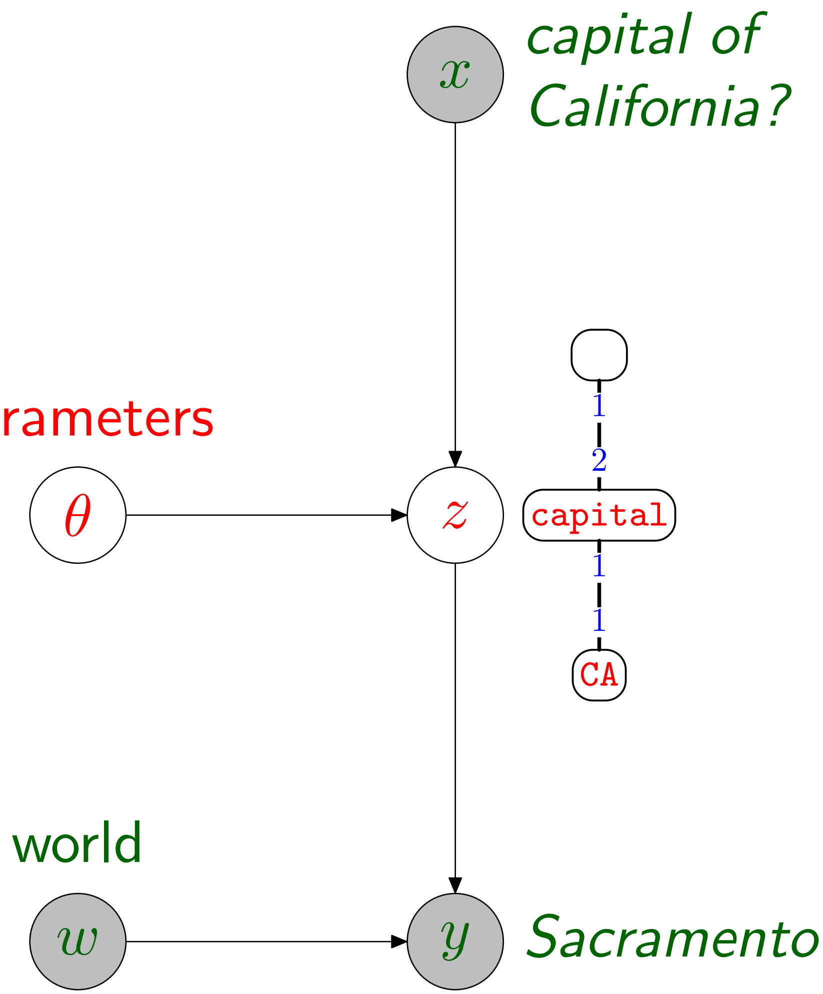
Graphical Model



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

(deterministic)

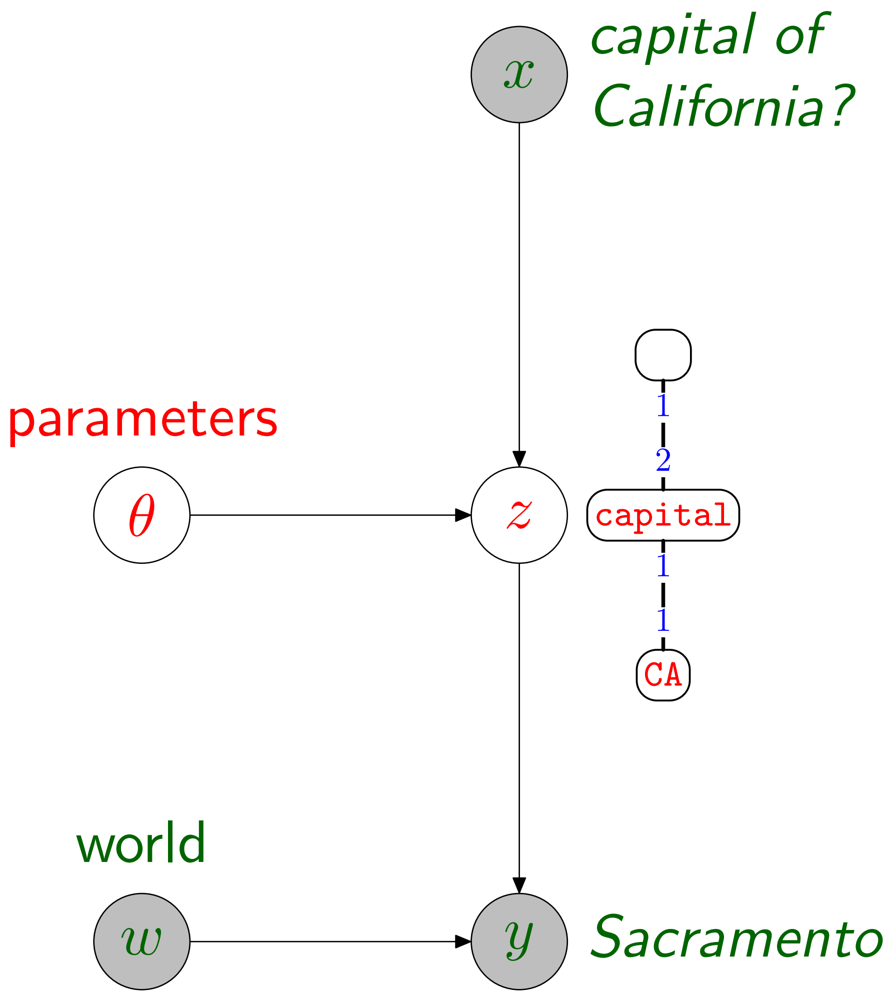
Graphical Model



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

(deterministic)

Graphical Model



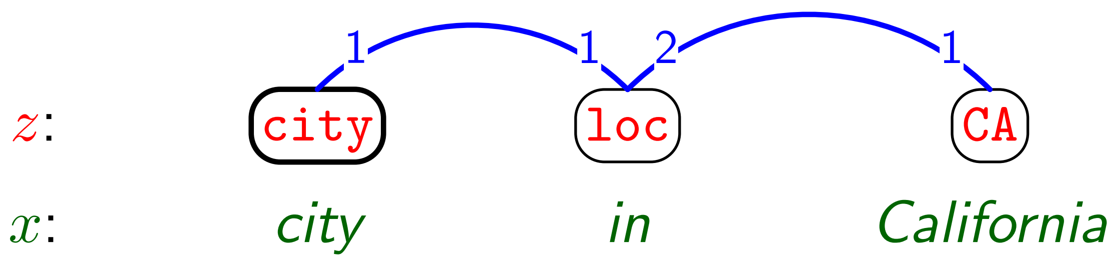
Semantic Parsing: $p(z \mid x, \theta)$

(probabilistic)

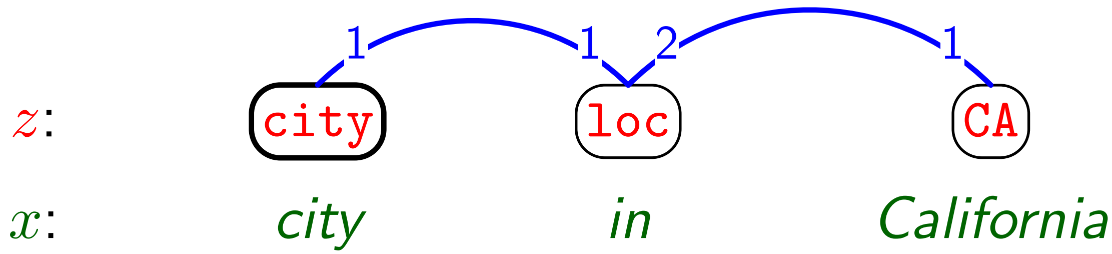
Interpretation: $p(y \mid z, w)$

(deterministic)

Semantic Parsing Log-linear Model

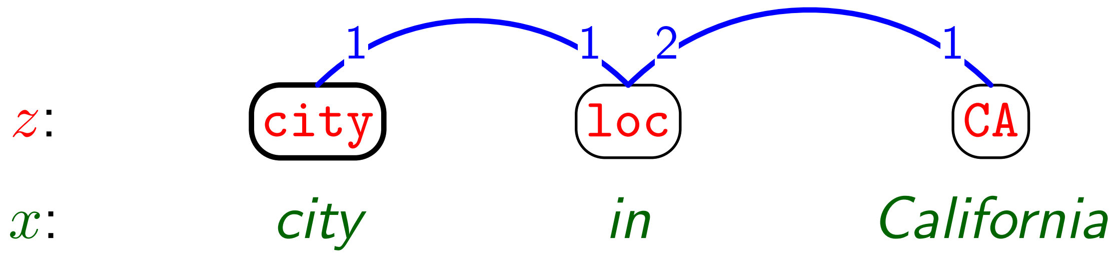


Semantic Parsing Log-linear Model



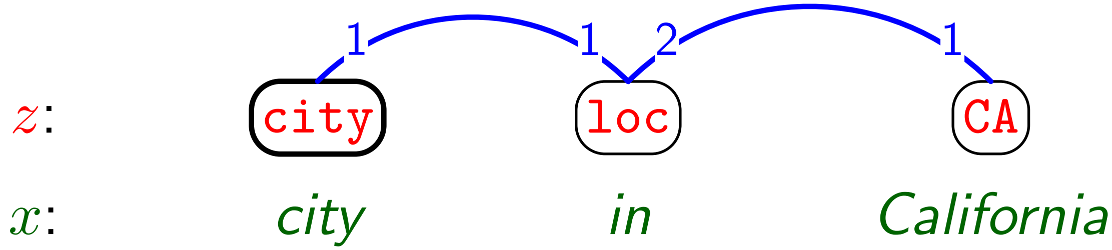
$$\text{features}(x, z) = \begin{pmatrix} \\ \\ \\ \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



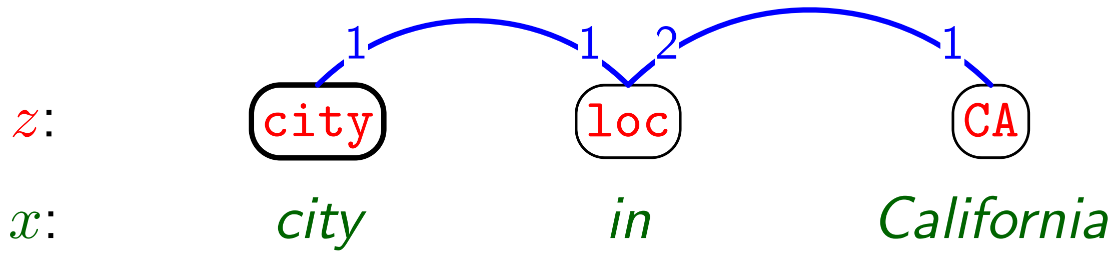
$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots\cdot \texttt{loc} \quad : 1 \\ \\ \\ \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



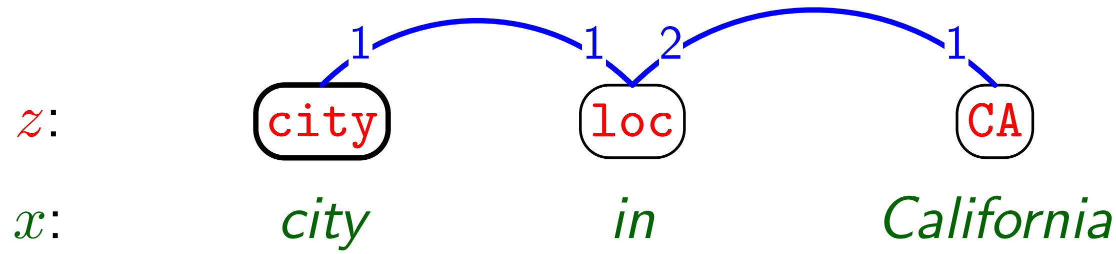
$$\text{features}(x, z) = \begin{pmatrix} \textit{in} \cdots\cdots \texttt{loc} & : 1 \\ \boxed{\texttt{city}}\text{-}1\text{—}1\text{-}\boxed{\texttt{loc}} & : 1 \\ & \end{pmatrix} \in \mathbb{R}^d$$

Semantic Parsing Log-linear Model



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

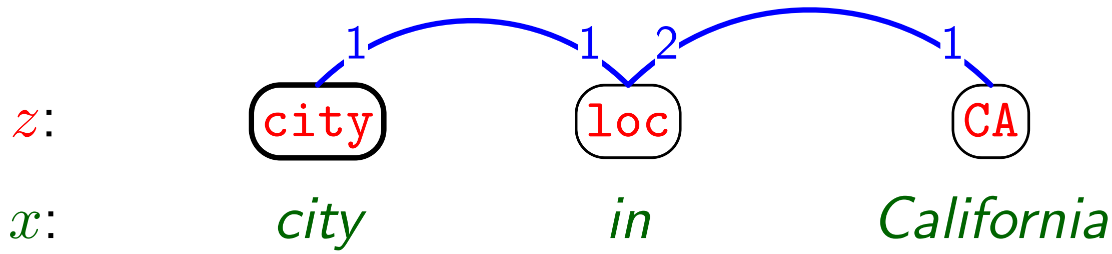
Semantic Parsing Log-linear Model



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

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

Semantic Parsing Log-linear Model



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

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

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

Learning

Objective Function:

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

Interpretation **Semantic parsing**

Learning

Objective Function:

$$\max_{\theta} \quad \underbrace{p(y \mid z, w)}_{\textbf{Interpretation}} \, \underbrace{p(z \mid x, \theta)}_{\textbf{Semantic parsing}}$$

Learning

Objective Function:

$$\max_{\theta} \sum_{z} \underbrace{p(y \mid z, w)}_{\textbf{Interpretation}} \, \underbrace{p(z \mid x, \theta)}_{\textbf{Semantic parsing}}$$

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

$(0, 0, \ldots, 0)$ $\xrightarrow{\text{enumerate/score DCS trees}}$

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

$(0, 0, \ldots, 0)$ $\xrightarrow{\text{enumerate/score DCS trees}}$

k-best list

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

Learning

Objective Function:

$$\max_{\theta} \sum_{z} \underbrace{p(y \mid z, w)}_{\textbf{Interpretation}} \underbrace{p(z \mid x, \theta)}_{\textbf{Semantic parsing}}$$

EM-like Algorithm:

parameters θ

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

enumerate/score DCS trees $\longrightarrow$

$\longleftarrow$ 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

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

$\xrightarrow{\text{enumerate/score DCS trees}}$

$\xleftarrow[\text{numerical optimization (L-BFGS)}]{}$

k-best list

`tree3` ✓

`tree8` ✓

`tree6` ✗

`tree2` ✗

`tree4` ✗

Learning

Objective Function:

$$\max_{\theta} \sum_{z} \underbrace{p(y \mid z, w)}_{\textbf{Interpretation}} \underbrace{p(z \mid x, \theta)}_{\textbf{Semantic parsing}}$$

EM-like Algorithm:

parameters θ

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

$\xrightarrow{\text{enumerate/score DCS trees}}$

$\xleftarrow[\text{numerical optimization (L-BFGS)}]{}$

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 →

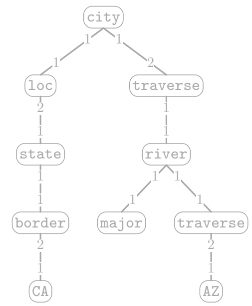
← numerical optimization (L-BFGS)

k-best list

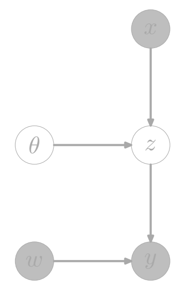
`tree3` ✓
`tree8` ✓
`tree2` ✗
`tree4` ✗
`tree9` ✗

Outline

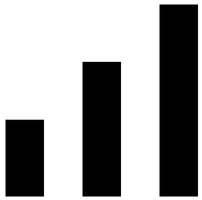
Representation



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?
$\Rightarrow$ *Walton County*

How many states have a city called Rochester?
$\Rightarrow$ *2*

What is the longest river that runs through a state that borders Tennessee?
$\Rightarrow$ *Missouri*

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

Input to Learning Algorithm

Training data (600 examples)

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

Lexicon (20 general, 22 specific)

`no`	$\Rightarrow$	*no*
`argmax`	$\Rightarrow$	*most*
`city`	$\Rightarrow$	*city*
`state`	$\Rightarrow$	*state*
`mountain`	$\Rightarrow$	*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
...	...

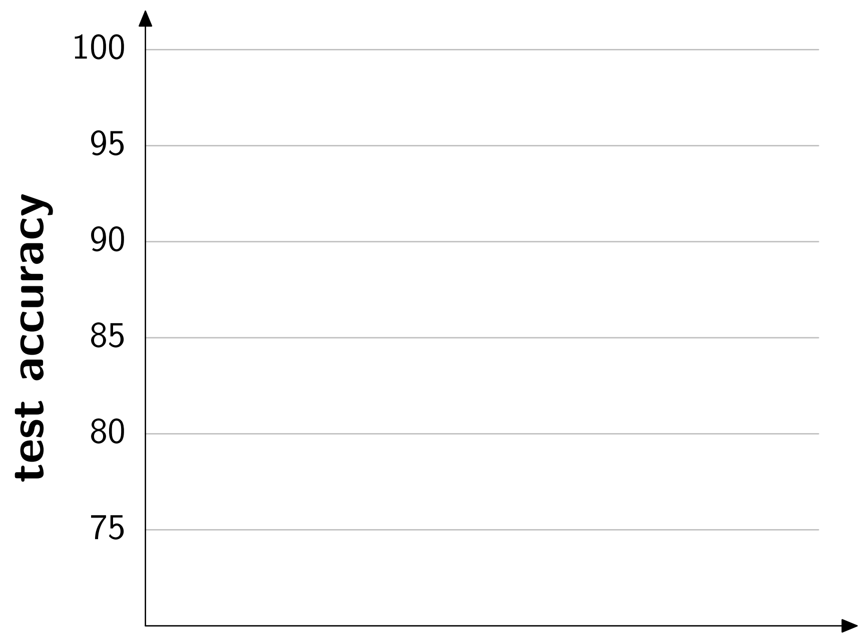
`border`

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

... ...

Experiment 1

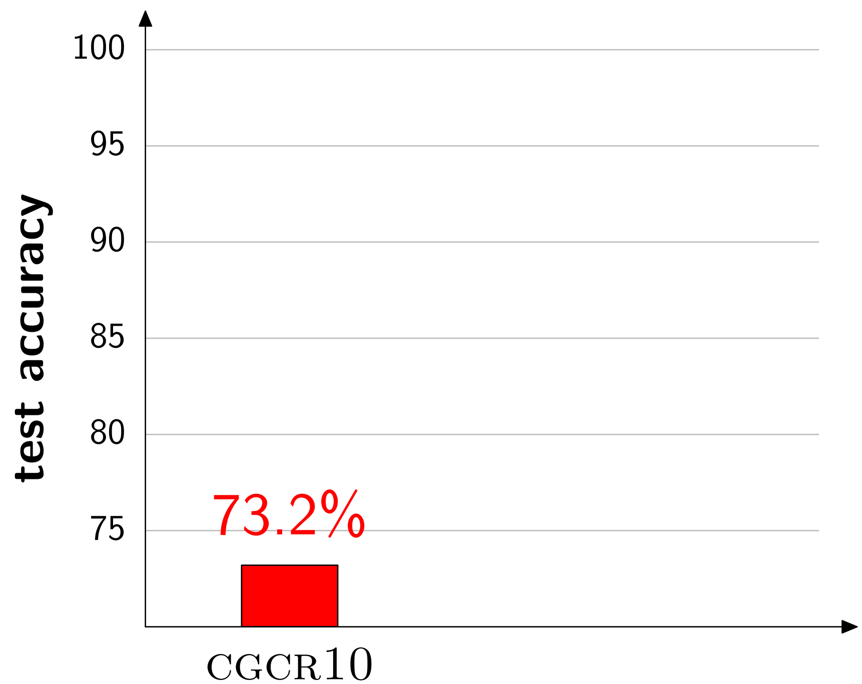
On GEO, 250 training examples, 250 test examples



Experiment 1

On GEO, 250 training examples, 250 test examples

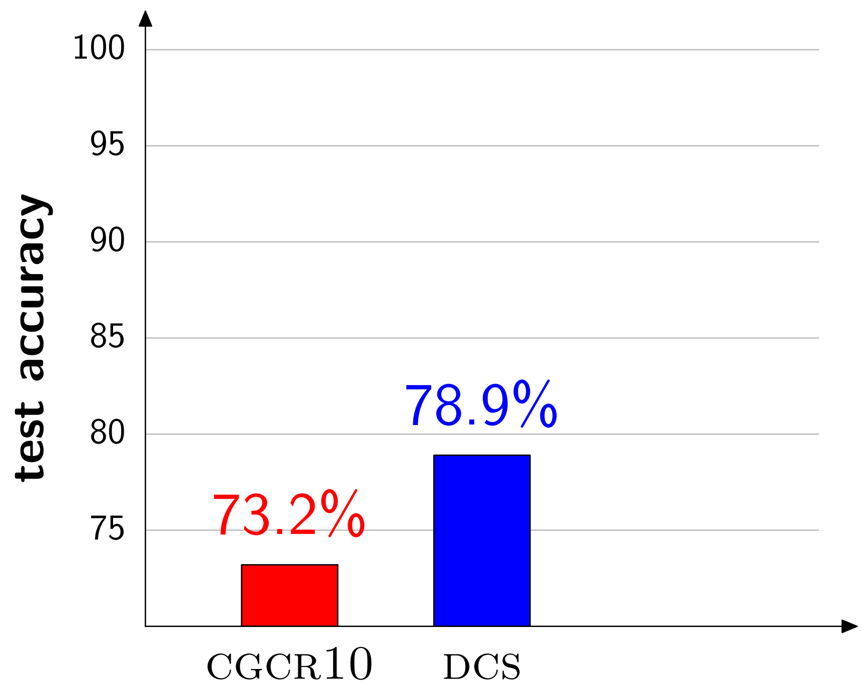
System	Description	Lexicon (gen./spec.)	Logical forms
CGCR10	FunQL [Clarke et al., 2010]	✔ ✔	✗



Experiment 1

On Geo, 250 training examples, 250 test examples

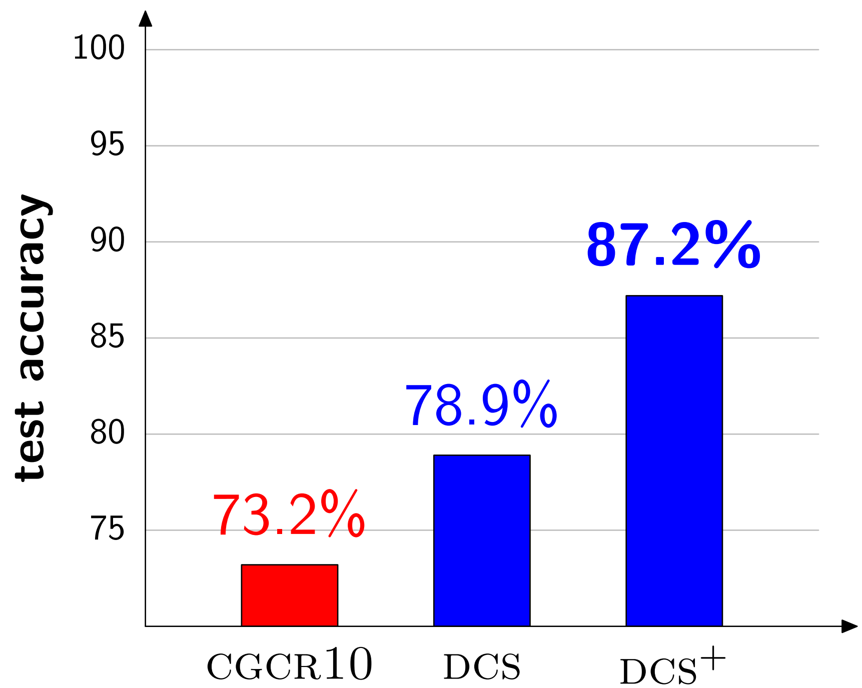
System	Description	Lexicon (gen./spec.)	Logical forms
CGCR10	FunQL [Clarke et al., 2010]	✔ ✔	✗
LJK11	DCS [Liang et al., 2011]	✔ ✗	✗



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

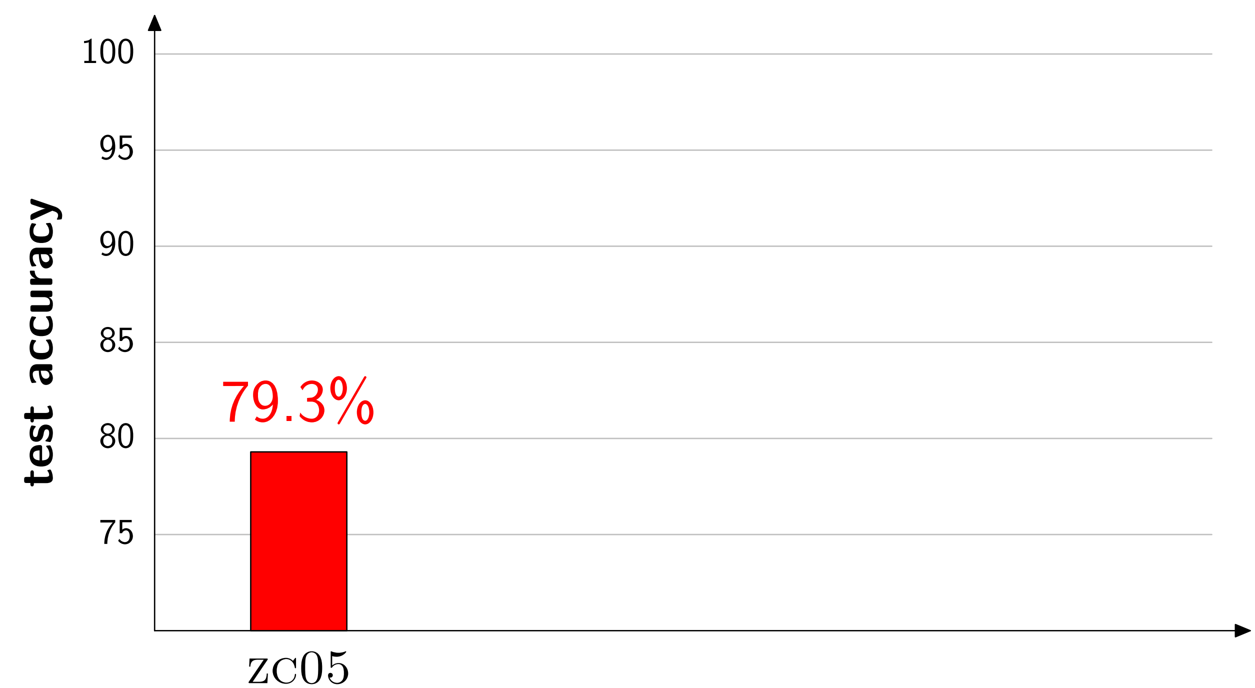
On GEO, 600 training examples, 280 test examples

System	Description	Lexicon	Logical forms

Experiment 2

On GEO, 600 training examples, 280 test examples

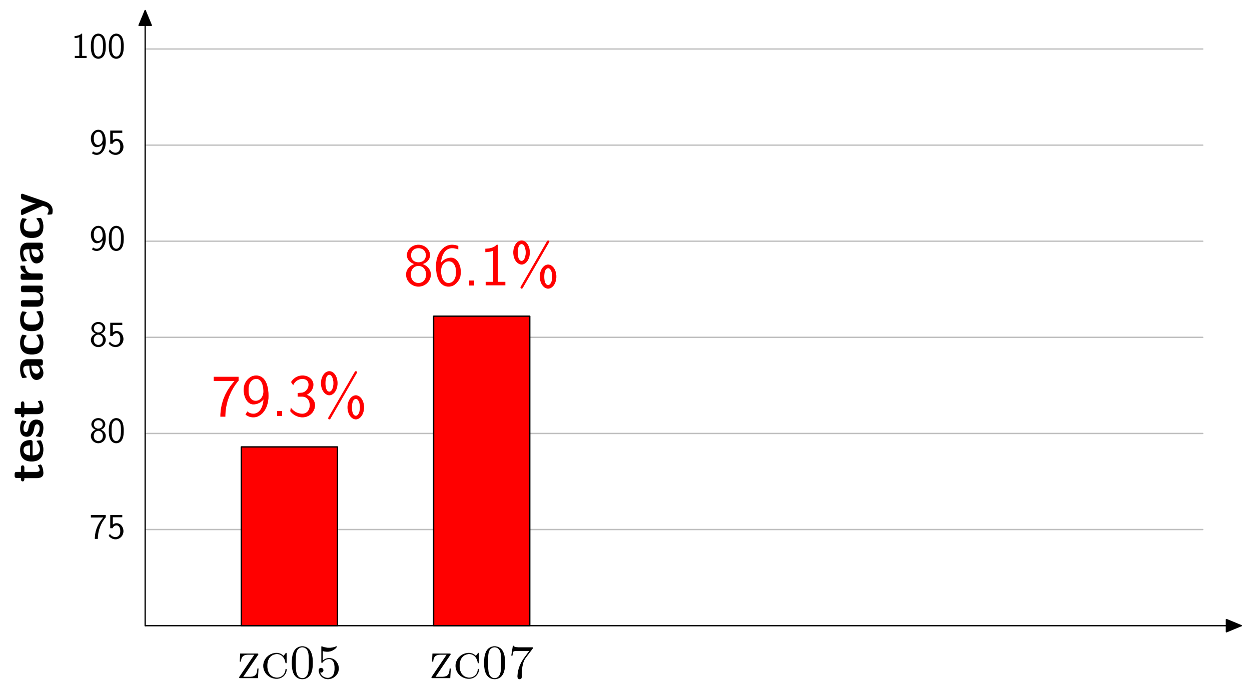
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

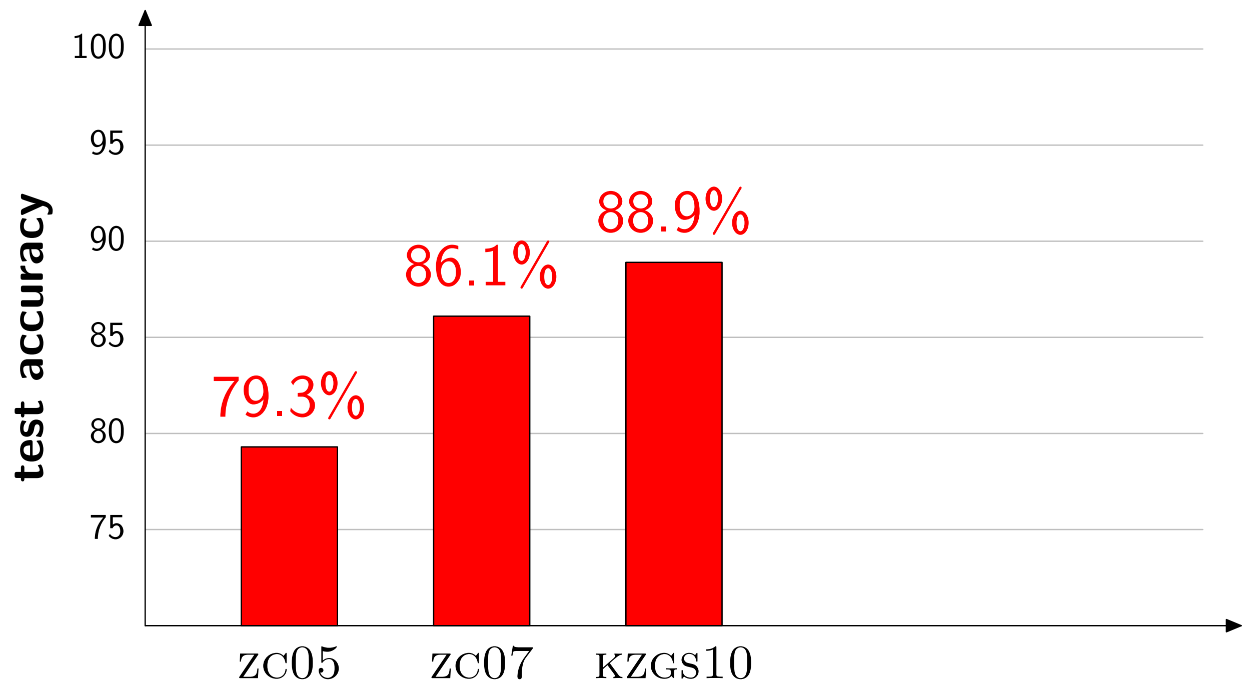
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

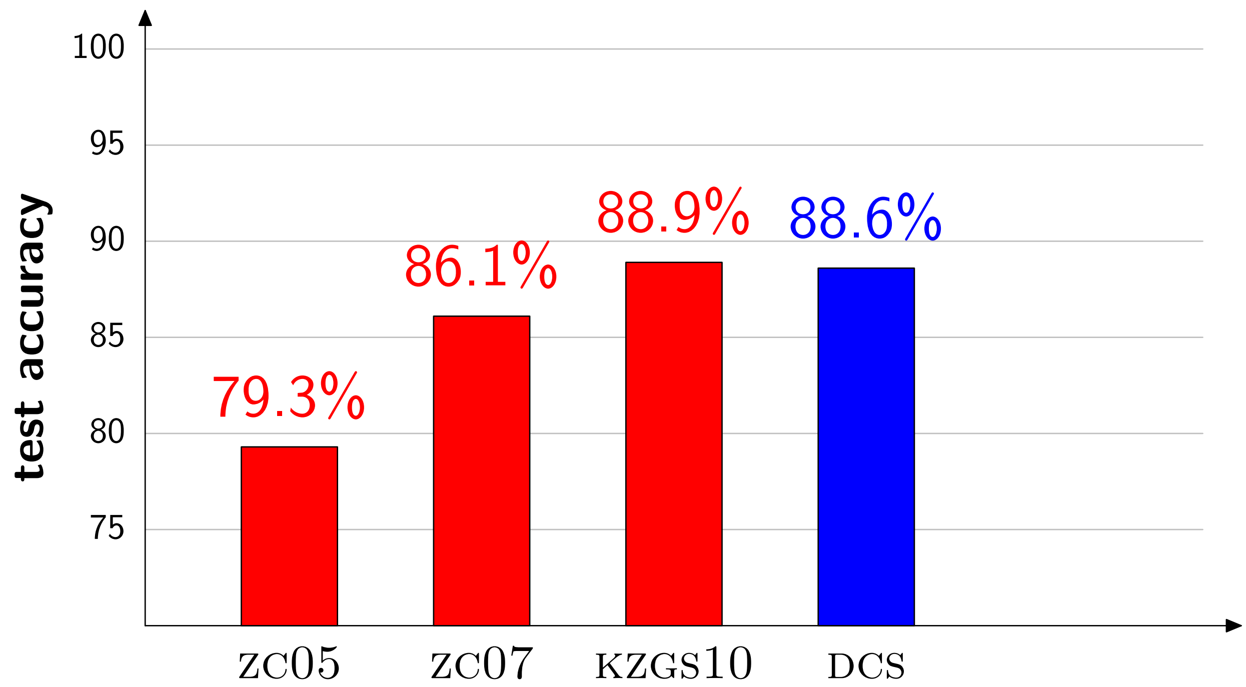
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓
KZGS10	CCG w/unification [Kwiatkowski et al., 2010]	✗ ✗	✓



Experiment 2

On GEO, 600 training examples, 280 test examples

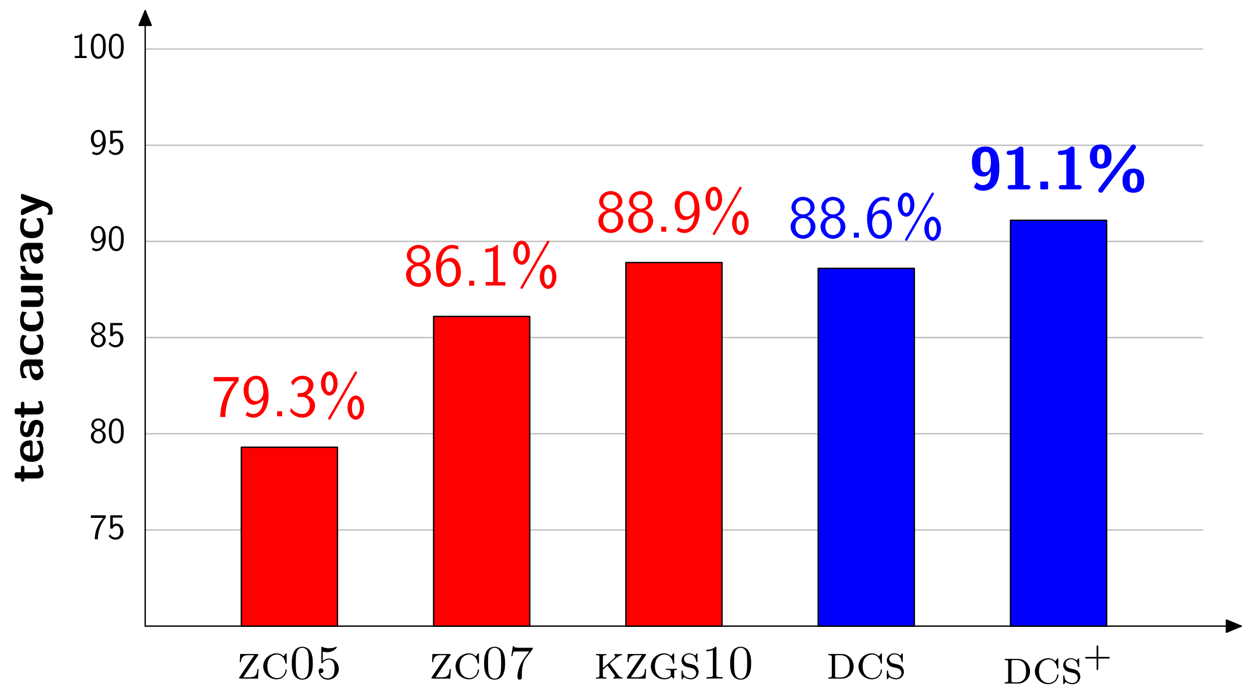
System	Description	Lexicon	Logical forms
ZC05	CCG [Zettlemoyer & Collins, 2005]	✗ ✗	✓
ZC07	relaxed CCG [Zettlemoyer & Collins, 2007]	✗ ✗	✓
KZGS10	CCG w/unification [Kwiatkowski et al., 2010]	✗ ✗	✓
LJK11	DCS [Liang et al., 2011]	✓ ✗	✗



Experiment 2

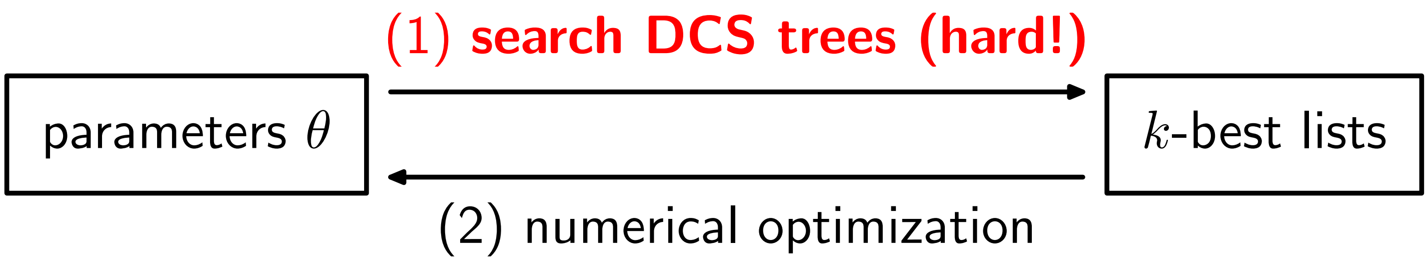
On GEO, 600 training examples, 280 test examples

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

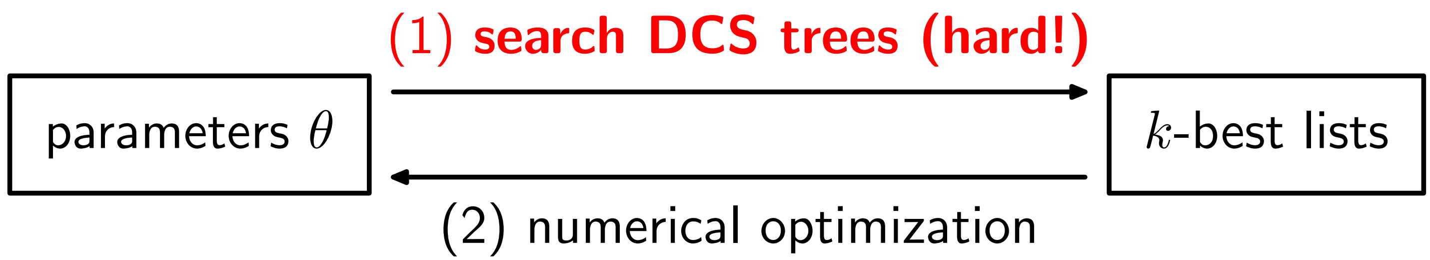


Some Intuition on Learning

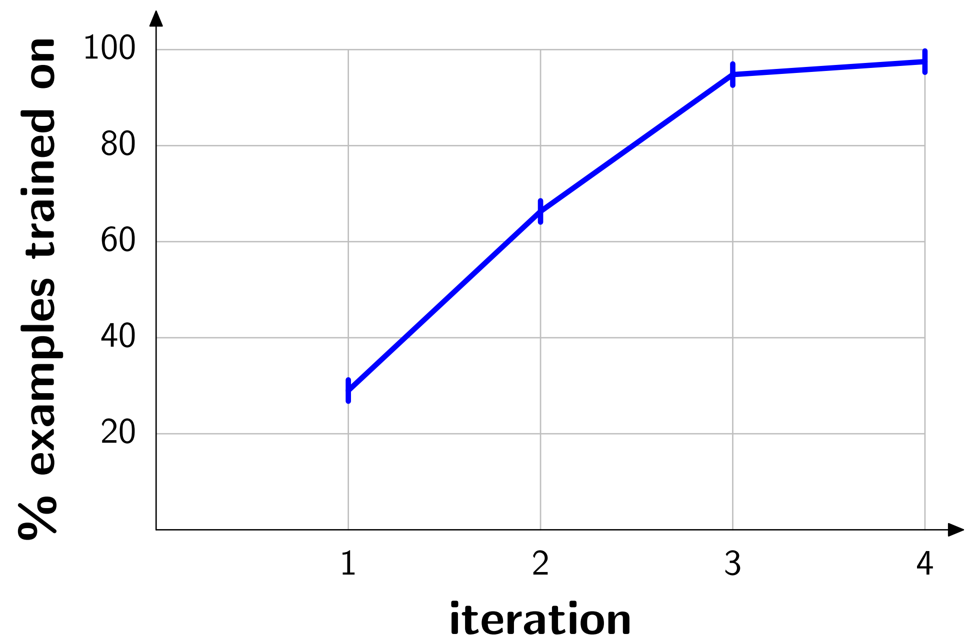
$$\boxed{\text{parameters } \theta} \underset{\text{(2) numerical optimization}}{\overset{\textbf{(1) search DCS trees (hard!)}}{\rightleftarrows}} \boxed{k\text{-best lists}}$$

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

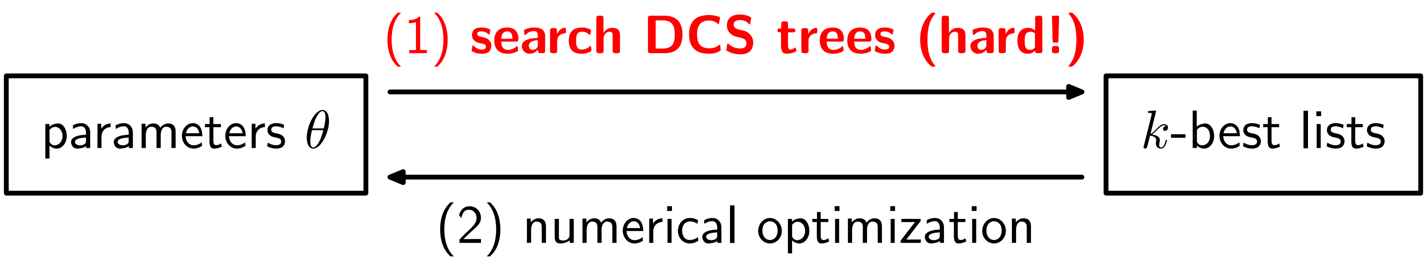
Some Intuition on Learning



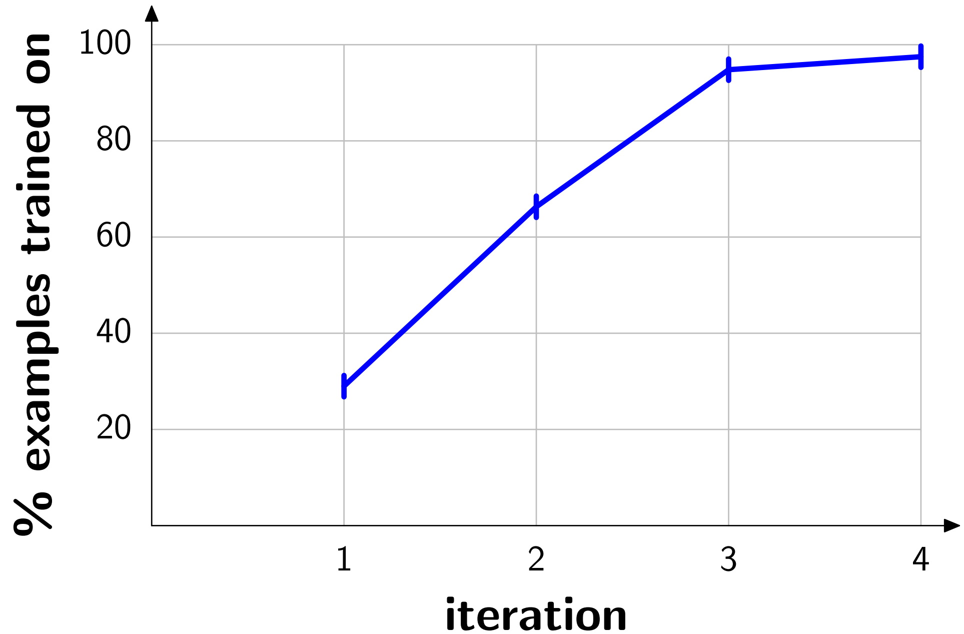
If no DCS tree on k-best list is correct, skip example in (2)



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

sentence → Semantic Parser → logical form → Interpretation → denotation

Learning from Weak Supervision

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

Summary

sentence → Semantic Parser → logical form → Interpretation → denotation

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