

*Syntactic and semantic parsing
for natural language understanding*

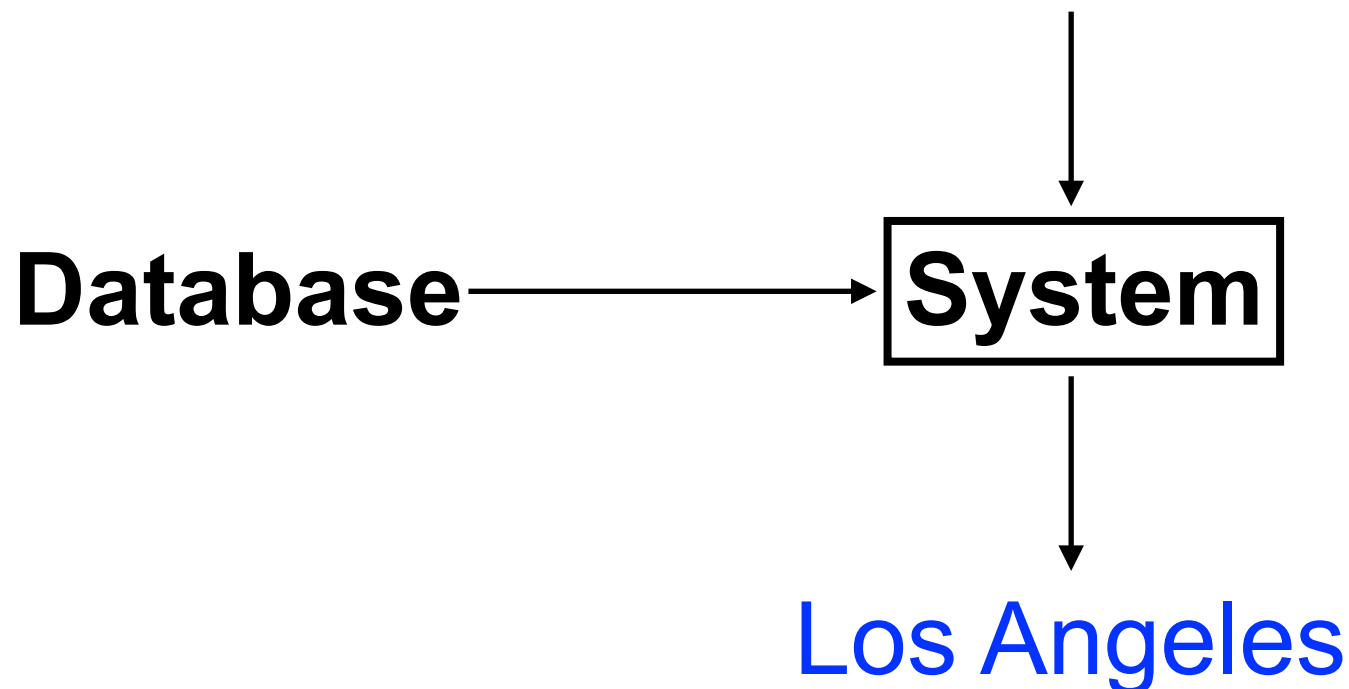
2. Question answering

Hiroshi Noji

Question answering

- ▶ **Main focus:** question answering with external **database**
- ▶ System may convert an input question into a **query** to the database, and obtain the answer

What is the most populous city in California?



Outline

- ▶ Difference between RTE and QA
 - Pure bottom-up approach like ccg2lambda is difficult for QA
 - Grounding is necessary
- ▶ Top-down semantic parsing for QA
 - Learning CCG from sentence/logical form pairs
 - Learning dependency-based compositional semantics
 - an approach to learn semantic parsing without supervision to logical forms

Concrete task: querying SPARQL

- ▶ Often a specific database accepts a predefined query language
 - e.g., Freebase is a very large open domain database, which accepts SPARQL
 - So our task is converting an input sentence into a SPARQL query

Input: *How many teams participate in UEFA?*

SPARQL: `select count(?x) where {
 ?a Team ?x .
 ?a League Uefa .
}`

Equivalent
lambda-expression: *$count(\lambda x. \exists a. Team(x, a) \wedge League(a, Uefa))$*

Problem

How many teams participate in UEFA?

Desirable
logical-form

count($\lambda x. \exists a. Team(x, a) \wedge League(a, Uefa)$)

Parser output

count($\lambda x. \exists a. Team(x, a) \wedge Participate(a, Uefa)$)

- ▶ Consider running a CCG parser to get a logical form
 - We can convert a logical expression into a SPARQL query by rules
- ▶ However, DB query must be written with keywords in the DB
 - Because the language used in DB is strict
 - This is in contrast to RTE, where the logical forms are inputs to a prover that uses the knowledge on word variations internally

Grounding

- ▶ For QA, we need to find the mapping from **words in the input** to **corresponding DB keywords**
- ▶ Technique to find such mappings is called **grounding**
 - In general, grounding connects a sentence with a real word entity
 - e.g., image grounding: we describe “panda” (text) by a figure of pandas (real entity)
 - In DB, the real entity is DB keywords (e.g., *League*)
- ▶ The bottom-up approaches we discussed so far (for RTE) are not related with grounding, so they are called a **non-grounding** approach

Partly bottom-up approach?

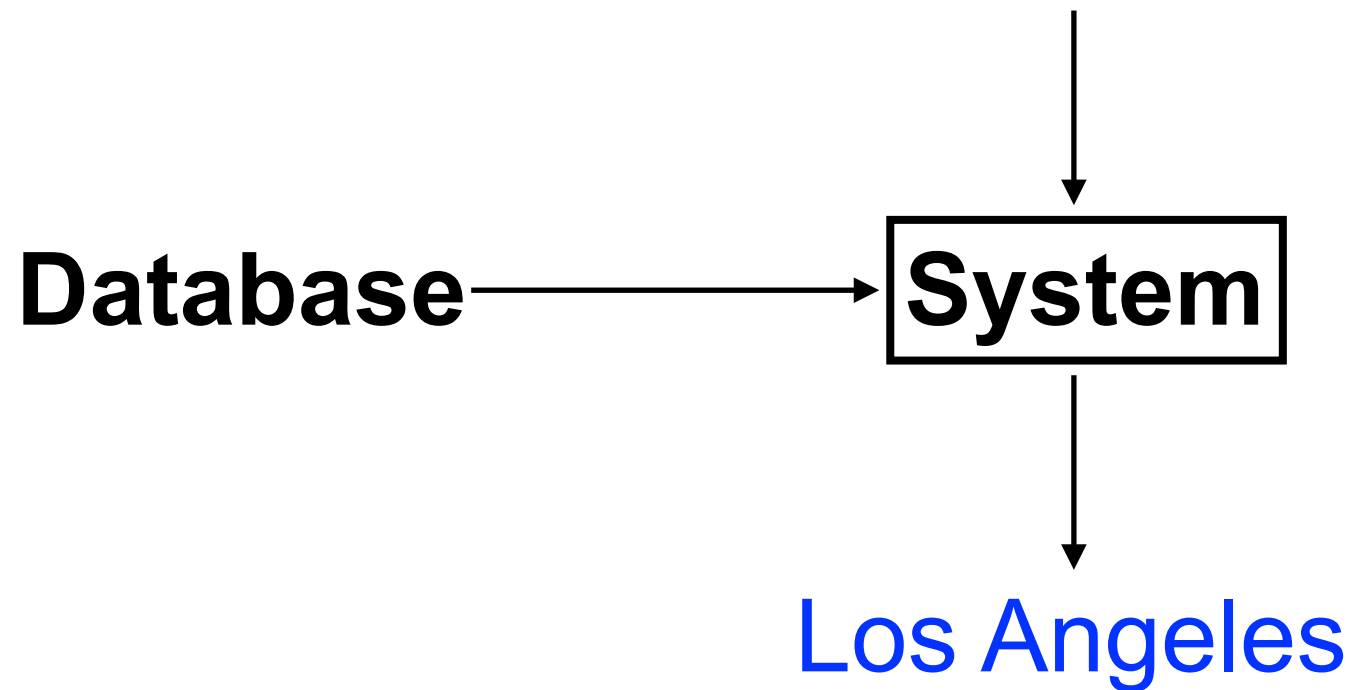
- ▶ So for DB, I don't know the fully bottom-up (non-grounding) approach built on a CCG parser or other parsers
- ▶ There is only a hybrid approach between bottom-up and top-down approach
 - First, parse an input sentence with a CCG parser to obtain a **non-grounding logical form**
 - Then convert it into a **grounded logical form** to be able to query to DB (using the idea of weak supervision, described later)
 - See: Reddy et al. Large-scale Semantic Parsing without Question-Answer Pairs. TACL 2016.

Outline

- ▶ Difference between RTE and QA
 - Pure bottom-up approach like ccg2lambda is difficult for QA
 - Grounding is necessary
- ▶ Top-down semantic parsing for QA
 - Learning CCG from sentence/logical form pairs
 - Learning dependency-based compositional semantics
 - an approach to learn semantic parsing without supervision to logical forms

Problem: obtaining logical forms

What is the most populous city in California?



► What **system** does:

- Convert a sentence into a logical form (query) \Rightarrow **difficult**
- Obtain the answer by querying to the database \Rightarrow **deterministic**

► **Challenge is how to convert a sentence into logical form**

Two different top-down approaches

- ▶ Top-down approaches are further divided into two, based on the form of supervision
 - **Fully supervised approach:**
 - Each training data is (sentence, **logical form**) pair
 - e.g., (*What's Bulgaria's capital?*, $\lambda x. capital(x, bulgaria)$)
 - **Pro**: Learning is more tractable
 - **Con**: Preparing many logical forms by hand is costly
 - **Weakly supervised approach:**
 - Each training data is (sentence, answer) pair
 - e.g., (*What's Bulgaria's capital?*, **Sofia**)
 - Logical form is **a latent variable in a model**
 - Much more challenging, but much more cheaper

Short history

- ▶ Fully-supervised approach was dominated until ~2011, 2012
- ▶ 2011, a successful weakly-supervised approach appeared:
 - Liang et al.,. Learning dependency-based compositional semantics. In *ACL* 2011.
 - Since then, weak supervision becomes a popular approach
- ▶ Advantage of weak supervision is **scalability**:
 - Writing logical form is impossible for ordinary people
 - But answering a question does not require linguistic knowledge
 - ⇒ We can use crowdsourcing to collect the data cheaply!

Fully supervised approach

Training data:

- a) What states border Texas
 $\lambda x.state(x) \wedge borders(x, texas)$
- b) What is the largest state
 $\arg \max(\lambda x.state(x), \lambda x.size(x))$
- c) What states border the state that borders the most states
 $\lambda x.state(x) \wedge borders(x, \arg \max(\lambda y.state(y), \lambda y.count(\lambda z.state(z) \wedge borders(y, z))))$



training a parser

How many states border Oregon? → **Semantic parser** → $count(\lambda x.state(x) \wedge borders(x, oregon))$

Note on logical forms of DB query

- ▶ How can we read $count(\lambda x.state(x) \wedge borders(x, oregon))$?
- ▶ $\lambda x.f(x)$ is a function
 - In DB query, a function can be seen as a set (entities satisfying $f(x)$ relation)
- ▶ So $count(\lambda x.f(x))$ returns the size of entities satisfying $f(x)$
- ▶ We assume a DB, which keeps unary and binary relations
 - $\lambda x.borders(x, oregon)$ is a operation retrieving all entries where the second column is *oregon*
 - $state(x)$ restricts the results to the values found in **state**

state	border	
<i>california</i>	<i>california</i>	<i>oregon</i>
<i>texas</i>	<i>nevada</i>	<i>oregon</i>
<i>nevada</i>	<i>long view</i>	<i>oregon</i>
...	<i>nevada</i>	<i>utah</i>
...	...	

CCG can help?

Training data:

a) What states border Texas

$\lambda x.state(x) \wedge borders(x, texas)$

b) What is the largest state

$\arg \max(\lambda x.state(x), \lambda x.size(x))$

c) What states border the state that borders the most states

$\lambda x.state(x) \wedge borders(x, \arg \max(\lambda y.state(y), \lambda y.count(\lambda z.state(z) \wedge borders(y, z))))$

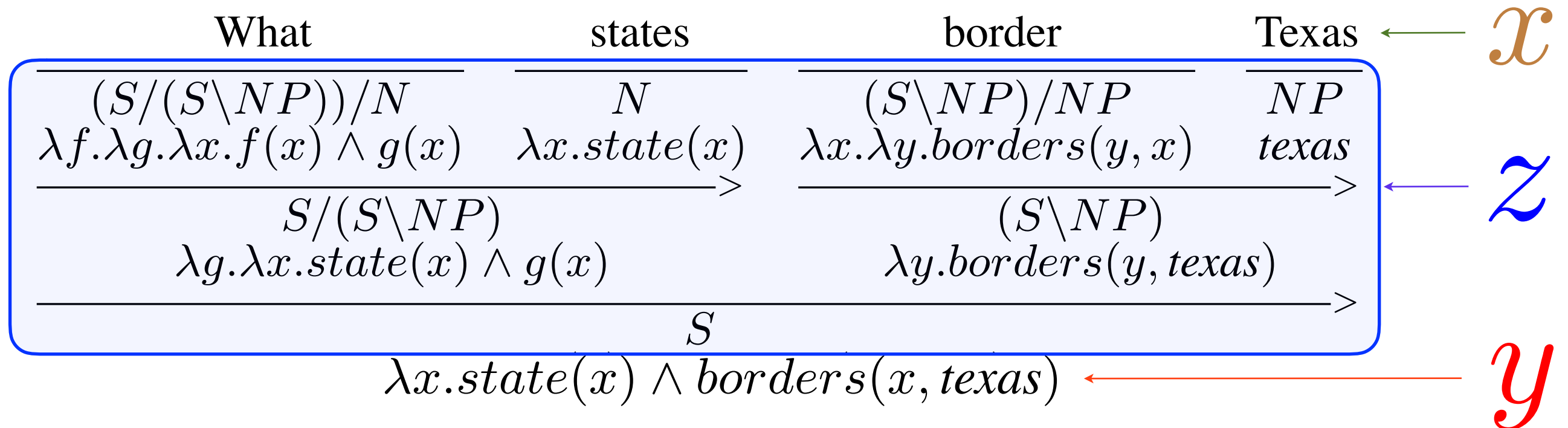
► Yes

- There are many approaches with this supervision relied on CCG
- Zettlemoyer & Collins, 2005, 2007; Kwiatkowski et al., 2010, 2011; Artzi & Zettlemoyer, 2011, 2013; etc.

► Difference from RTE:

- We do not use the output of CCG parser, but **induce the grammar**

Inducing CCG from logical forms



► We treat CCG derivation as a latent variable (z) in the model

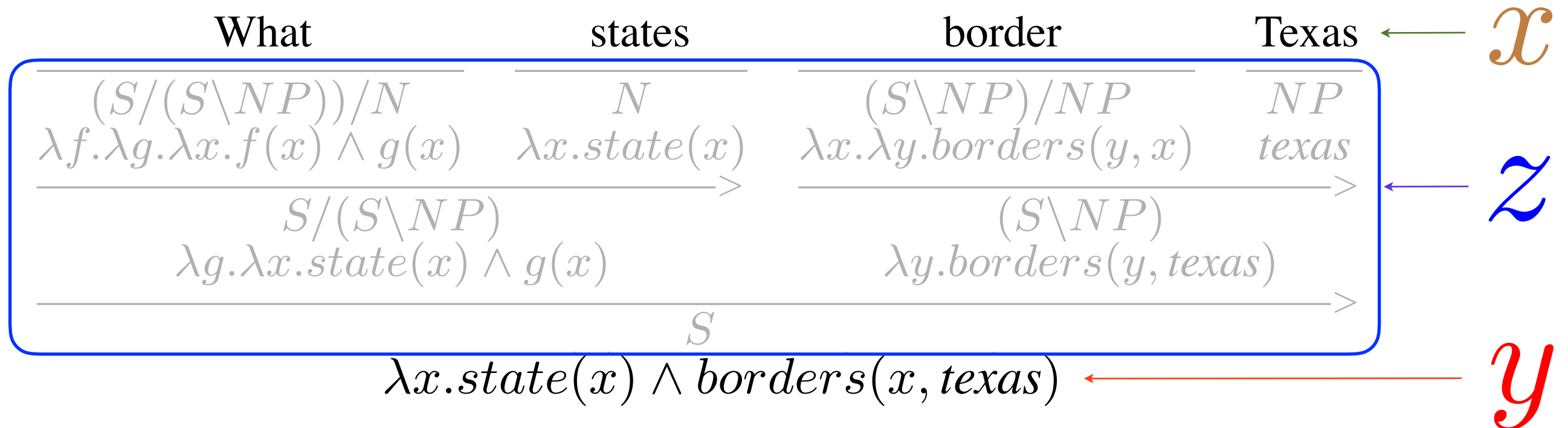
► Objective function: $\sum_{i=1}^n \log \left(\sum_z p(y, z | x, \theta) \right)$

- maximize the likelihood marginalizing z
- can be done by a variant of SGD (or a latent-variable perceptron)

Why we don't use existing parsers?

- ▶ Probably the main reason is historical
 - We can of course utilize the results of existing CCG parsers
 - But the approach without existing parsers succeeded and got more popular
 - Note that even using other parsers, **grounding (supervised learning by gold logical forms) is necessary**
- ▶ Advantage of inducing grammar (not using existing parsers):
 - Free from parsing errors (error propagation)
 - (Possibly) developing a system for other languages is easier
 - CCG parsers are not available in many languages

Is this fully-supervised?



- In some sense, this approach is also a kind of weakly-supervised approach
 - in that we do not assume a gold syntactic derivation into the desired logical form (y)
 - We infer the latent CCG derivation (z) that bridges x and y
 - EM-like algorithm

Assumption (seed knowledge)

► Knowledge about CCG rules:

- forward/backward application/composition, etc.
- e.g., $X/Y: f \quad Y/Z: g \rightarrow X/Z: \lambda x.f(g\ x) \quad (>B)$

► **Type of logical form** for each CCG category

- NP : a (constant)
- N : $\lambda x.p(x)$
- S\NP/NP : $\lambda x.\lambda y.p(x,y)$

► Initial lexicon (domain-independent important words):

- What: S/(S\NP)/N: $\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$

Finding categories on words

What states border Texas

$\lambda x.state(x) \wedge borders(x, texas)$

NP *texas*

N $\lambda x.state(x)$

S\NP/NP $\lambda x.\lambda y.borders(x, y)$

S\NP/NP $\lambda x.\lambda y.borders(y, x)$

S\NP $\lambda x.borders(x, texas)$

S/(S\NP)/N $\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$

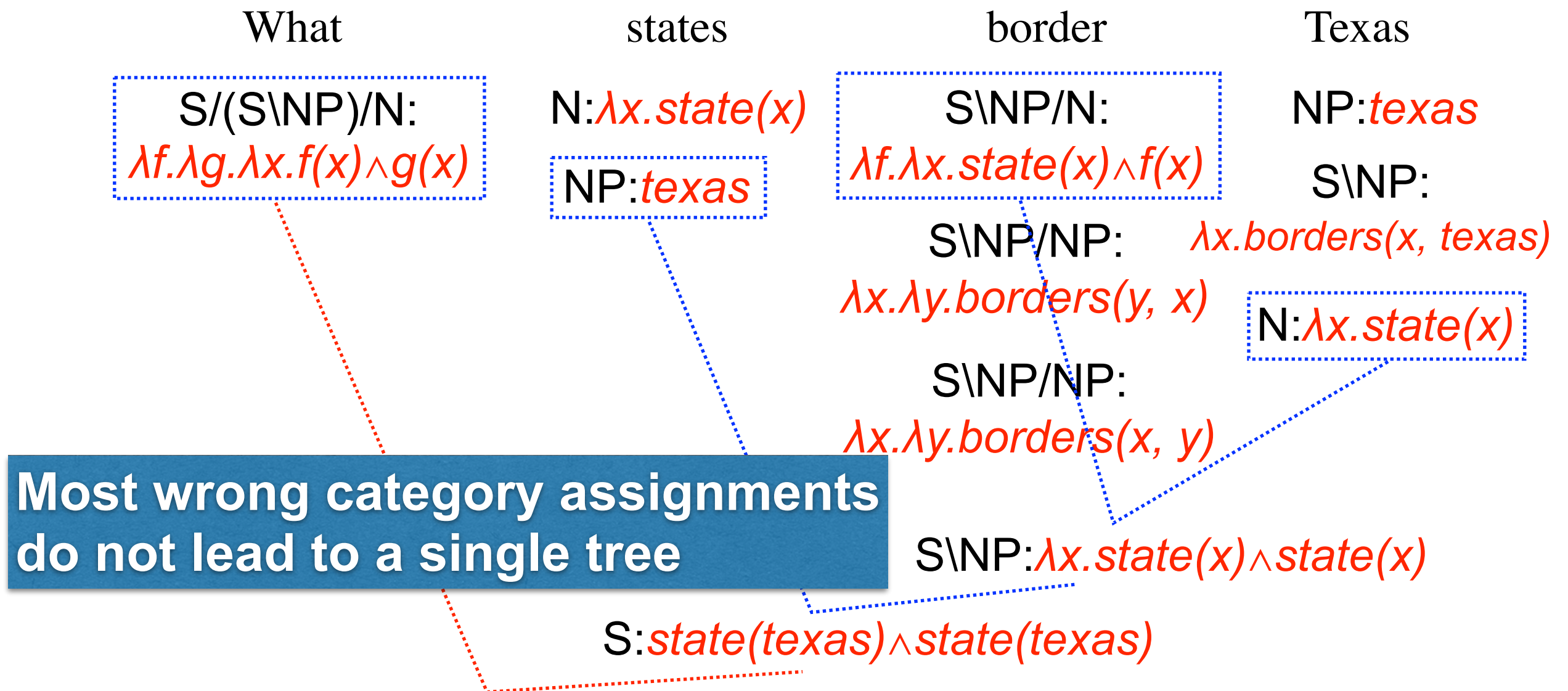
S\NP/N $\lambda f.\lambda x.state(x) \wedge f(x)$

...

...

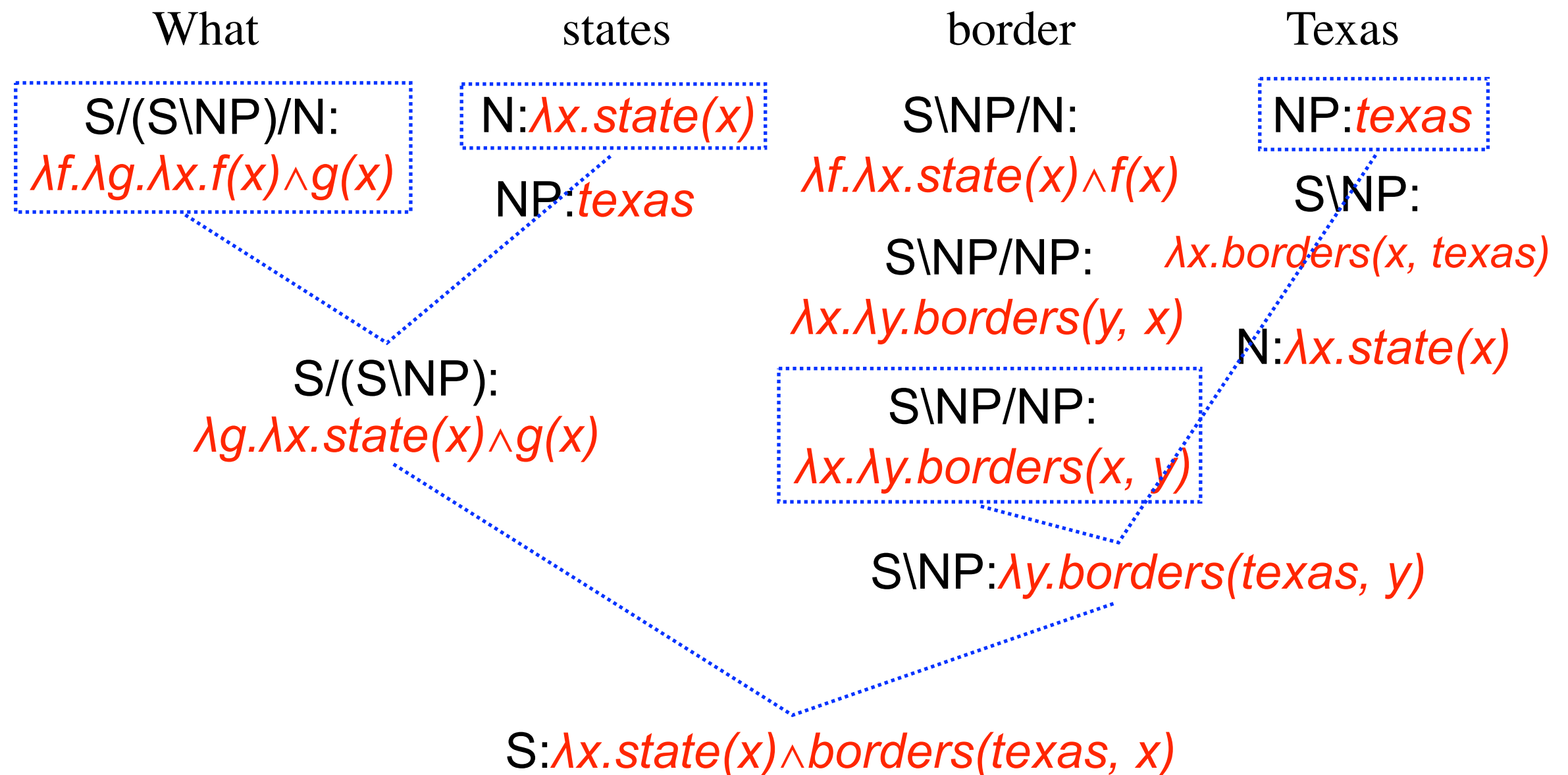
- First step is decomposing the final logical form into smaller pieces
- Corresponding CCG categories are limited due to the type constraints (isomorphism)

How learning proceeds?



- ▶ There is no constraint between each word and category
 - Learning this mapping is the main challenge
- ▶ We want to find mapping leading to the correct logical form

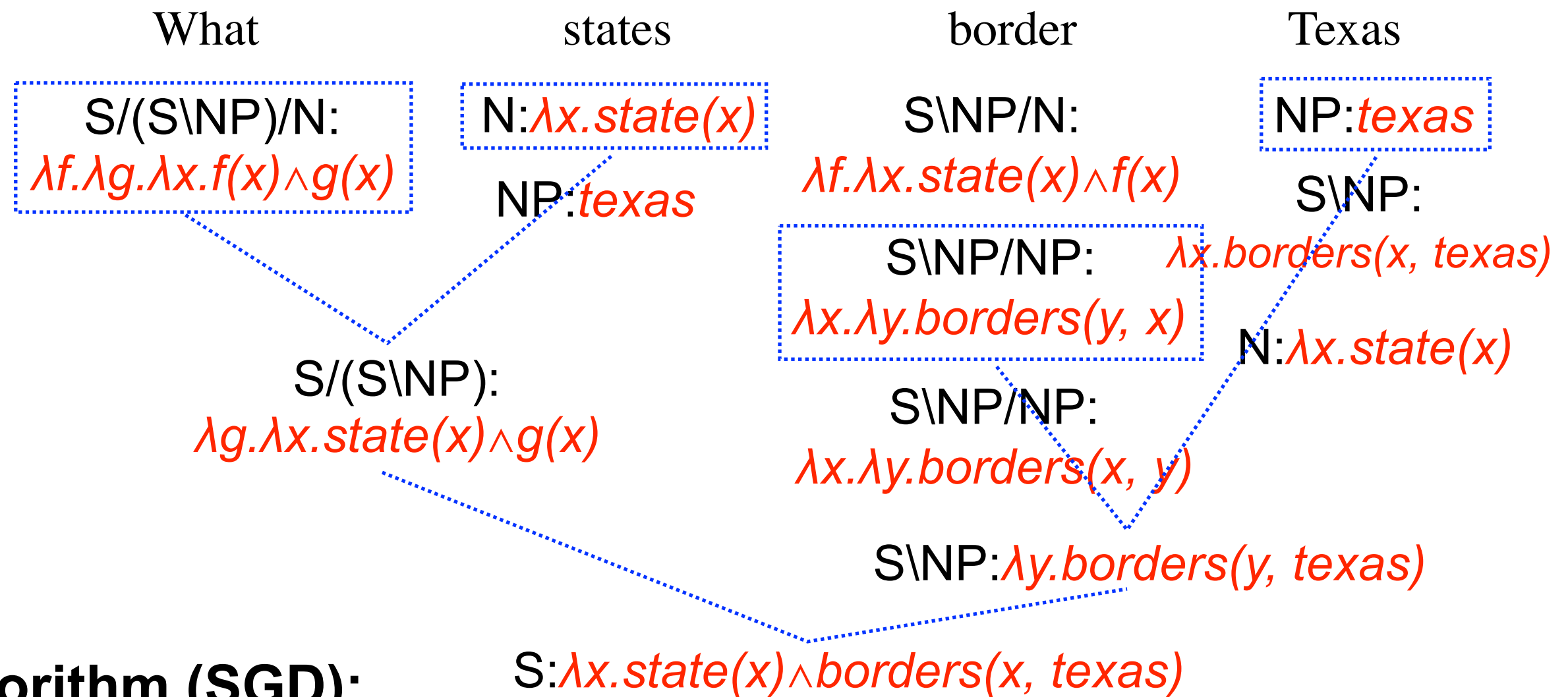
How learning proceeds?



Other incorrect category assignments
(mostly) lead to wrong logical forms

gold: $\lambda x. state(x) \wedge borders(x, texas)$

How learning proceeds?



for each (sentence, logical form) pair:

1. find derivations leading to the given logical form
2. strengthen the weights appeared in the found derivations

states \rightarrow $N:\lambda x.state(x)$ 5.0 \Rightarrow 5.8 (+0.8)

states \rightarrow $NP:texas$ 0.2 \Rightarrow -0.5 (-0.7)

Why did this approach succeed?

- ▶ Strong inductive bias of CCG
 - Combinatory rules of CCG highly restrict the candidate lexical categories → **facilitate mapping of words and categories**
 - Such implicit bias to help machine learning algorithm is called “inductive bias”, which is essential for many NLP systems
- ▶ Sentences are not complex
 - Question sentences are not very complex
 - They are typically 5~10 words
 - search space for a model is not so quite large

Later extensions

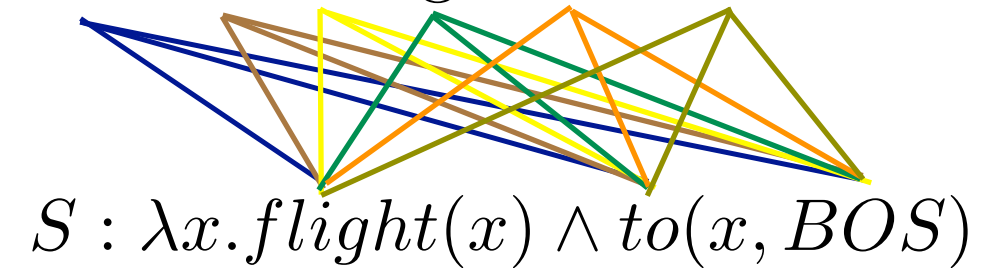
► Zettlemoyer and Collins (UAI 2005)

- first approach; described so far
- use some initial lexicon (e.g., *what, every*)

► Kwiatkowski et al., (EMNLP 2010)

- Reducing supervision signals by learning mapping of every word
- using alignment technique in machine translation (IBM model) to obtain better initial parameters

I want a flight to Boston



► Kwiatkowski et al., (EMNLP 2011)

- Further refinements in the probabilistic model

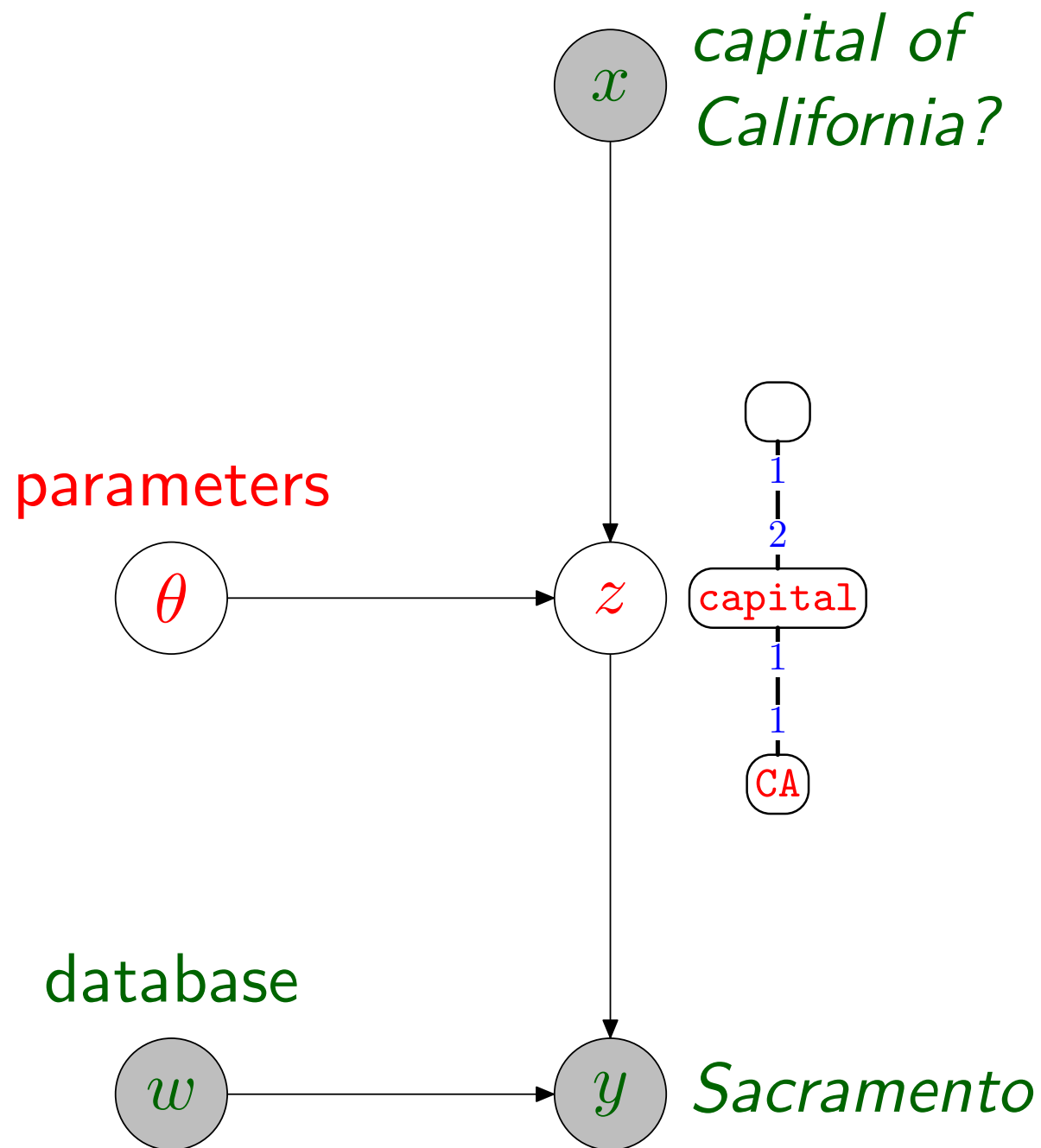
Outline

- ▶ Difference between RTE and QA
 - Pure bottom-up approach like ccg2lambda is difficult for QA
 - Grounding is necessary
- ▶ Top-down semantic parsing for QA
 - Learning CCG from sentence/logical form pairs
 - Learning dependency-based compositional semantics
 - an approach to learn semantic parsing without supervision to logical forms

Weakly-supervised approach

- ▶ CCG induction requires (sentence, logical form) pairs, but annotating logical forms are costly
 - requires expert knowledge
- ▶ Can we learn a semantic parser only from (sentence, answer) pairs?
 - Liang et al. showed this is possible
 - **Idea:**
 - Logical form is treated as a latent variable in the model
 - This logical form is simpler (weaker than first-order logic) and aimed at querying DB (**very task-specific**)
 - Learning correct logicals form via SGD-like algorithm

Logical form is latent variable



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

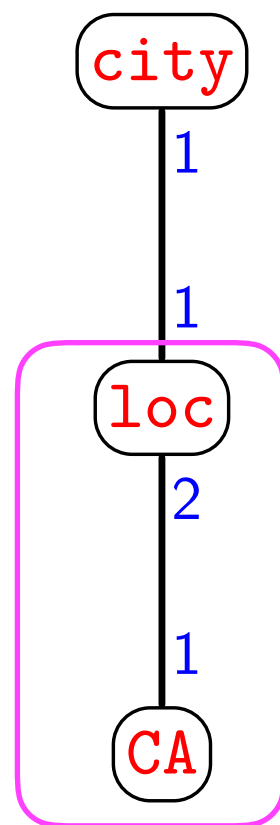
Interpretation: $p(y \mid z, w)$
(deterministic)

DCS

Dependency-based compositional semantics

For “*city in California*”

DCS tree



Constraints

$c \in \text{city}$

$c_1 = \ell_1$

$\ell \in \text{loc}$

$\ell_2 = s_1$

$s \in \text{CA}$

Database

city

San Francisco
Chicago
Boston
...

loc

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

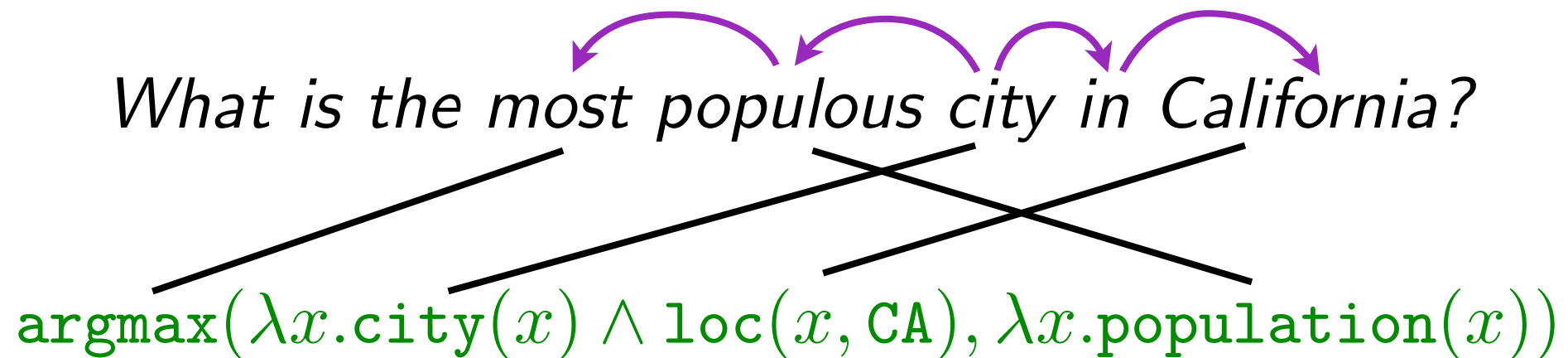
CA

California

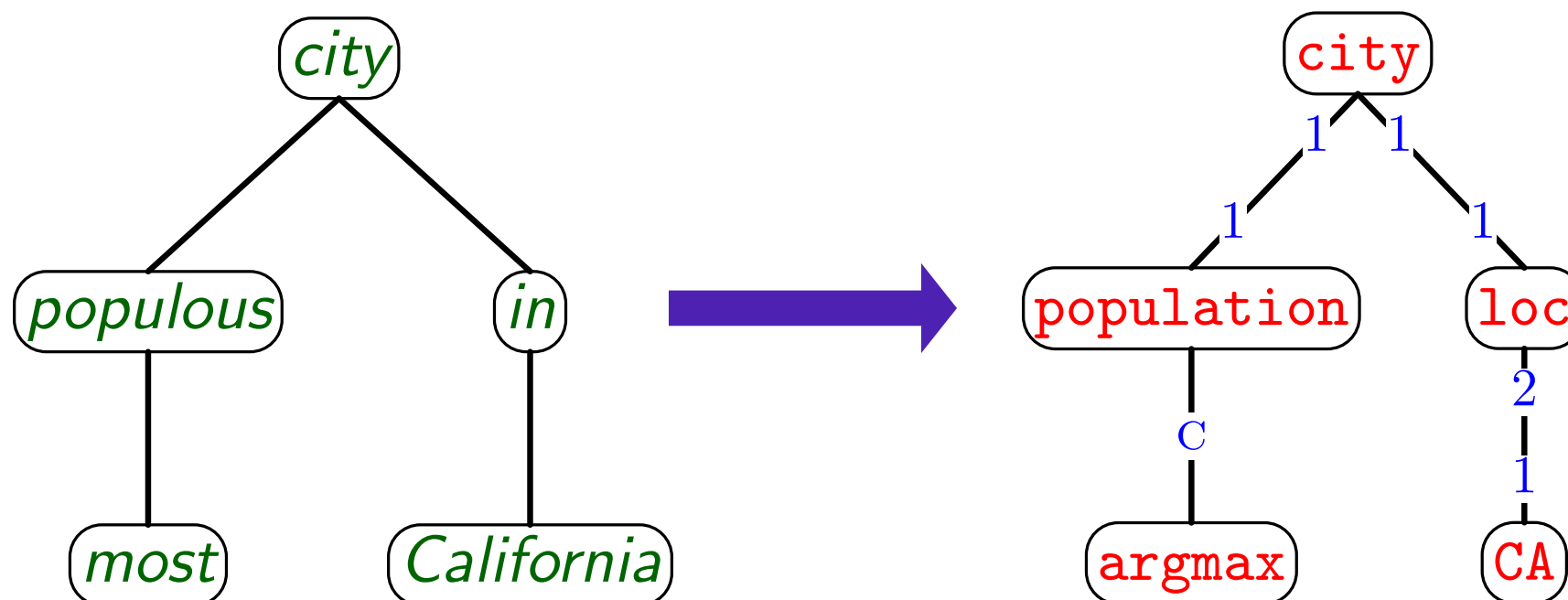
A subtree represents a set
(Rows of **loc** where 2nd column is **CA**)

Comparison to lambda expression

- ▶ Gap between syntax of sentence and lambda expression is large



- ▶ Syntax of DCS much resembles dependency structure

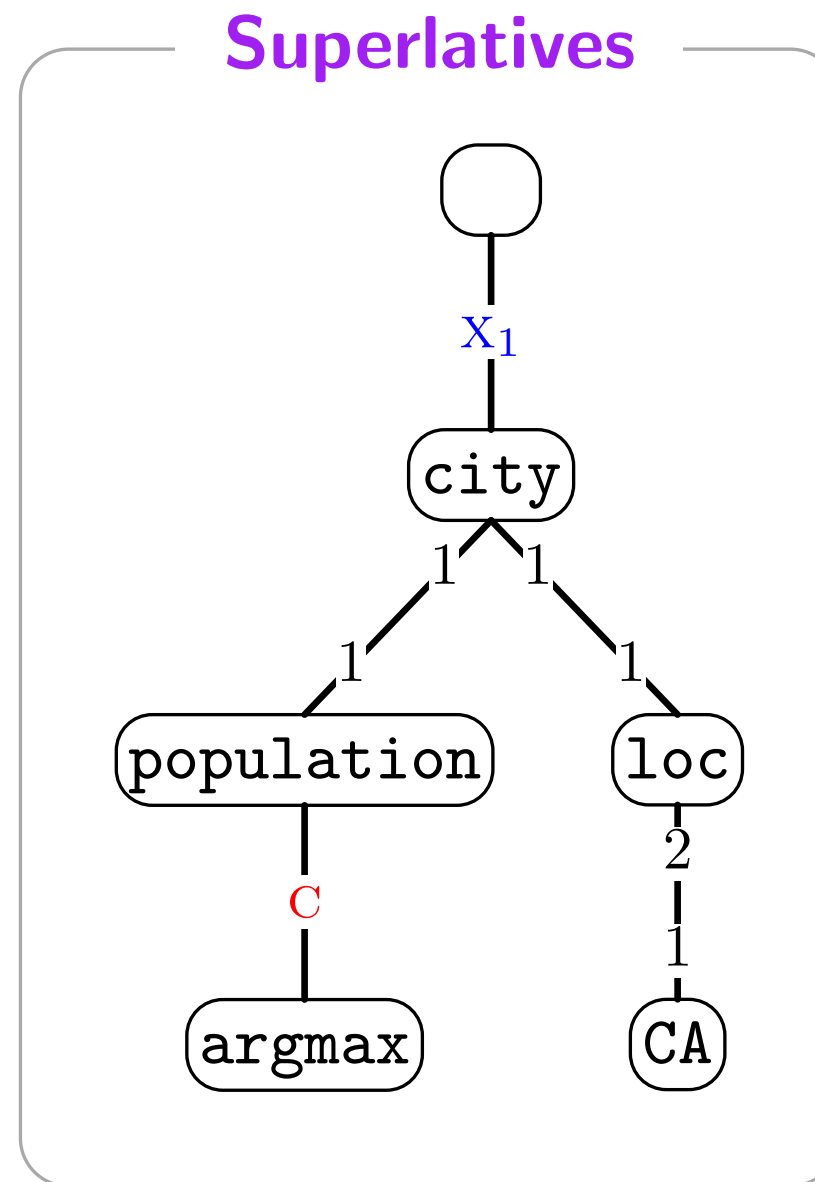


Trick for resemblance to dependency

most populous city in California

Execute at semantic scope

Mark at syntactic scope



- Explanation is simple, but actual mechanism is complex

Learning

- ▶ Basically the same as CCG-based approach
 - Though DCS resembles dependency structure, we do not use any existing parsers
- ▶ Difference: We do not have the correct logical form
 - **We can infer whether the created DCS tree is correct or not, by directly querying it to DB**
 - **If DB returns the correct answer, the DCS tree is probably correct (weak-supervision)**
 - Note: We can convert a DCS tree into a DB-specific logical form

Initial lexicon

What is the most populous city in CA ?

city state river population city
state river population CA
river population city in CA ?

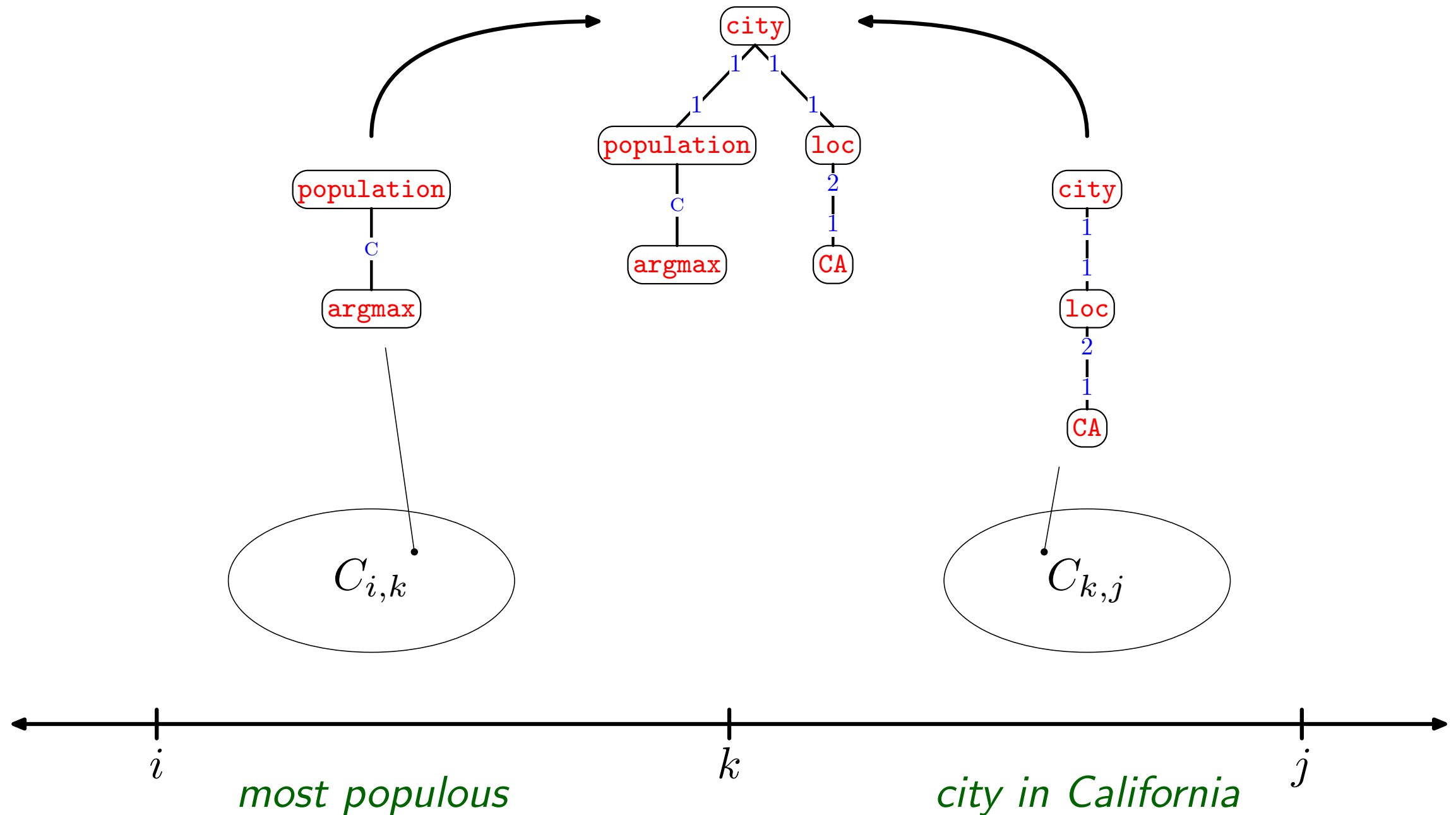
Lexical Triggers:

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

► Learning from answer only is very hard

- We assume some initial lexicon (more than initial CCG-based approach), which is necessary

SGD-like algorithm

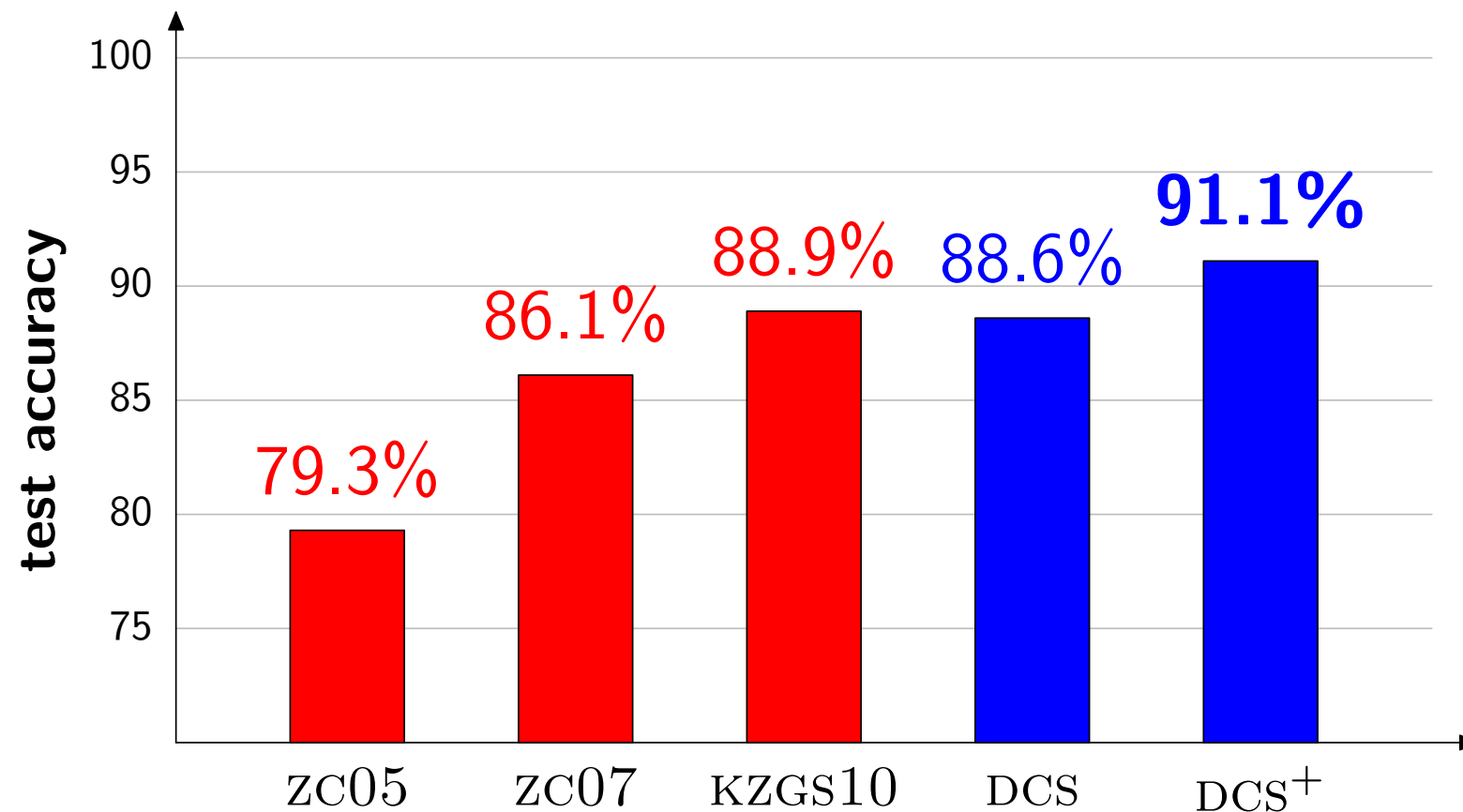


- Find K-best DCS derivations under the current model (beam search)
- Push the weights of features appeared in derivations that return the correct answer

Results

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]	✗	✗	✓
DCS	our system	✓	✗	✗
DCS ⁺	our system	✓	✓	✗



Plan of the last lecture

- ▶ We have seen a short history of semantic parsing for QA until DCS appeared in 2011
 - That time the domain was very limited (e.g., US geography)
 - Words are limited in a domain, and learning was not so hard
- ▶ In the last lecture, we discuss more recent research efforts
 - Open-domain QA (on Wikipedia, and Google Freebase)
 - QA from a semi-structured table
 - Context-dependent QA
 - (multi-modal QA?)