Learning Map Sentences to Meaning

Luke Zettlemoyer
University of Washington
Computer Science & Engineering

joint work with Michael Collins, Sharon Goldwater,
Tom Kwiatkowski, Mark Steedman

From Text to Meaning

We want to build systems that recover meaning from text

Increasingly Informative Meaning Representation

From Text to Meaning

We want to build systems that recover meaning from text

Information Extraction
Recover information about pre-specified entities and relations

Increasingly Informative Meaning Representation

Example Task
Relationship Extraction

OBAMA is **PRESIDENT**

From Text to Meaning

We want to build systems that recover meaning from text

Broad-coverage Semantics
Focus on specific phenomena, e.g. matching verbs to their arguments

Increasingly Informative Meaning Representation

Example Task
Summarization


The Washington Post
'JUSTICE HAS BEEN DONE'
U.S. forces kill Osama bin Laden

Us forces killed Osama Bin Laden in his compound in Abbottabad.

From Text to Meaning

We want to build systems that recover meaning from text

Supervised Semantic Parsing
Recover Complete Meaning Representations

Increasingly Informative Meaning Representation

Example Task
Database Querying

Question:
What states border texas?

Database

Answer:
Oklahoma
New Mexico
Arkansas
Louisiana

Mapping Sentences to Meaning

```
Texas borders Kansas.
```

Mapping Sentences to Meaning

`Texas borders Kansas.`

next-to(TEX,KAN)

Mapping Sentences to Meaning

What states border Texas?

$\lambda x.state(x) \wedge next\text{-}to(x, TEX)$

Mapping Sentences to Meaning

`What states border Texas?`

$\lambda x.state(x) \wedge next\text{-}to(x, TEX)$

Machine Learning Problem

Given: Many input, output pairs

Learn: A function that maps sentences to lambda-calculus expressions

More Examples

Input: `What is the largest state?`

Output: $argmax(\lambda x.state(x), \lambda y.size(y))$

Input: `What states border the largest state?`

Output: $\lambda z.state(z) \wedge borders(z, argmax(\lambda x.state(x), \lambda y.size(y)))$

Input: `What states border states that border states ... that border Texas?`

Output: $\lambda x.state(x) \wedge \exists y.state(y) \wedge \exists z.state(z) \wedge \ldots \wedge borders(x,y) \wedge borders(y,z) \wedge borders(z,texas)$

Many Potential Applications

This talk: Natural language interfaces to databases

> What states border Texas?
[Louisiana, Arkansas, Oklahoma, New Mexico]

> Which is the largest?
[New Mexico]

> List the rivers that run through it.
[…]

Many Potential Applications

This talk: Natural language interfaces to databases

> What states border Texas?

[Louisiana, Arkansas, Oklahoma, New Mexico]

> Which is the largest?

[New Mexico]

> List the rivers that run through it.

[…]

Soon: Conversational systems

Long Term: Machine translation, Document understanding

Why Machine Learning?

Need to analyze complex sentences:

- show me all flights both direct and connecting to either san francisco or oakland from boston that arrive before 2pm
- where does delta fly to that american doesn't
- which airline has more business class flights than any other airline
- eastern flies from atlanta to denver what type of aircraft do you use before 6pm

Why Machine Learning?

Need to analyze complex sentences:

- show me all flights both direct and connecting to either san francisco or oakland from boston that arrive before 2pm
- where does delta fly to that american doesn't
- which airline has more business class flights than any other airline
- eastern flies from atlanta to denver what type of aircraft do you use before 6pm

Traditional Approach: hand-engineered systems

- Many person-years spent on each application

Machine Learning: only need training data

- Techniques apply across applications

A Challenge: Structured Input, Output

Machine Learning: Input X and Output Y

- given training data, a set of pairs $(x, y), x \in X, y \in Y$
- find a function $f : X \to Y$

A Challenge: Structured Input, Output

Machine Learning: Input X and Output Y

- given training data, a set of pairs $(x, y), x \in X, y \in Y$
- find a function $f : X \to Y$

Binary classification: $x \in \mathbb{R}^d,\ y \in \{-1, +1\}$

A Challenge: Structured Input, Output

Machine Learning: Input X and Output Y

- given training data, a set of pairs $(x, y), x \in X, y \in Y$
- find a function $f : X \rightarrow Y$

Binary classification: $x \in \mathbb{R}^d, \; y \in \{-1, +1\}$

This talk: x is a sentence, y is a lambda-calculus expression

`what states border texas`

$$\Downarrow f$$

λ*x*.*state*(*x*) ∧ *next-to*(*x*,*TEX*)

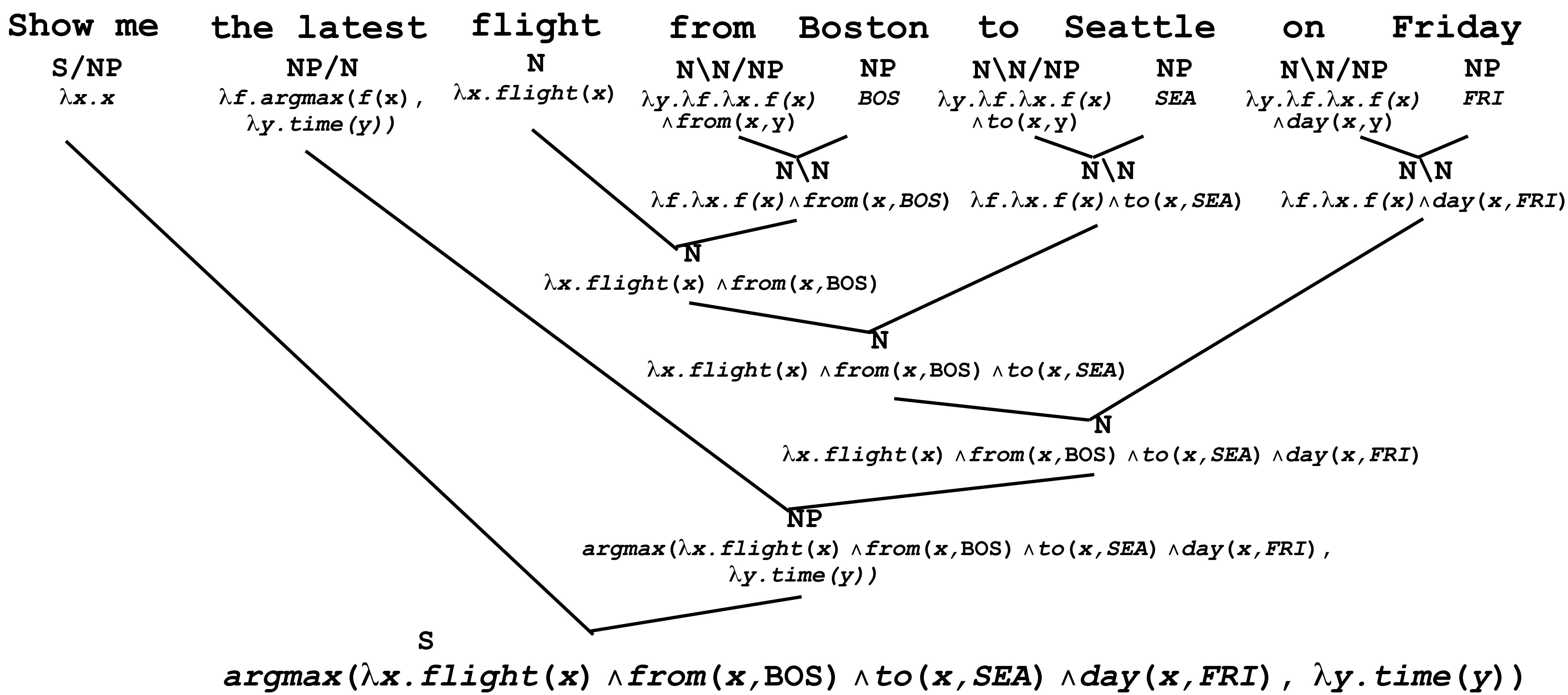
Key Challenge: outputs have rich structure (lambda-calculus)

A Challenge: Learning Hidden Structure

Approach 1. Fully annotated training examples (parse trees):

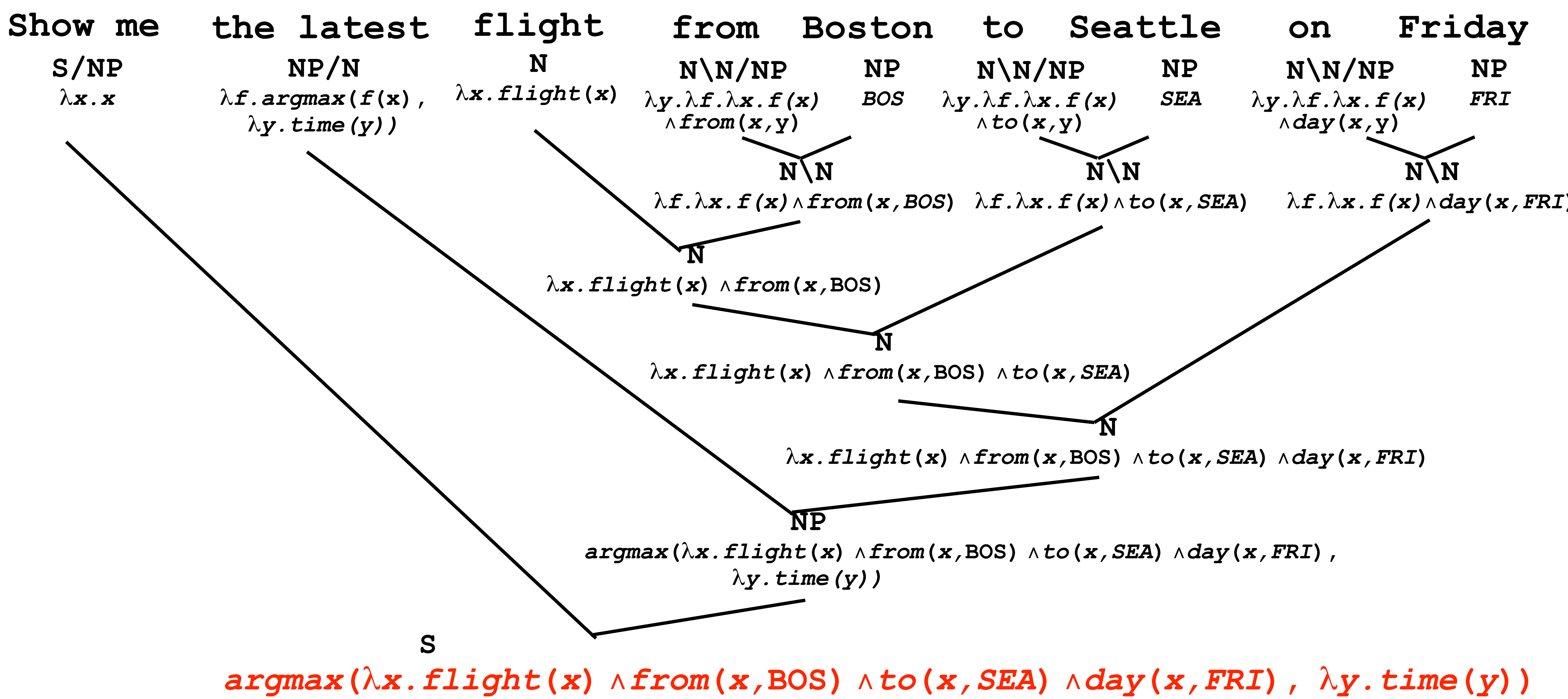
A Challenge: Learning Hidden Structure

Approach 1. Fully annotated training examples (parse trees):



A Challenge: Learning Hidden Structure

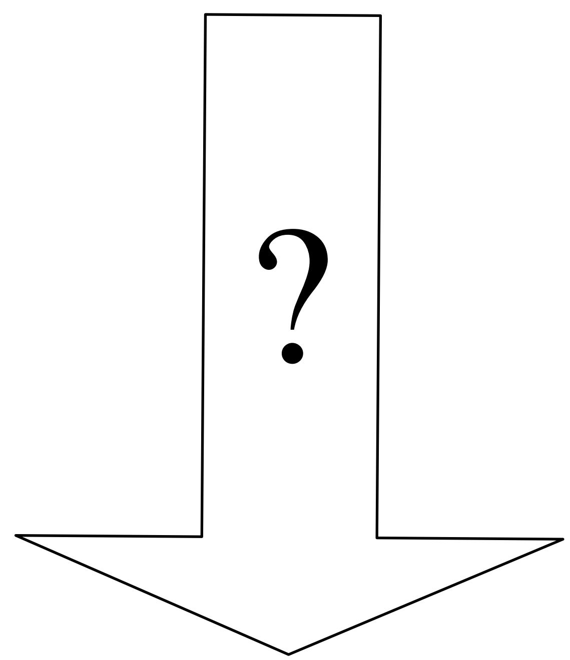
Our approach. Only requires annotations of final meanings



A Challenge: Learning Hidden Structure

Our approach. Only requires annotations of final meanings

`Show me the latest flight from Boston to Seattle on Friday`



$argmax(\lambda x.flight(x) \wedge from(x,BOS) \wedge to(x,SEA) \wedge day(x,FRI), \lambda y.time(y))$

Talk Outline

Learning to map sentences to meaning:

- Representing and recovering meaning
- An example supervised learning algorithm
- Other problems: interpreting instructions, grounding, task-oriented dialog, talking to robots

Combinatory Categorial Grammars (CCG)

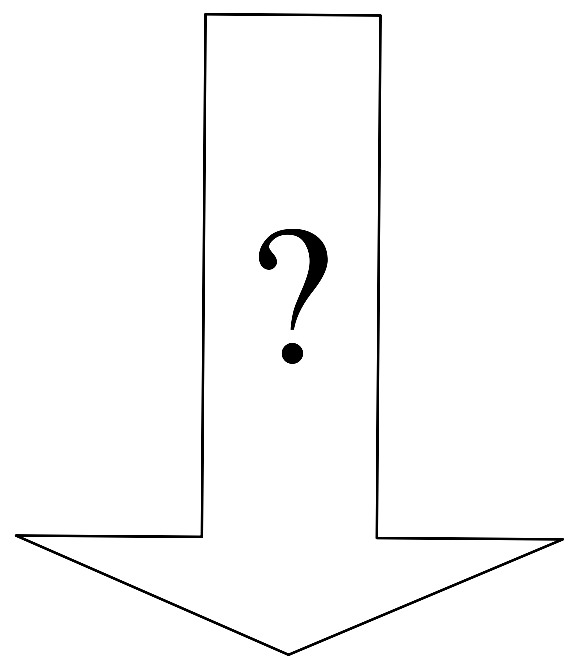
We will learn a linguistically-plausible CCG grammar:

- mildly context-sensitive formalism
- explains a wide range of linguistic phenomena: coordination, long distance dependencies, etc.
- joint model of syntax and semantics
- statistical parsing algorithms exist

[Steedman 96,00]

Compositional Semantics

`The Mississippi traverses Texas`



`runs-through(MISS-RIV,TEX)`

Compositional Semantics

The Mississippi	traverses	Texas
MISS-RIV	$\lambda x.\lambda y.runs\text{-}through(y,x)$	*TEX*

[Montague, 70]

Compositional Semantics

The Mississippi traverses Texas

MISS-RIV $\boxed{\lambda x}.\lambda y.\textit{runs-through}(y,x)$ $\boxed{\textit{TEX}}$

[Montague, 70]

Compositional Semantics

The Mississippi traverses Texas

MISS-RIV λ***x.***λ***y.runs-through(y,x)*** ***TEX***

λ***y.runs-through(y,TEX)***

[Montague, 70]

Compositional Semantics

The Mississippi | traverses | Texas

$\boxed{\textit{MISS-RIV}}$ | $\lambda x.\lambda y.\textit{runs-through}(y,x)$ | $\textit{TEX}$

$\boxed{\lambda y}.\textit{runs-through}(y,\textit{TEX})$

[Montague, 70]

Compositional Semantics

`The Mississippi` `traverses` `Texas`

$MISS\text{-}RIV$ $\lambda x.\lambda y.runs\text{-}through(y,x)$ TEX

$\lambda y.runs\text{-}through(y,TEX)$

$runs\text{-}through(MISS\text{-}RIV,TEX)$

[Montague, 70]

Combinatory Categorial Grammar (CCG)

The Mississippi	traverses	Texas
NP	**(S\NP)/NP**	**NP**
MISS-RIV	$\lambda x.\lambda y.runs\text{-}through(y,x)$	***TEX***

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The	Mississippi	traverses	Texas
	NP	(S\NP)/NP	NP
	MISS-RIV	$\lambda x.\lambda y.runs\text{-}through(y,x)$	*TEX*

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The Mississippi	traverses	Texas
NP	**(S\NP)/NP**	**NP**
MISS-RIV	$\lambda x.\lambda y.\mathit{runs\text{-}through}(y,x)$	***TEX***

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The Mississippi	traverses	Texas
NP	(S\NP)$\boxed{\text{/NP}}$	$\boxed{\text{NP}}$
MISS-RIV	λ***x***.λ***y***.***runs-through***(***y***,***x***)	*TEX*

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The Mississippi	traverses	Texas
NP	**(S\NP)/NP**	**NP**
MISS-RIV	λ***x***.λ***y.runs-through*(*y*,*x*)**	***TEX***

S\NP

λ***y.runs-through*(*y*,*TEX*)**

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The Mississippi	traverses	Texas
$\boxed{\mathbf{NP}}$	**(S\NP)/NP**	**NP**
MISS-RIV	λ***x***.λ***y.runs-through*(*y*,*x*)**	***TEX***

$\mathbf{S}\boxed{\mathbf{\backslash NP}}$

λ***y.runs-through*(*y*,*TEX*)**

[Steedman 96,00]

Combinatory Categorial Grammar (CCG)

The Mississippi — traverses — Texas

NP — **(S\NP)/NP** — **NP**

MISS-RIV — λ***x***.λ***y.runs-through*(*y*,*x*)** — ***TEX***

S\NP

λ***y.runs-through*(*y*,*TEX*)**

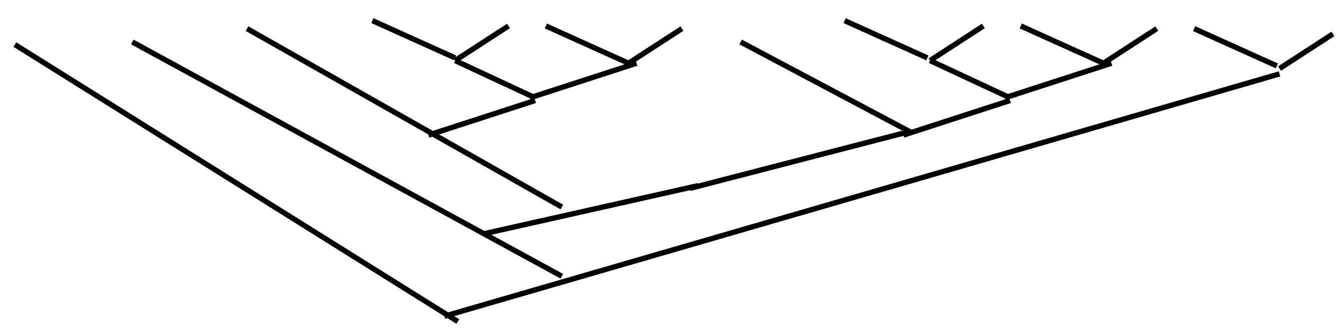
S

***runs-through*(*MISS-RIV*,*TEX*)**

[Steedman 96,00]

Models Complex Linguistic Effects

Show me flights from Newark and New York to San Francisco or Oakland that are nonstop.



$$\lambda x.flight(x) \wedge nonstop(x) \wedge (from(x,NEW) \vee from(x,NYC)) \wedge (to(x,SFO) \vee to(x,OAK))$$

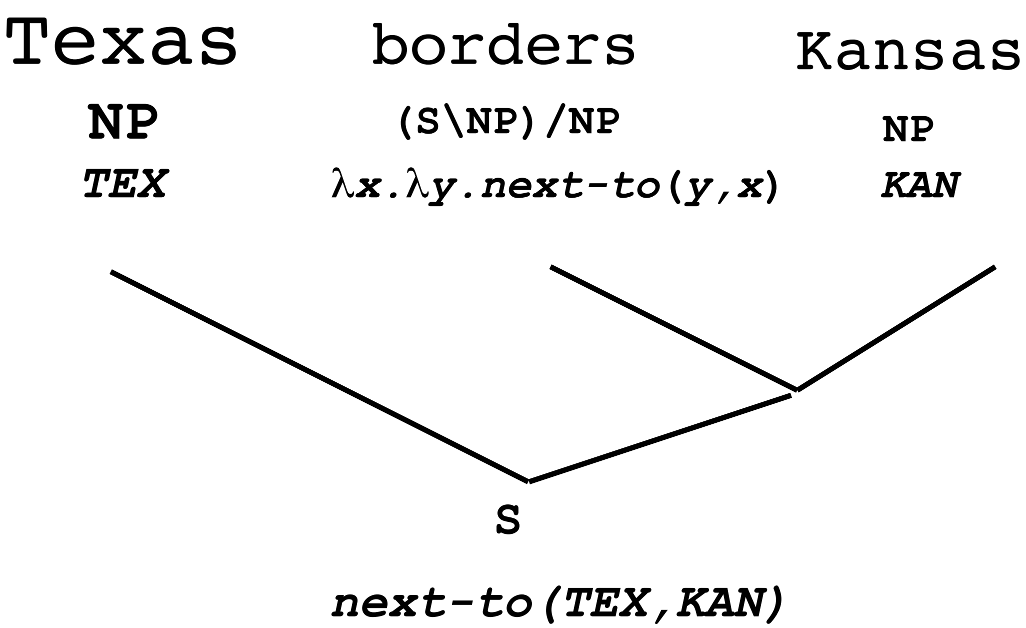
[Steedman 96,00]

Many Meanings: Lexical Ambiguity

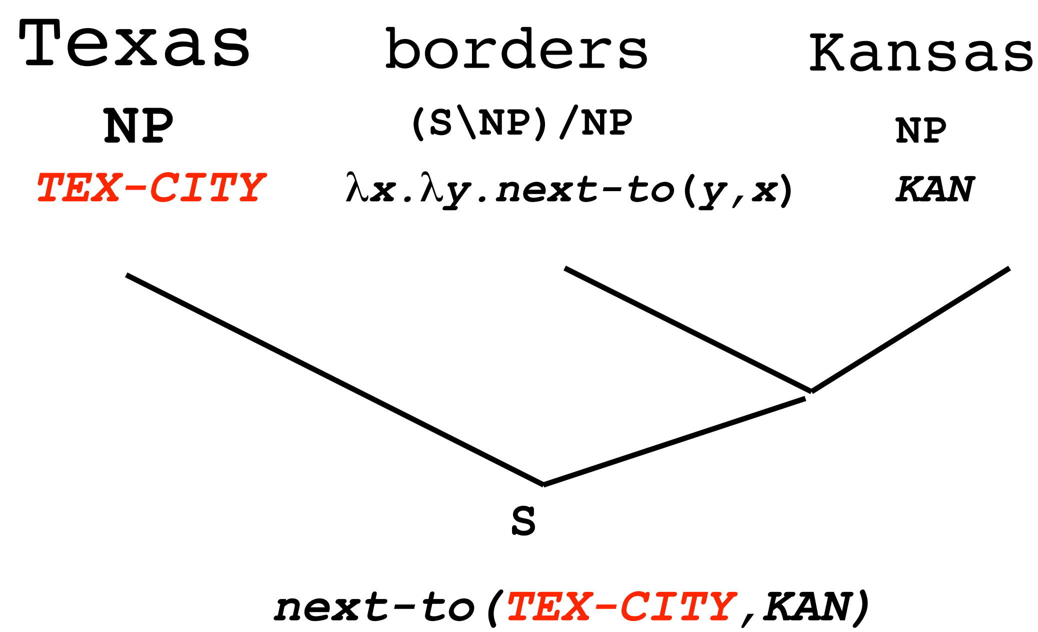
```
Texas borders Kansas
```

Many Meanings: Lexical Ambiguity

Texas borders Kansas



or



Many Meanings: Structural Ambiguity

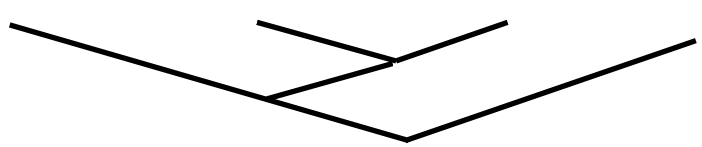
```
flights from Newark or from New York
  that are nonstop
```

Many Meanings: Structural Ambiguity

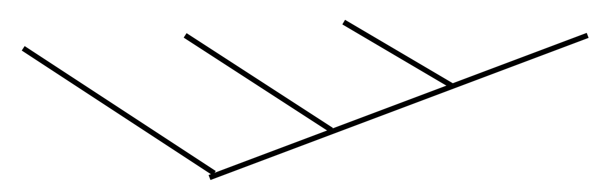
flights from Newark or from New York
that are nonstop

[[flights from Newark or from New York] that are nonstop]

[flights from Newark or [from New York that are nonstop]]



or



$\lambda x.flight(x) \wedge nonstop(x) \wedge (from(x,NEW) \vee from(x,NYC))$

$\lambda x.flight(x) \wedge (from(x,NEW) \vee (from(x,NYC) \wedge nonstop(x)))$

A Supervised Learning Problem

Training Examples:

`What states border Texas?`

$\lambda x.state(x) \wedge next\text{-}to(x, TEX)$

A function f that maps sentences to meaning.

A Multilingual Learning Algorithm

Key challenge: learn from data with different natural languages and meaning representations

English, logical-form:

NL: `what states border texas`
MR: $\lambda x.state(x) \wedge next\_to(x, tex)$

Turkish, functional query language:

NL: `texas a siniri olan eyaletler nelerdir`
MR: $answer(state(next\_to\_2(stateid\ \ tex)))$

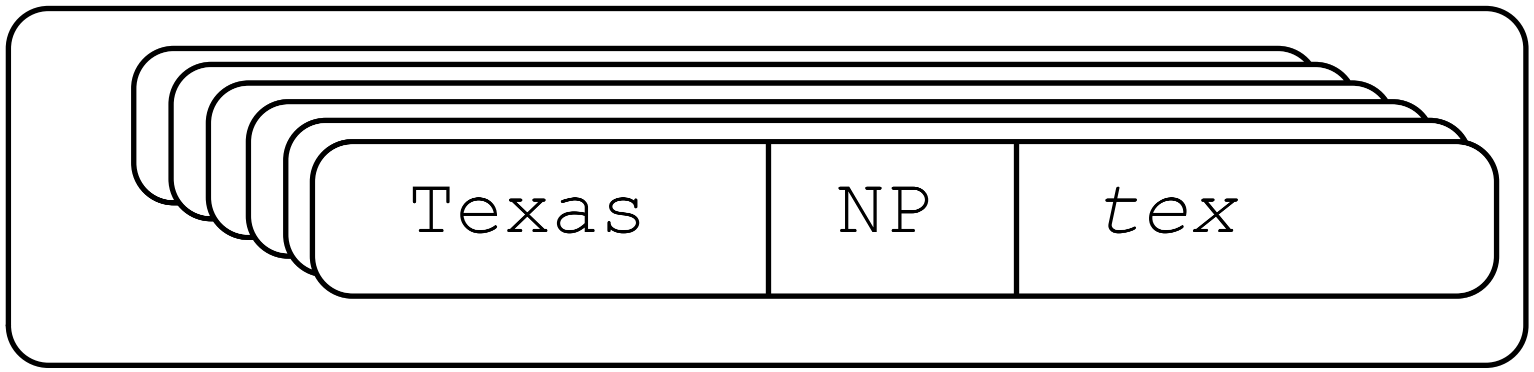
[Kwiatkowski, et al 2010]

Will Lean: Probabilistic CCG

Lexicon:

Parameters:

$\Lambda =$

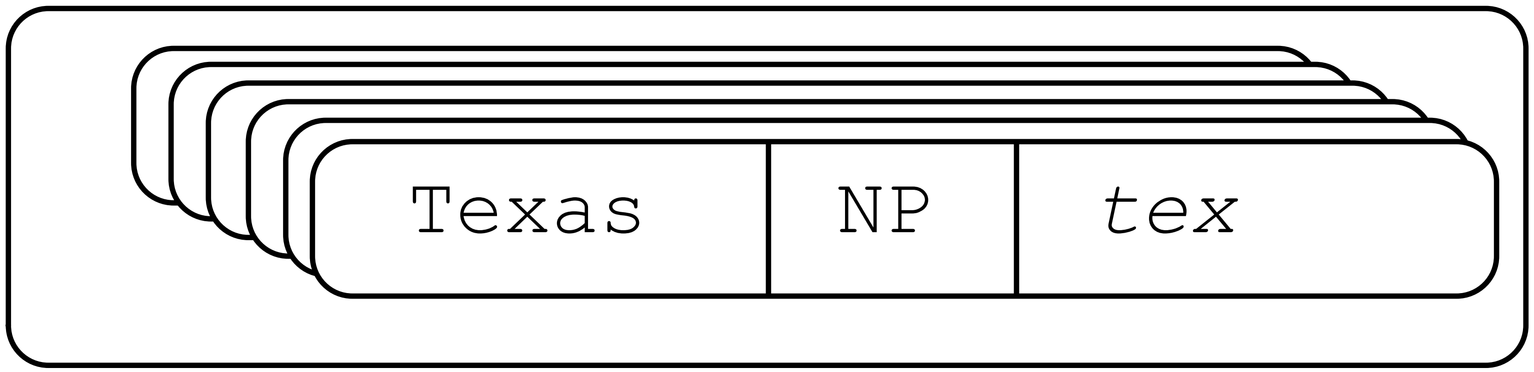


$, \ \theta$

Will Lean: Probabilistic CCG

Lexicon: Parameters:

$\Lambda =$



$, \quad \theta$

Probability distribution: sentence x, parse y, logical form z

- Log-linear model:

$$P(y,z|x;\theta,\Lambda) = \frac{e^{\theta\cdot\phi(x,y,z)}}{\sum_{(y',z')} e^{\theta\cdot\phi(x,y',z')}}$$

- Parsing:

$$f(x) = \arg\max_z p(z|x;\theta,\Lambda)$$

where $$p(z|x;\theta,\Lambda) = \sum_y p(y,z|x;\theta,\Lambda)$$

Splitting lexical items

Initial, Fully Specified Lexical Entries:

`what states border texas` $:=$ S : $\lambda x.state(x) \wedge next\text{-}to(x, tex)$

Splitting lexical items

Initial, Fully Specified Lexical Entries:

$\texttt{what states border texas} := \mathsf{S} : \lambda x.state(x) \wedge next\text{-}to(x,tex)$

Will need to split:

$\texttt{what states} := \mathsf{S/(S|NP)} : \lambda f.\lambda x.state(x) \wedge f(x)$

$\texttt{border texas} := \mathsf{S|NP} : \lambda x.next\text{-}to(x,tex)$

Splitting lexical items

Initial, Fully Specified Lexical Entries:

$\texttt{what states border texas} := \mathsf{S} : \lambda x.state(x) \wedge next\text{-}to(x, tex)$

Will need to split:

$\texttt{what states} := \mathsf{S/(S|NP)} : \lambda f.\lambda x.state(x) \wedge f(x)$

$\texttt{border texas} := \mathsf{S|NP} : \lambda x.next\text{-}to(x, tex)$

Challenge:

Do not have a-priori knowledge of how words align with meaning

$\texttt{texas a siniri olan eyaletler nelerdir} :=$
$\mathsf{S} : \lambda x.state(x) \wedge next\text{-}to(x, tex)$

Algorithm will run on all languages!

Splitting logical forms

Solve a higher-order unification problem [Huet 75]

For logical meaning h find all pairs (f, g) such that:

$h = f(g)$, or - application

$h = \lambda x. f(g(x))$ - composition

$$h = \lambda x.state(x) \wedge next\text{-}to(x, tex)$$

$f = \lambda q \lambda x. q(x)$	$g = \lambda x.state(x) \wedge next\_to(x, tex)$
$f = \lambda q \lambda x. q(x) \wedge next\_to(x, tex)$	$g = \lambda x.state(x)$
$f = \lambda q \lambda x.state(x) \wedge q(x)$	$g = \lambda x.next\_to(x, tex)$
$f = \lambda y \lambda x.state(x) \wedge next\_to(x, y)$	$g = tex$
$f = \lambda q.q$	$g = \lambda x.state(x) \wedge next\_to(x, tex)$

Splitting lexical items

$what \vdash S/(S|NP) : \lambda x \lambda y.x(y)$

$what\ states \vdash S/(S|NP) : \lambda x \lambda y.x(y)$

$what\ states\ border \vdash S/(S|NP) : \lambda x \lambda y.x(y)$

$what \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$what\ states \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$what\ states\ border \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$what \vdash S/(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$what\ states \vdash S/(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$what\ states\ border \vdash S/(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$what \vdash S|NP : \lambda x.state(x)$

$what\ states \vdash S|NP : \lambda x.state(x)$

$what\ states\ border \vdash S|NP : \lambda x.state(x)$

$what \vdash S/(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$what\ states \vdash S/(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$what\ states\ border \vdash S/(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$what \vdash S|NP : \lambda x.next\_to(x\ tex)$

$what\ states \vdash S|NP : \lambda x.next\_to(x\ tex)$

$what\ states\ border \vdash S|NP : \lambda x.next\_to(x\ tex)$

$what \vdash S/NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

$what\ states \vdash S/NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

$what\ states\ border \vdash S/NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

$what \vdash NP : tex$

$what\ states \vdash NP : tex$

$what\ states\ border \vdash NP : tex$

$states\ border\ texas \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$border\ texas \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$texas \vdash S|NP : \lambda x state(x) \wedge next\_to(x\ tex)$

$states\ border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y)$

$border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y)$

$texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y)$

$states\ border\ texas \vdash S|NP : \lambda x.state(x)$

$border\ texas \vdash S|NP : \lambda x.state(x)$

$texas \vdash S|NP : \lambda x.state(x)$

$states\ border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$texas \vdash S\backslash(S|NP) : \lambda x \lambda y.x(y) \wedge next\_to(y\ tex)$

$states\ border\ texas \vdash S|NP : \lambda x.next\_to(x\ tex)$

$border\ texas \vdash S|NP : \lambda x.next\_to(x\ tex)$

$texas \vdash S|NP : \lambda x.next\_to(x\ tex)$

$states\ border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$border\ texas \vdash S\backslash(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$texas \vdash S\backslash(S|NP) : \lambda x \lambda y.state(y) \wedge x(y)$

$states\ border\ texas \vdash NP : tex$

$border\ texas \vdash NP : tex$

$texas \vdash NP : tex$

$states\ border\ texas \vdash S\backslash NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

$border\ texas \vdash S\backslash NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

$texas \vdash S\backslash NP : \lambda x \lambda y.state(y) \wedge next\_to(y\ x)$

Two Step Learning Algorithm

Input:
  Set of (sentence, meaning) pairs

Iterate:
  For each (sentence, meaning) pair

1. Add items to CCG lexicon
2. Update parameters of parsing model

By interleaving step 1 with step 2 we can use the parsing model to guide lexical expansion

Trace of Learning Algorithm

Iteration: **1**
Training pair: (x_n, z_n)

1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.

S

texas a siniri olan eyaletler nelerdir

$\lambda x.state(x) \wedge next\text{-}to(x, tex)$

Trace of Learning Algorithm

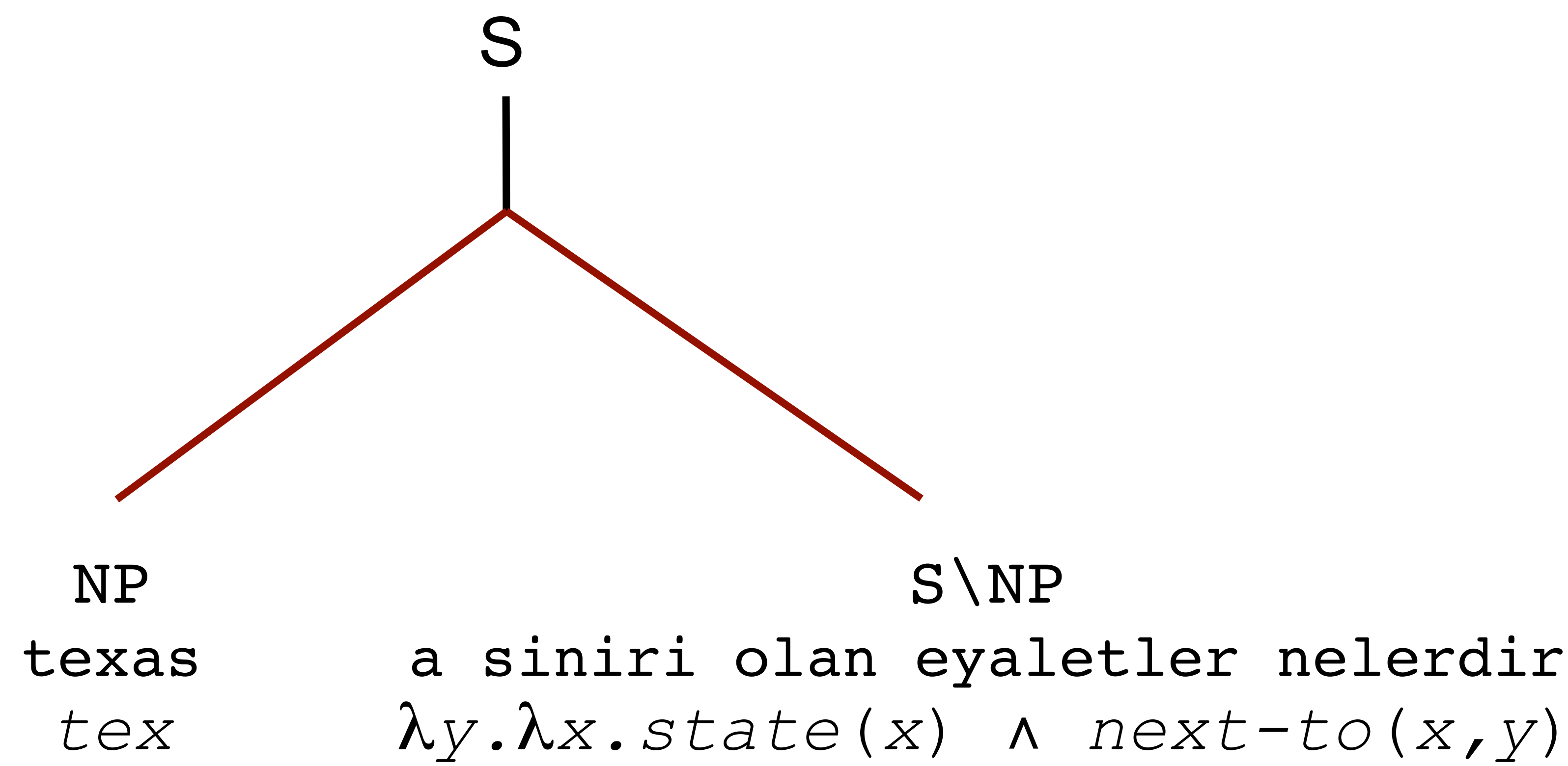
Iteration: **1**
Training pair: (x_n, z_n)

1. Find highest scoring correct parse.

2. Find split, of any node, that most increases the score.

3. Add resultant items to lexicon.

4. Update parameters.



Trace of Learning Algorithm

Iteration: **1**
Training pair: (x_n, z_n)

1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.

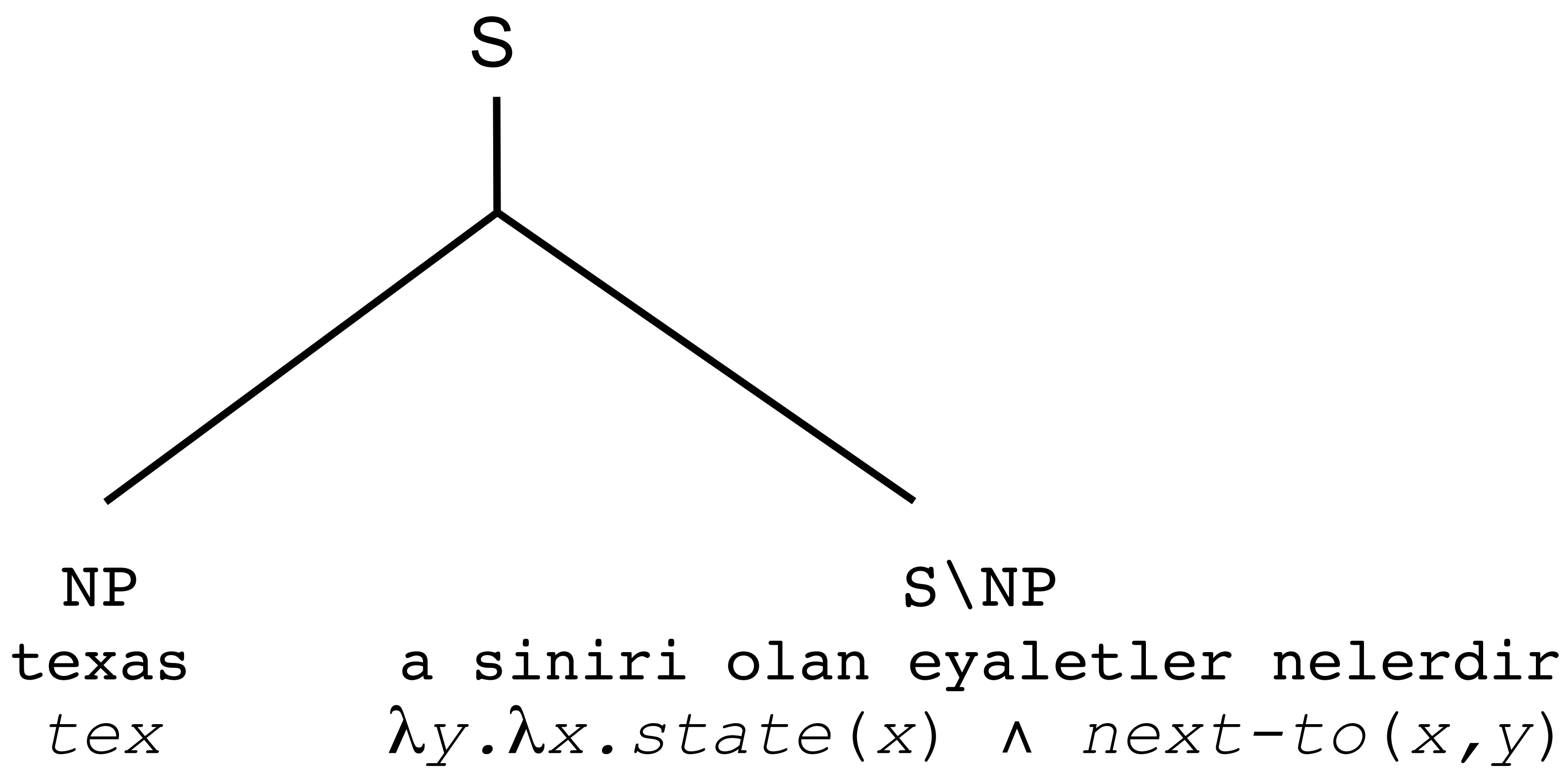
S

NP	S\NP
texas	a siniri olan eyaletler nelerdir
tex	$\lambda y.\lambda x.state(x) \wedge next\text{-}to(x,y)$

Trace of Learning Algorithm

Iteration: **1**
Training pair: (x_n, z_n)

1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.

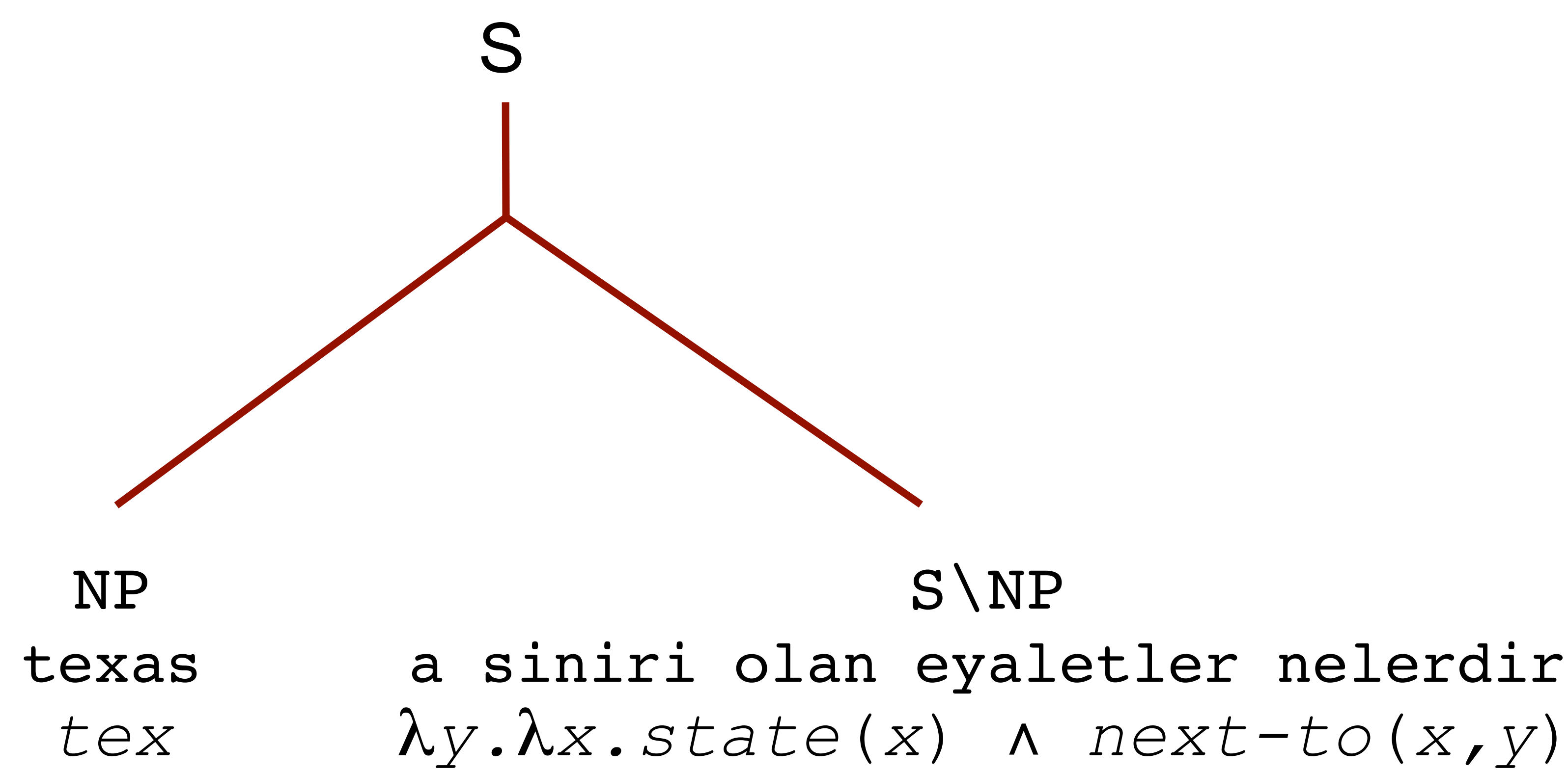


$$\frac{\partial O_i}{\partial \theta_j} = E_{p(y|x_i,z_i;\theta,\Lambda)}[\phi_j(x_i, y, z_i)] - E_{p(y,z|x_i;\theta,\Lambda)}[\phi_j(x_i, y, z)]$$

Trace of Learning Algorithm

Iteration: **2**
Training pair: (x_n, z_n)

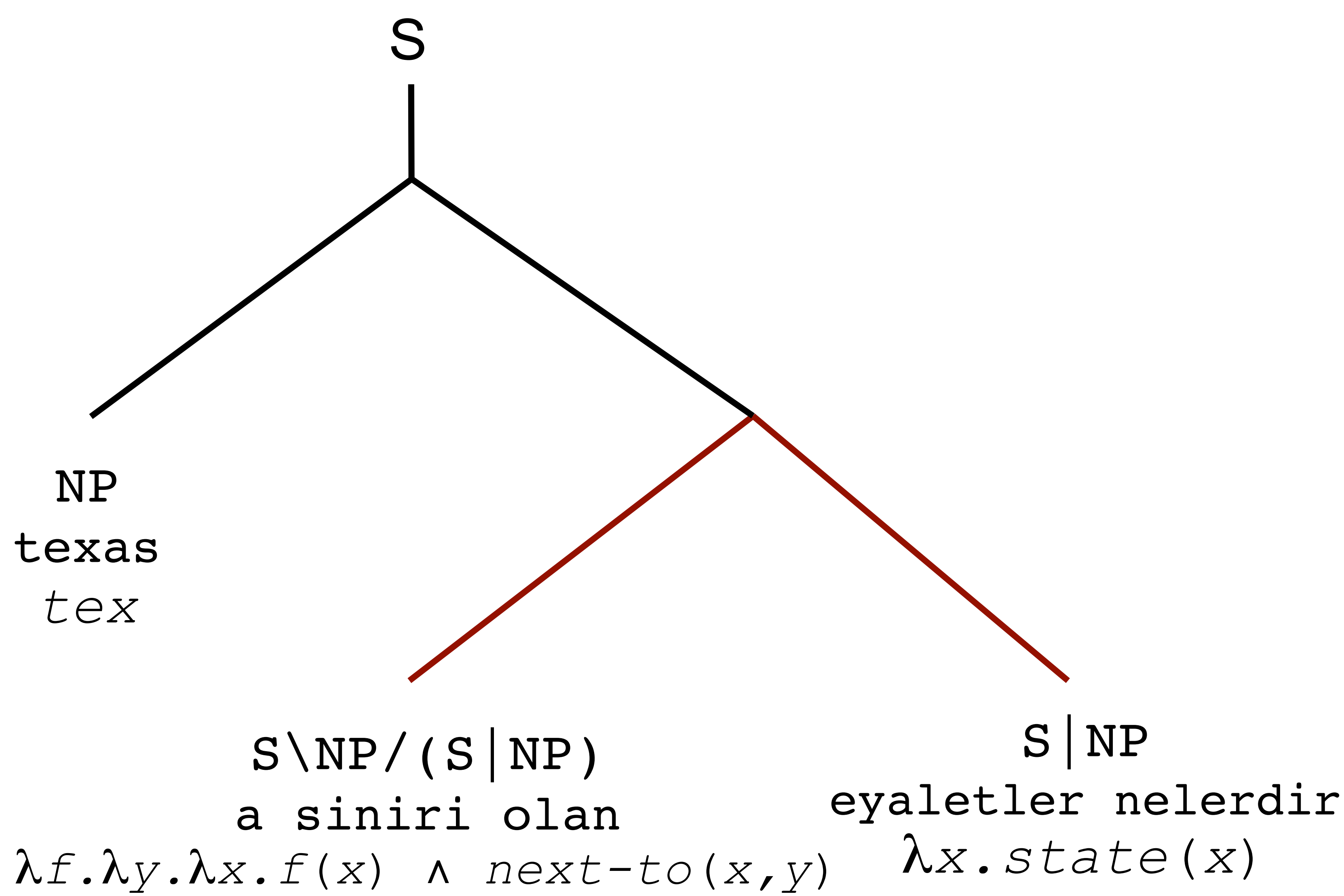
1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.



Trace of Learning Algorithm

Iteration: **2**
Training pair: (x_n, z_n)

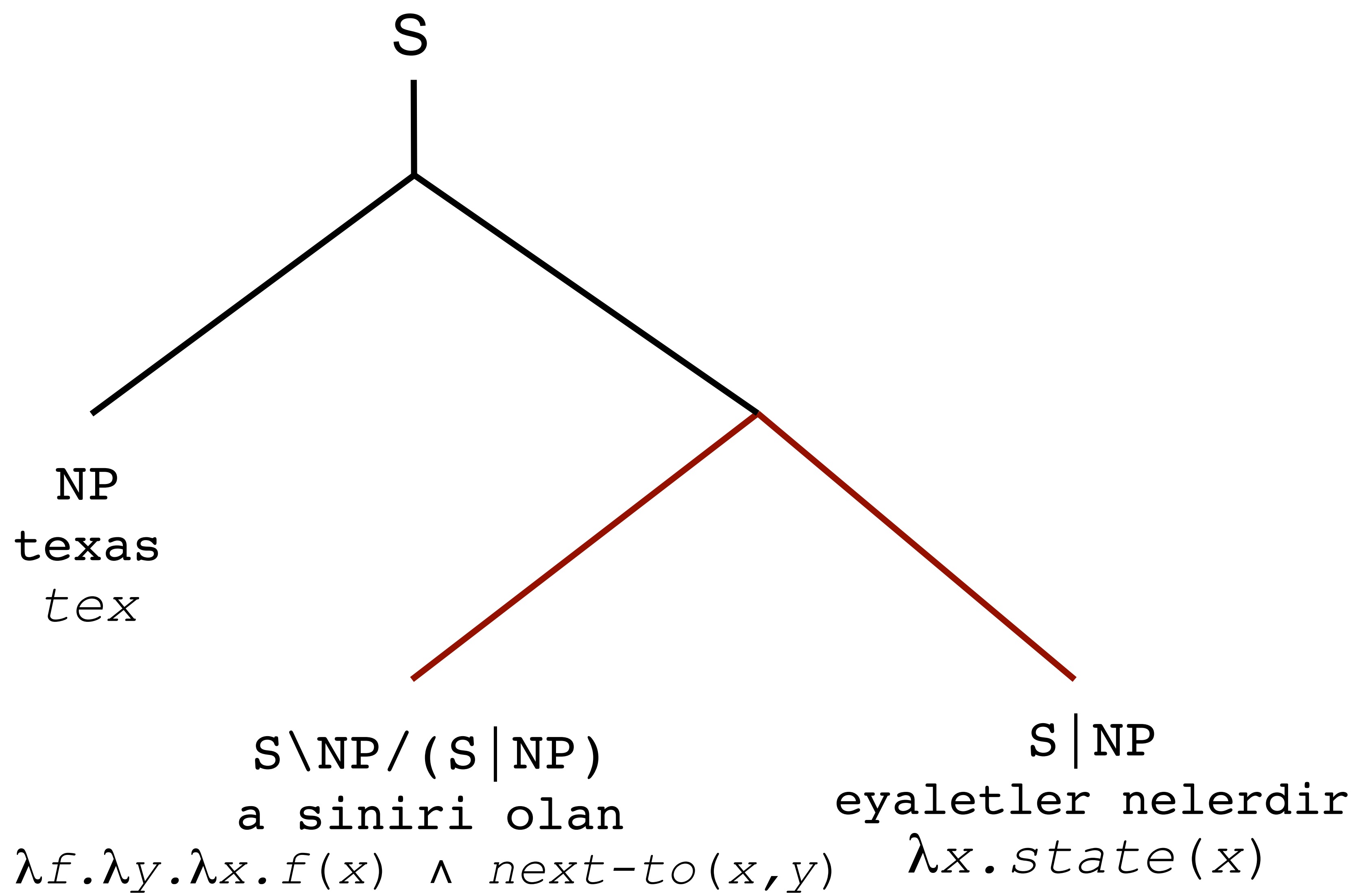
1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.



Trace of Learning Algorithm

Iteration: **2**
Training pair: (x_n, z_n)

1. Find highest scoring correct parse.
2. Find split, of any node, that most increases the score.
3. Add resultant items to lexicon.
4. Update parameters.



Results on an English Benchmark

Accuracy (% correct)

FOPL		FunQL	
UBL	**87.9**	UBL	**84.3**
λ-WASP	86.6	WASP	74.8
ZC05	79.3	Lu08	81.5
ZC07	86.1	KRISP	71.7

[Kwiatkowski et al. 2010]

Results Across Languages

Accuracy (% correct)

	FOPL		FunQL		
	UBL	λ-WASP	UBL	WASP	Lu08
English	**81.8**	75.6	**80.4**	70.0	72.8
Spanish	**81.4**	80.0	**79.7**	72.4	79.2
Japanese	**83.0**	81.2	**80.5**	74.4	76.0
Turkish	**71.8**	68.8	**74.2**	62.4	66.8

[Kwiatkowski et al. 2010]

Example Test Parses

which rivers run through states that border the state with the capital austin

Example Test Parses

which rivers run through states that border the state with the capital austin

$$\frac{\qquad\qquad\qquad\qquad}{\begin{array}{c}\mathtt{NP} \\ \iota x.state(x) \wedge capital(x,aus)\end{array}}$$

Example Test Parses

which rivers run through states that border the state with the capital austin

NP

$\iota x.state(x) \wedge capital(x,aus)$

NP

$\lambda y.state(y) \wedge$
$next\text{-}to(y,\iota x.state(x) \wedge capital(x,aus))$

Example Test Parses

which rivers run through states that border the state with the capital austin

NP

$\iota x.state(x) \wedge capital(x,aus)$

NP

$\lambda y.state(y) \wedge$
$next\text{-}to(y, \iota x.state(x) \wedge capital(x,aus))$

S

$\lambda z.river(z) \wedge \exists y.state(y) \wedge loc(z,y) \wedge$
$next\text{-}to(y, \iota x.state(x) \wedge capital(x,aus))$

Learning Summary

```
Show me flights from Newark and New York to San Francisco
   or Oakland that are nonstop.
```

$\lambda x.flight(x) \wedge nonstop(x) \wedge (from(x,NEW) \vee from(x,NYC)) \wedge$
$(to(x,SFO) \vee to(x,OAK))$

Challenges:

- Structured input and output, hidden structure not annotated

Learning Summary

```
Show me flights from Newark and New York to San Francisco
   or Oakland that are nonstop.
```

$\lambda x.flight(x) \wedge nonstop(x) \wedge (from(x,NEW) \vee from(x,NYC)) \wedge$
$(to(x,SFO) \vee to(x,OAK))$

Challenges:

- Structured input and output, hidden structure not annotated

Solution:

- Machine learning combined with a detailed linguistic formalism
- Key idea 1: an algorithm that generates lexical items with higher-order unification
- Key idea 2: a learning method that estimates a probabilistic model to select a sparse sub-set of the possible lexical entries

Previous Work: Spontaneous, Unedited Input

`Boston to Seattle the latest on Friday.`

$$\mathit{argmax}(\ \lambda x.\mathit{from}(x,BOS) \wedge \mathit{to}(x,SEA) \wedge \mathit{day}(x,FRI),\ \lambda y.\mathit{time}(y))$$

A challenge: grammar-based approaches may not be robust to "real world" input

Key idea: learn a relaxed grammar that models disfluencies

[Zettlemoyer & Collins, 2007; Kwiatkowski et al. 2011]

Previous Work: Context-dependent

Show me flights from New York to Singapore.

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN)$

Which of those are nonstop?

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land nonstop(x)$

Show me the cheapest one.

$argmax(\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land nonstop(x), \lambda y.cost(y))$

What about connecting?

$argmax(\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land connect(x), \lambda y.cost(y))$

[Zettlemoyer & Collins, 2009]

Current Work: Self-learning dialog systems

Challenge: system must learn from its mistakes

USER: i would like to fly from atlanta georgia to london england on september twenty fourth in the early evening I would like to return on october first departing from london in the late morning

SYSTEM: leaving what city? *(ASK:$\lambda x.from(fl, x)$)*

USER: atlanta georgia.

SYSTEM: leaving atlanta. *(CONF:$from(fl,ATL)$)* going to which city ? *(ASK:$\lambda x.to(fl, x)$)*

USER london

SYSTEM: arriving in *(CONF:$to(fl, LON)$)* what date would to depart Atlanta? *(ASK:$\lambda x.from(fl, ATL) \wedge depart\text{-} date(fl, x)$)*

USER september twenty fourth in the early evening

[conversation continues]

[Artzi & Zettlemoyer, 2011]

Future: Mapping Speech to Logical From

```
Uh, flights New York to Singapore, sure
```

ACCEPT: $\lambda x.\mathit{flight}(x) \wedge \mathit{from}(x,\mathit{NYC}) \wedge \mathit{to}(x,\mathit{SIN})$

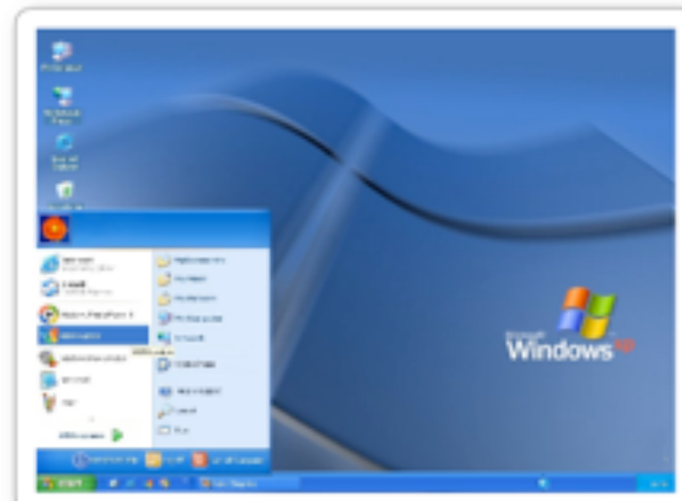
Previous Work: Mapping Instructions to Actions

Input

Instructions:
step-by-step
descriptions of actions

1. Click **Start**, point to **Search**, and then click **For Files or Folders.**
2. In the **Search for** box, type "msdownld.tmp"
3. In the **Look in** list, click **My Computer**, and then click **Search Now**.
4. ...

Target environment:
where actions
need to be executed



Output

Action sequence
executable in the environment

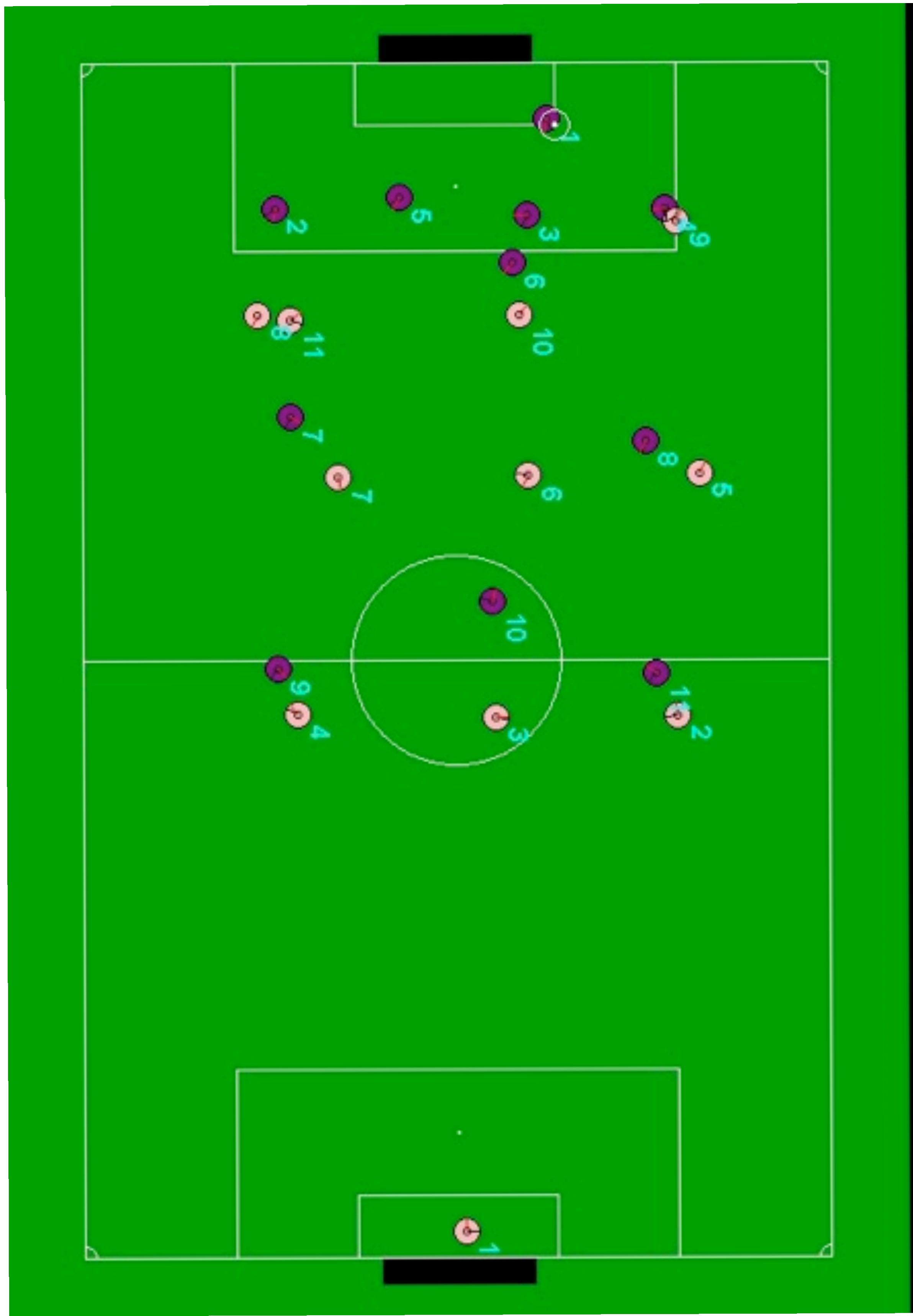
```
LEFT_CLICK( Start )
LEFT_CLICK( Search )
...
TYPE_INTO( Search for: , "msdownld.tmp")
...
```

[Branavan et al, 2009]

Current Work: Leaning Grounded Language

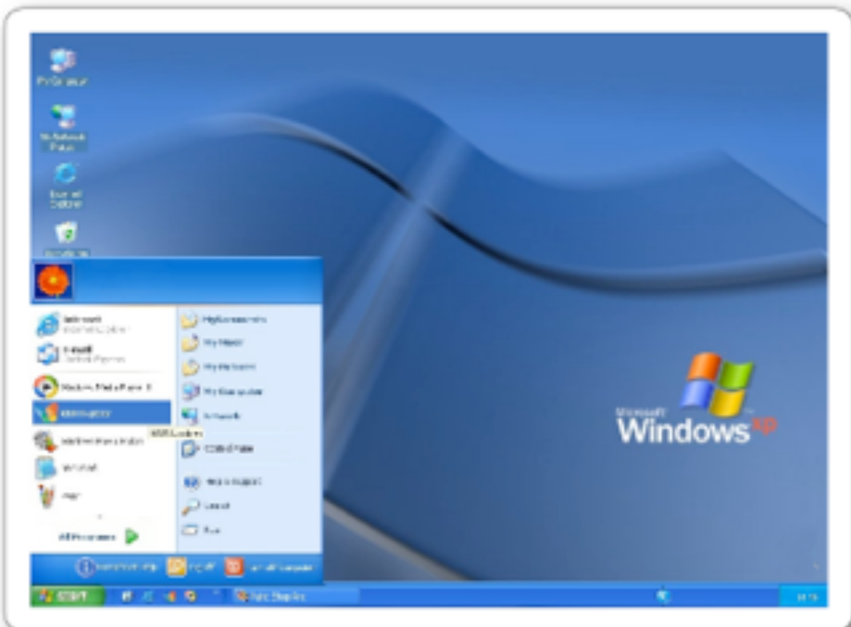
Challenge: Learn to sportscast, given only text and the game log

```
Purple10 is rushing down the
   field with only three
   defenders
Purple10 passes out front to
   Purple9 near the side
Purple9 passes back to Purple10
   in the middle
Purple10 again has a good chance
   to score a goal here
Purple10 dribbles toward the
   goal
Pink3 tries to stay in front of
   Purple10
Purple10 passes to Purple9 on
   the side while getting open
....
```



Future: General language use in grounded settings

Conversational interaction in simulated environments:



- Can gather user input: *Which printer do you want to use?*

- Can help with learning: *Can you show me how to X?*

Learning through explanation in robotic environments:

Can we teach the robot to play?

- *This is a pawn.*
- *Pawns can move forward one square at a time.*
- *unless it is the first move, then they can …*

Learning Map Sentences to Meaning

special thanks to

Yoav Artzi, Regina Barzilay, Branavan, Michael Collins, Sharon Goldwater, Raphael Hoffman, Tom Kwiatkowski, Mark Steedman, Dan Weld, Adrienne Wang, Mark Yatskar

for more info:

http://www.cs.washington.edu/homes/lsz/