

Semantic Parsing with Combinatory Categorial Grammars

Yoav Artzi, Nicholas FitzGerald and Luke Zettlemoyer
University of Washington

ACL 2013 Tutorial
Sofia, Bulgaria



Language to Meaning



More informative

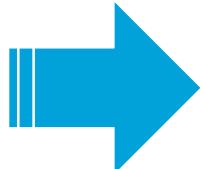
Language to Meaning

Information Extraction

Recover information about pre-specified relations and entities

Example Task

Relation Extraction



More informative

$is_a(OBAMA, PRESIDENT)$

Language to Meaning

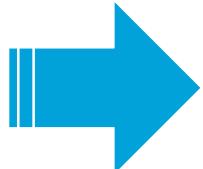
Broad-coverage
Semantics

Focus on specific
phenomena (e.g., verb-
argument matching)

More informative

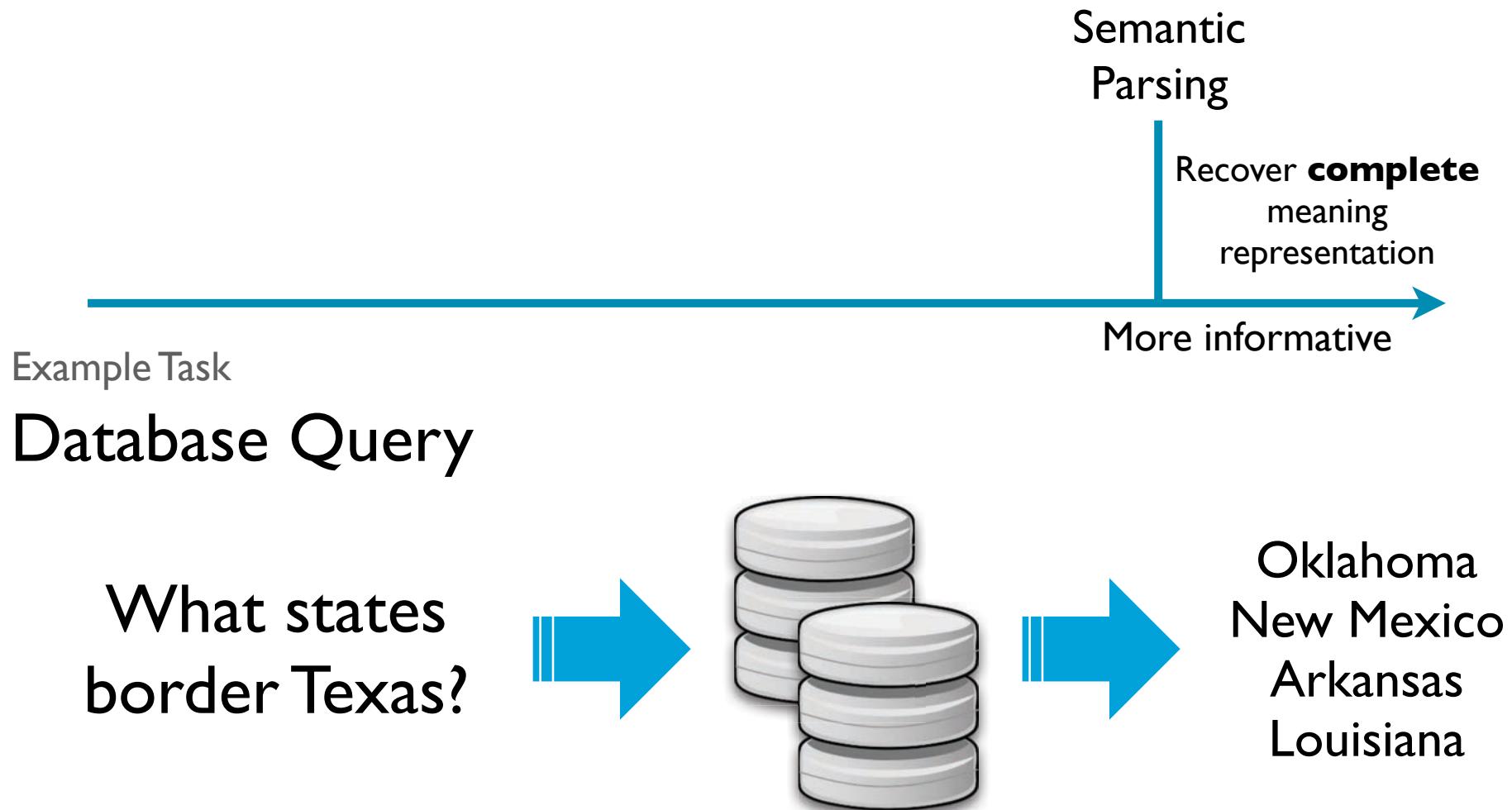
Example Task

Summarization



Obama wins
election. Big party
in Chicago.
Romney a bit
down, asks for
some tea.

Language to Meaning

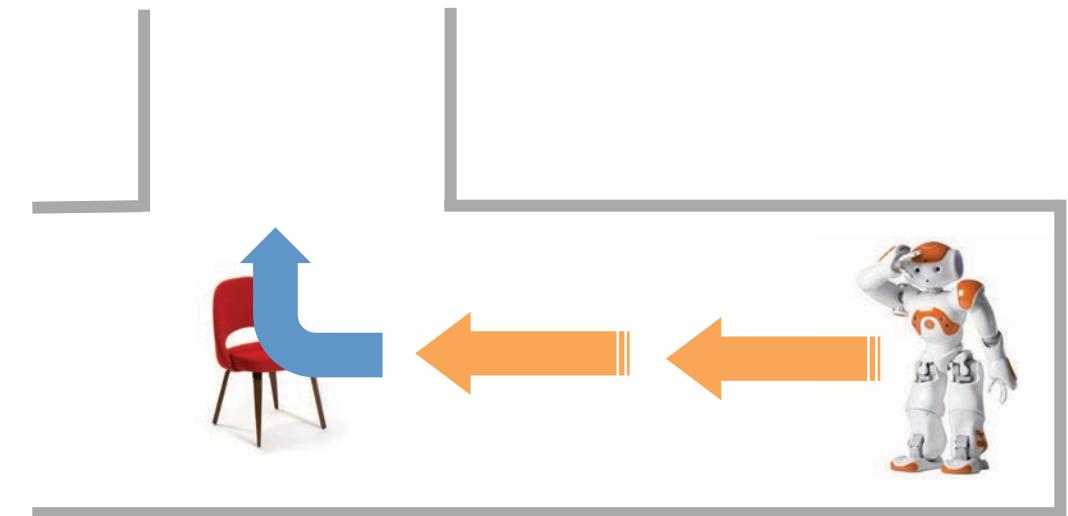


Language to Meaning

Example Task

Instructing a Robot

at the chair,
turn right



Language to Meaning



Complete meaning is sufficient to complete the task

- Convert to database query to get the answer
- Allow a robot to do planning

Language to Meaning



at the chair, move forward three steps past the sofa

$$\lambda a. pre(a, \iota x. chair(x)) \wedge move(a) \wedge len(a, 3) \wedge dir(a, forward) \wedge past(a, \iota y. sofa(y))$$

Language to Meaning



at the chair, move forward three steps past the sofa

$$\lambda a. \text{pre}(a, \iota x. \text{chair}(x)) \wedge \text{move}(a) \wedge \text{len}(a, 3) \wedge \\ \text{dir}(a, \text{forward}) \wedge \text{past}(a, \iota y. \text{sofa}(y))$$

Language to Meaning

at the chair, move forward three steps past the sofa

$$\lambda a. \text{pre}(a, \iota x. \text{chair}(x)) \wedge \text{move}(a) \wedge \text{len}(a, 3) \wedge \\ \text{dir}(a, \text{forward}) \wedge \text{past}(a, \iota y. \text{sofa}(y))$$


$f : \text{sentence} \rightarrow \text{logical form}$

Language to Meaning

at the chair, move forward three steps past the sofa



$f : \text{sentence} \rightarrow \text{logical form}$

Central Problems

Parsing

Learning

Modeling

Parsing Choices

- Grammar formalism
- Inference procedure

Inductive Logic Programming [Zelle and Mooney 1996]

SCFG [Wong and Mooney 2006]

CCG + CKY [Zettlemoyer and Collins 2005]

Constrained Optimization + ILP [Clarke et al. 2010]

DCS + Projective dependency parsing [Liang et al. 2011]

Learning

- What kind of supervision is available?
- Mostly using latent variable methods

Annotated parse trees [Miller et al. 1994]

Sentence-LF pairs [Zettlemoyer and Collins 2005]

Question-answer pairs [Clarke et al. 2010]

Instruction-demonstration pairs [Chen and Mooney 2011]

Conversation logs [Artzi and Zettlemoyer 2011]

Visual sensors [Matuszek et al. 2012a]

Semantic Modeling

- What logical language to use?
- How to model meaning?

Variable free logic [Zelle and Mooney 1996; Wong and Mooney 2006]

High-order logic [Zettlemoyer and Collins 2005]

Relational algebra [Liang et al. 2011]

Graphical models [Tellex et al. 2011]

Today

Parsing

Combinatory Categorial Grammars

Learning

Unified learning algorithm

Modeling

Best practices for semantics design

Parsing

Learning

Modeling

Parsing

Learning

Modeling

- Lambda calculus
- Parsing with Combinatory Categorial Grammars
- Linear CCGs
- Factored lexicons

Parsing

Learning

Modeling

- Structured perceptron
- A unified learning algorithm
- Supervised learning
- Weak supervision

Parsing

Learning

Modeling

- Semantic modeling for:
 - Querying databases
 - Referring to physical objects
 - Executing instructions

UW SPF

Open source semantic parsing framework

<http://yoavartzi.com/spf>

Semantic
Parser

Flexible High-Order
Logic Representation

Learning
Algorithms

Includes ready-to-run examples

Parsing

Learning

Modeling

- Lambda calculus
- Parsing with Combinatory Categorial Grammars
- Linear CCGs
- Factored lexicons

Lambda Calculus

- Formal system to express computation
- Allows high-order functions

$$\lambda a.\text{move}(a) \wedge \text{dir}(a, \text{LEFT}) \wedge \text{to}(a, \iota y.\text{chair}(y)) \wedge \\ \text{pass}(a, \mathcal{A}y.\text{sofa}(y) \wedge \text{intersect}(\mathcal{A}z.\text{intersection}(z), y))$$

Lambda Calculus

Base Cases

- Logical constant
- Variable
- Literal
- Lambda term

Lambda Calculus

Logical Constants

- Represent objects in the world

NYC, CA, RAINIER, LEFT, ...
located_in, depart_date, ...

Lambda Calculus

Variables

- Abstract over objects in the world
- Exact value not pre-determined

x, y, z, \dots

Lambda Calculus

Literals

- Represent function application

city(AUSTIN)

located_in(AUSTIN, TEXAS)

Lambda Calculus

Literals

- Represent function application

city(AUSTIN)

located_in(AUSTIN, TEXAS)

Predicate

Arguments

Logical expression

List of logical expressions

Lambda Calculus

Lambda Terms

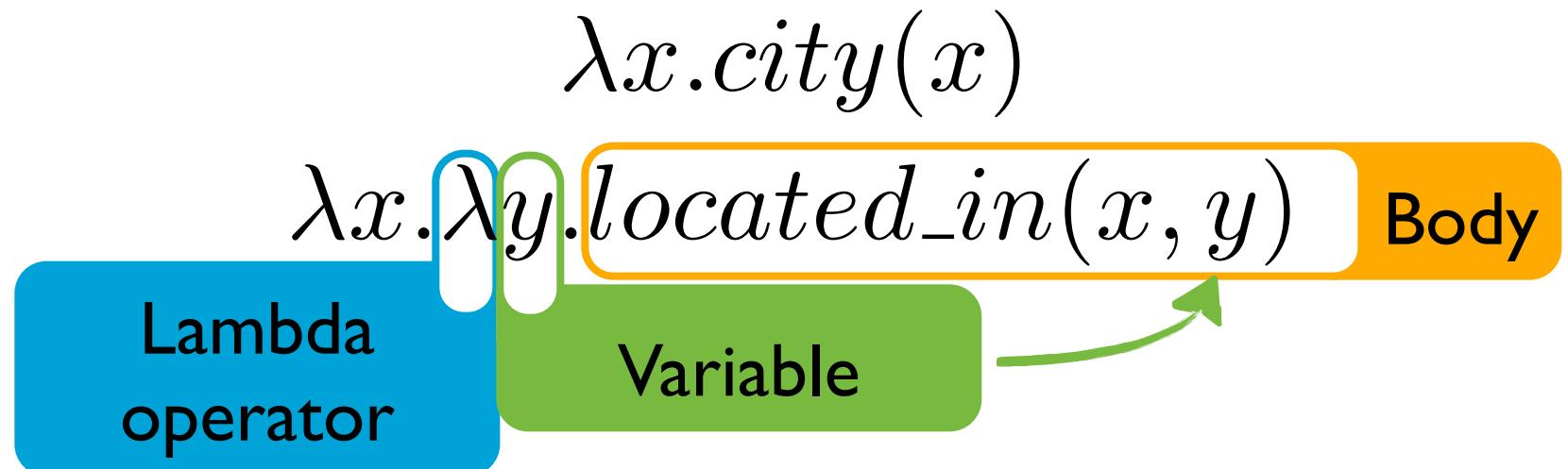
- Bind/scope a variable
- Repeat to bind multiple variables

$$\lambda x.\text{city}(x)$$
$$\lambda x.\lambda y.\text{located_in}(x, y)$$

Lambda Calculus

Lambda Terms

- Bind/scope a variable
- Repeat to bind multiple variables



Lambda Calculus

Quantifiers?

- Higher order constants
- No need for any special mechanics
- Can represent all of first order logic

$$\forall(\lambda x.\text{big}(x) \wedge \text{apple}(x))$$
$$\neg(\exists(\lambda x.\text{lovely}(x)))$$
$$\iota(\lambda x.\text{beautiful}(x) \wedge \text{grammar}(x))$$

Lambda Calculus

Syntactic Sugar

$$\wedge(A, \wedge(B, C)) \Leftrightarrow A \wedge B \wedge C$$

$$\vee(A, \vee(B, C)) \Leftrightarrow A \vee B \vee C$$

$$\neg(A) \Leftrightarrow \neg A$$

$$Q(\lambda x. f(x)) \Leftrightarrow Qx. f(x)$$

for $Q \in \{\iota, \mathcal{A}, \exists, \forall\}$

$$\lambda x. flight(x) \wedge to(x, move)$$
$$\lambda x. flight(x) \wedge to(x, NYC)$$
$$\lambda x. NYC(x) \wedge x(to, move)$$

 $\lambda x. flight(x) \wedge to(x, move)$

 $\lambda x. flight(x) \wedge to(x, NYC)$

 $\lambda x. NYC(x) \wedge x(to, move)$

Simply Typed Lambda Calculus

- Like lambda calculus
- But, typed

 $\lambda x. flight(x) \wedge to(x, move)$

 $\lambda x. flight(x) \wedge to(x, NYC)$

 $\lambda x. NYC(x) \wedge x(to, move)$

Lambda Calculus

Typing

t Truth-value
 e Entity

- Simple types
- Complex types

$$\langle e, t \rangle$$
$$\langle\langle e, t \rangle, e \rangle$$

Lambda Calculus

Typing

t Truth-value
 e Entity

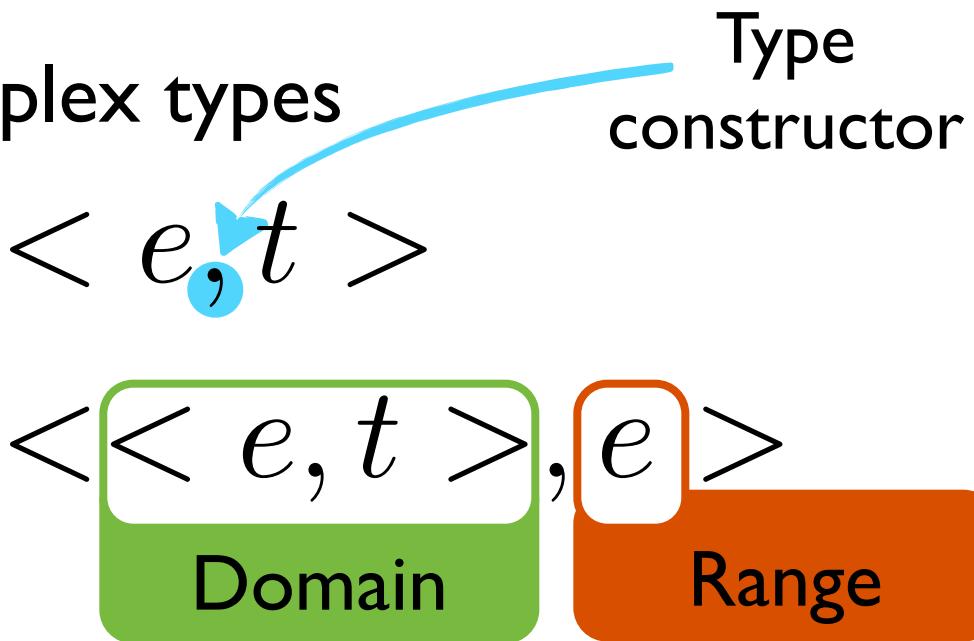
- Simple types
 - Complex types
- $\langle e, t \rangle$
- Type constructor
- $\langle \langle e, t \rangle, e \rangle$
- Domain Range

Lambda Calculus

Typing

t
 e
—
 tr
—
 loc
—

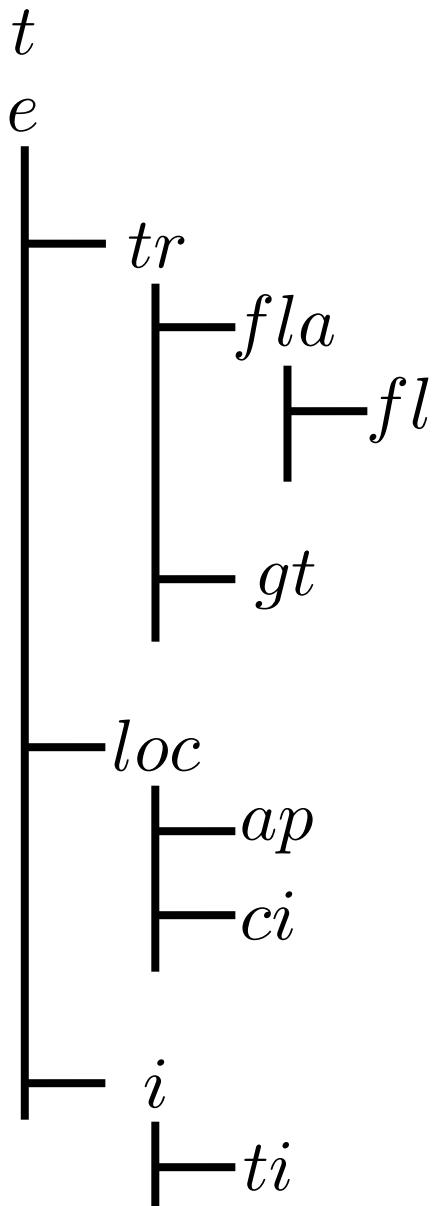
- Simple types
- Complex types



- Hierarchical typing system

Lambda Calculus

Typing



- Simple types
 - Complex types
 - Hierarchical typing system
- $\langle e, t \rangle$
- Type constructor
- $\langle \langle e, t \rangle, e \rangle$
- Domain Range

Simply Typed Lambda Calculus

$$\lambda a.\text{move}(a) \wedge \text{dir}(a, \text{LEFT}) \wedge \text{to}(a, \iota y.\text{chair}(y)) \wedge \\ \text{pass}(a, \mathcal{A}y.\text{sofa}(y) \wedge \text{intersect}(\mathcal{A}z.\text{intersection}(z), y))$$

Type information usually omitted

Capturing Meaning with Lambda Calculus

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV
CA	AZ

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA
Wrangel	AK
Silk	
Rocky	

Show me mountains in states
bordering Texas



[Zettlemoyer and Collins 2005]

Capturing Meaning with Lambda Calculus

SYSTEM how can I help you ?

USER i ' d like to fly to new york

SYSTEM flying to new york . leaving what city ?

USER from boston on june seven with american airlines

SYSTEM flying to new york . what date would you like to depart boston ?

USER june seventh

SYSTEM do you have a preferred airline ?

USER american airlines

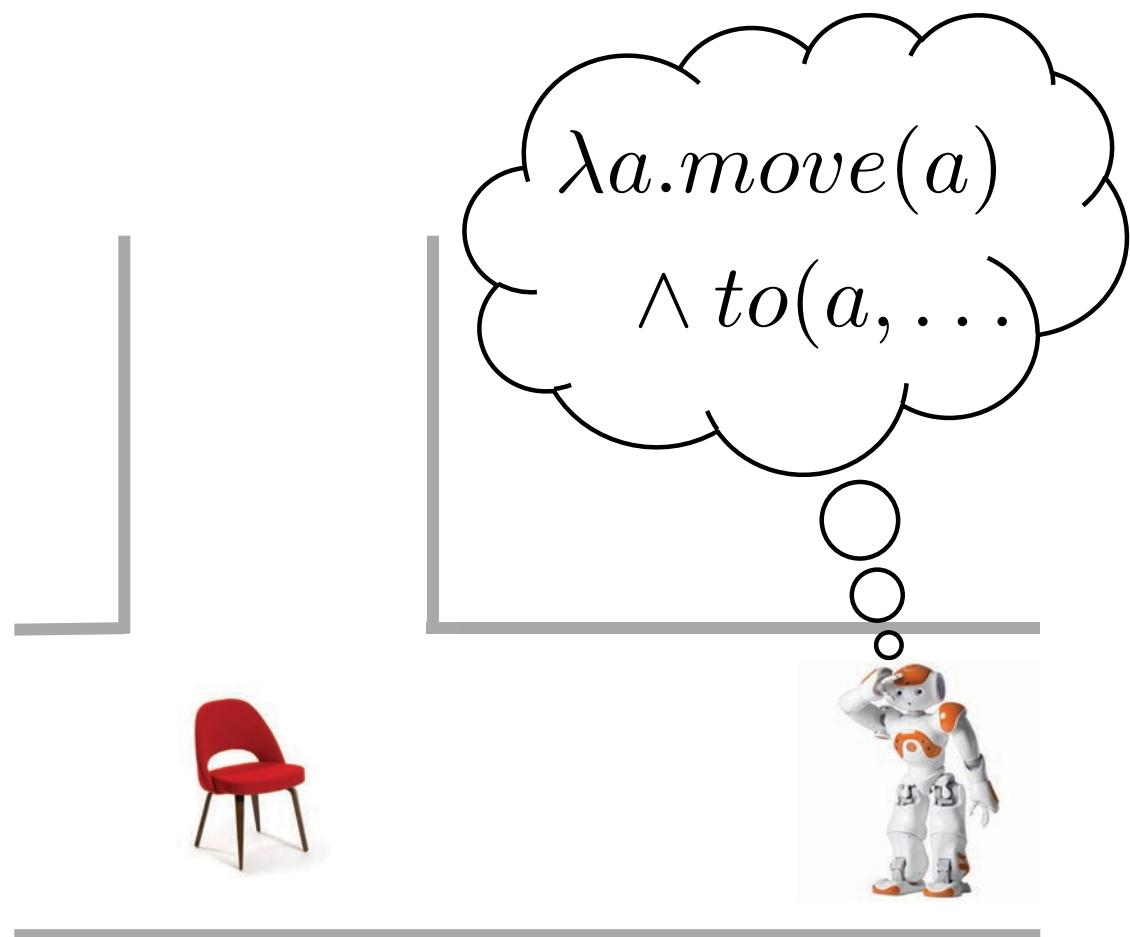
SYSTEM o . k . leaving boston to new york on june seventh flying with american airlines . where would you like to go to next ?

USER back to boston on june tenth

[CONVERSATION CONTINUES]

Capturing Meaning with Lambda Calculus

go to the chair
and turn right



Capturing Meaning with Lambda Calculus

- Flexible representation
- Can capture full complexity of natural language

More on modeling meaning later

Constructing Lambda Calculus Expressions

at the chair, move forward three steps past the sofa


$$\lambda a. \text{pre}(a, \iota x. \text{chair}(x)) \wedge \text{move}(a) \wedge \text{len}(a, 3) \wedge \\ \text{dir}(a, \text{forward}) \wedge \text{past}(a, \iota y. \text{sofa}(y))$$

Combinatory Categorial Grammars

$$\begin{array}{c} \text{CCG} \qquad \text{is} \qquad \text{fun} \\ \hline NP \qquad S \setminus NP / ADJ \qquad ADJ \\ CCG \qquad \lambda f. \lambda x. f(x) \qquad \lambda x. fun(x) \\ \hline \qquad \qquad \qquad \longrightarrow \\ \qquad \qquad \qquad S \setminus NP \\ \qquad \qquad \qquad \lambda x. fun(x) \\ \hline \qquad \qquad \qquad \longleftarrow \\ \qquad \qquad \qquad \overset{S}{fun(CCG)} \end{array}$$

Combinatory Categorial Grammars

- Categorial formalism
- Transparent interface between syntax and semantics
- Designed with computation in mind
- Part of a class of mildly context sensitive formalisms (e.g., TAG, HG, LIG) [Joshi et al. 1990]

CCG Categories

$ADJ : \lambda x. fun(x)$

- Basic building block
- Capture syntactic and semantic information jointly

CCG Categories

Syntax

ADJ

$\lambda x. fun(x)$

Semantics

- Basic building block
- Capture syntactic and semantic information jointly

CCG Categories

Syntax

$$ADJ : \lambda x. fun(x)$$
$$(S \setminus NP) / ADJ : \lambda f. \lambda x. f(x)$$
$$NP : CCG$$

- Primitive symbols: N, S, NP, ADJ and PP
- Syntactic combination operator (/,\)
- Slashes specify argument order and direction

CCG Categories

$ADJ : \lambda x. fun(x)$ Semantics

$(S \setminus NP) / ADJ : \lambda f. \lambda x. f(x)$

$NP : CCG$

- λ -calculus expression
- Syntactic type maps to semantic type

CCG Lexical Entries

$\text{fun} \vdash ADJ : \lambda x. \text{fun}(x)$

- Pair words and phrases with meaning
- Meaning captured by a CCG category

CCG Lexical Entries

fun

Natural
Language

$ADJ : \lambda x. fun(x)$

CCG Category

- Pair words and phrases with meaning
- Meaning captured by a CCG category

CCG Lexicons

$\text{fun} \vdash ADJ : \lambda x. \text{fun}(x)$

is $\vdash (S \setminus NP) / ADJ : \lambda f. \lambda x. f(x)$

$\text{CCG} \vdash NP : CCG$

- Pair words and phrases with meaning
- Meaning captured by a CCG category

Between CCGs and CFGs

	CFGs	CCGs
Combination operations	Many	Few
Parse tree nodes	Non-terminals	Categories
Syntactic symbols	Few dozen	Handful, but can combine
Paired with words	POS tags	Categories

Parsing with CCGs

$$\begin{array}{ccc} \text{CCG} & \text{is} & \text{fun} \\ \hline NP & \overline{S \setminus NP / ADJ} & \overline{ADJ} \\ CCG & \lambda f. \lambda x. f(x) & \lambda x. fun(x) \end{array}$$

Use lexicon to match words and
phrases with their categories

CCG Operations

- Small set of operators
 - Input: 1-2 CCG categories
 - Output: A single CCG category
- Operate on syntax semantics together
- Mirror natural logic operations

CCG Operations

Application

$$B : g \quad A \setminus B : f \Rightarrow A : f(g) \quad (<)$$

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

- Equivalent to function application
- Two directions: forward and backward
 - Determined by slash direction

CCG Operations

Application

Argument	Function	Result	
$B : g$	$A \setminus B : f$	$\Rightarrow A : f(g)$	(<)
$A/B : f$	$B : g$	$\Rightarrow A : f(g)$	(>)

- Equivalent to function application
- Two directions: forward and backward
 - Determined by slash direction

Parsing with CCGs

$$\begin{array}{ccc} \text{CCG} & \text{is} & \text{fun} \\ \hline NP & \overline{S \setminus NP / ADJ} & \overline{ADJ} \\ CCG & \lambda f. \lambda x. f(x) & \lambda x. fun(x) \end{array}$$

Use lexicon to match words and
phrases with their categories

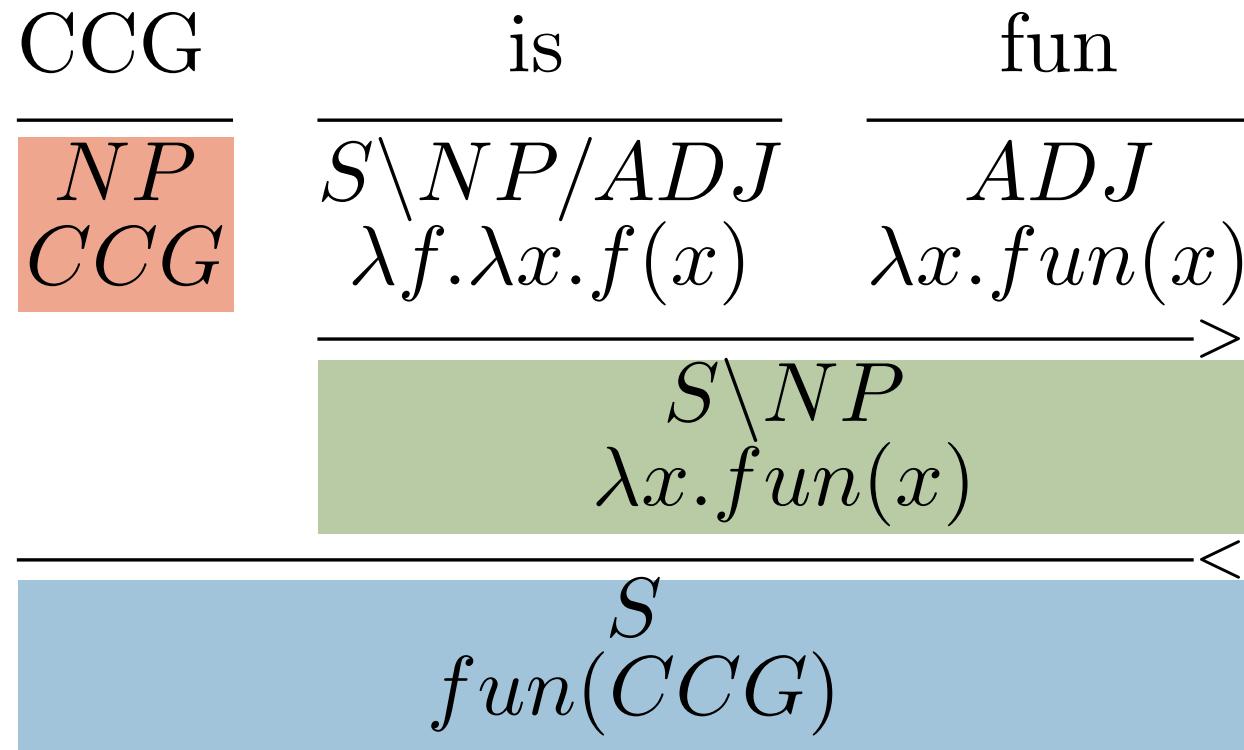
Parsing with CCGs

$$\frac{\text{CCG}}{\frac{NP}{CCG} \quad \frac{\text{is}}{\frac{S \setminus NP / ADJ}{\lambda f. \lambda x. f(x)}} \quad \frac{\text{fun}}{\frac{ADJ}{\lambda x. fun(x)}}} \Rightarrow \boxed{\frac{S \setminus NP}{\lambda x. fun(x)}}$$

Combine categories using operators

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

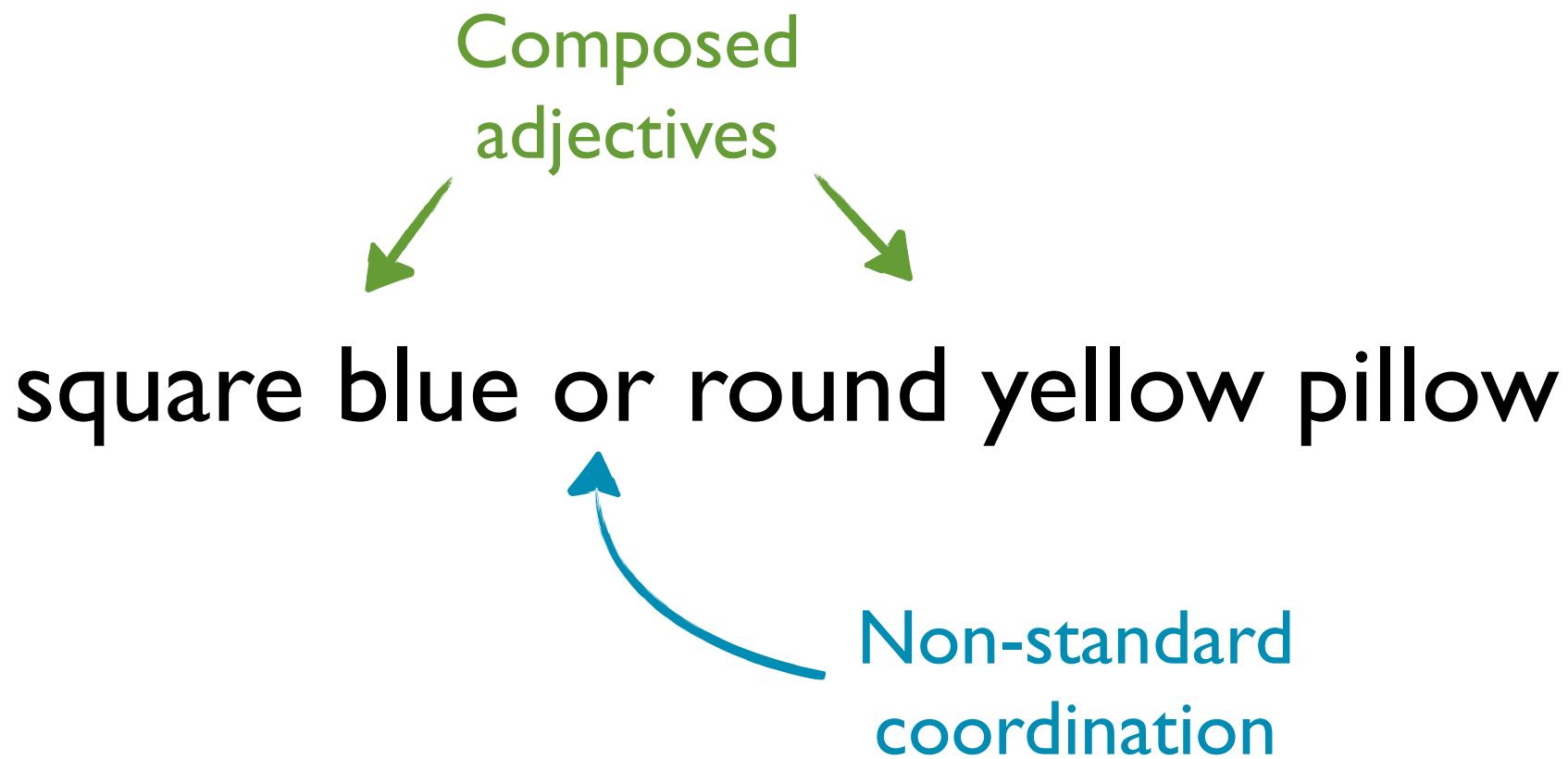
Parsing with CCGs



Combine categories using operators

$$B : g \quad A \setminus B : f \Rightarrow A : f(g) \quad (<)$$

Parsing with CCGs



CCG Operations

Composition

$$A/B : f \quad B/C : g \Rightarrow A/C : \lambda x. f(g(x)) \quad (> B)$$

$$B\backslash C : g \quad A\backslash B : f \Rightarrow A\backslash C : \lambda x. f(g(x)) \quad (< B)$$

- Equivalent to function composition*
- Two directions: forward and backward

* Formal definition of logical composition in supplementary slides

CCG Operations

Composition

$$\begin{array}{c} f \\ \boxed{A/B : f} \end{array} \quad \begin{array}{c} g \\ \boxed{B/C : g} \end{array} \Rightarrow \begin{array}{c} f \circ g \\ \boxed{A/C : \lambda x. f(g(x))} \end{array} \quad (> B)$$
$$B \setminus C : g \quad A \setminus B : f \Rightarrow A \setminus C : \lambda x. f(g(x)) \quad (< B)$$

- Equivalent to function composition*
- Two directions: forward and backward

* Formal definition of logical composition in supplementary slides

CCG Operations

Type Shifting

$$ADJ : \lambda x.g(x) \Rightarrow N/N : \lambda f.\lambda x.f(x) \wedge g(x)$$

$$PP : \lambda x.g(x) \Rightarrow N\backslash N : \lambda f.\lambda x.f(x) \wedge g(x)$$

$$AP : \lambda e.g(e) \Rightarrow S\backslash S : \lambda f.\lambda e.f(e) \wedge g(e)$$

$$AP : \lambda e.g(e) \Rightarrow S/S : \lambda f.\lambda e.f(e) \wedge g(e)$$

- Category-specific unary operations
- Modify category type to take an argument
- Helps in keeping a compact lexicon

CCG Operations

Type Shifting

Input	Output
$ADJ : \lambda x.g(x) \Rightarrow$	$N/N : \lambda f.\lambda x.f(x) \wedge g(x)$

$$PP : \lambda x.g(x) \Rightarrow N\backslash N : \lambda f.\lambda x.f(x) \wedge g(x)$$

$$AP : \lambda e.g(e) \Rightarrow S\backslash S : \lambda f.\lambda e.f(e) \wedge g(e)$$

$$AP : \lambda e.g(e) \Rightarrow S/S : \lambda f.\lambda e.f(e) \wedge g(e)$$

- Category-specific unary operations
- Modify category type to take an argument
- Helps in keeping a compact lexicon

CCG Operations

Type Shifting

Input	Output
$ADJ : \lambda x.g(x) \Rightarrow$	$N/N : \lambda f.\lambda x.f(x) \wedge g(x)$

$$PP : \lambda x.g(x) \Rightarrow N\backslash N : \lambda f.\lambda x.f(x) \wedge g(x)$$

$$AP : \lambda e.g(e) \Rightarrow S\backslash S : \lambda f.\lambda e.f(e) \wedge g(e)$$

Topicalization

$$AP : \lambda e.g(e) \Rightarrow S/S : \lambda f.\lambda e.f(e) \wedge g(e)$$

- Category-specific unary operations
- Modify category type to take an argument
- Helps in keeping a compact lexicon

CCG Operations

Coordination

and $\vdash C : conj$
or $\vdash C : disj$

- Coordination is special cased
 - Specific rules perform coordination
 - Coordinating operators are marked with special lexical entries

Parsing with CCGs

square

blue

or

round

yellow

pillow

Parsing with CCGs

square	blue	or	round	yellow	pillow
$\frac{ADJ}{\lambda x.square(x)}$	$\frac{ADJ}{\lambda x.blue(x)}$	$\frac{C}{disj}$	$\frac{ADJ}{\lambda x.round(x)}$	$\frac{ADJ}{\lambda x.yellow(x)}$	$\frac{N}{\lambda x.pillow(x)}$

Use lexicon to match words and
phrases with their categories

Parsing with CCGs

square	blue	or	round	yellow	pillow
$\begin{array}{c} ADJ \\ \lambda x.square(x) \end{array}$	$\begin{array}{c} ADJ \\ \lambda x.blue(x) \end{array}$	$\frac{C}{disj}$	$\begin{array}{c} ADJ \\ \lambda x.round(x) \end{array}$	$\begin{array}{c} ADJ \\ \lambda x.yellow(x) \end{array}$	
$\lambda f.\lambda x.f(x) \wedge square(x)$					$\begin{array}{c} N \\ \lambda x.pillow(x) \end{array}$

Shift adjectives to combine

$$ADJ : \lambda x.g(x) \Rightarrow N/N : \lambda f.\lambda x.f(x) \wedge g(x)$$

Parsing with CCGs

square	blue	or	round	yellow	pillow
$\frac{ADJ}{\lambda x.square(x)}$	$\frac{ADJ}{\lambda x.blue(x)}$	$\frac{C}{disj}$	$\frac{ADJ}{\lambda x.round(x)}$	$\frac{ADJ}{\lambda x.yellow(x)}$	$\frac{N}{\lambda x.pillow(x)}$
$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge square(x)}$	$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge blue(x)}$		$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge round(x)}$	$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge yellow(x)}$	

Shift adjectives to combine

$$ADJ : \lambda x.g(x) \Rightarrow N/N : \lambda f.\lambda x.f(x) \wedge g(x)$$

Parsing with CCGs

square	blue	or	round	yellow	pillow
$\begin{array}{c} ADJ \\ \lambda x.square(x) \end{array}$	$\begin{array}{c} ADJ \\ \lambda x.blue(x) \end{array}$	$\frac{C}{disj}$	$\begin{array}{c} ADJ \\ \lambda x.round(x) \end{array}$	$\begin{array}{c} ADJ \\ \lambda x.yellow(x) \end{array}$	$\frac{N}{\lambda x.pillow(x)}$
$\lambda f.\lambda x.f(x) \wedge square(x)$	$\lambda f.\lambda x.f(x) \wedge blue(x)$		$\lambda f.\lambda x.f(x) \wedge round(x)$	$\lambda f.\lambda x.f(x) \wedge yellow(x)$	
$\lambda f.\lambda x.f(x) \wedge square(x) \wedge blue(x)$			$\lambda f.\lambda x.f(x) \wedge round(x) \wedge yellow(x)$		

Compose pairs of adjectives

$$A/B : f \quad B/C : g \Rightarrow A/C : \lambda x. f(g(x)) \quad (> B)$$

Parsing with CCGs

square	blue	or	round	yellow	pillow
ADJ $\lambda x.square(x)$	ADJ $\lambda x.blue(x)$	C $disj$	ADJ $\lambda x.round(x)$	ADJ $\lambda x.yellow(x)$	N $\lambda x.pillow(x)$
N/N $\lambda f.\lambda x.f(x) \wedge square(x)$	N/N $\lambda f.\lambda x.f(x) \wedge blue(x)$		N/N $\lambda f.\lambda x.f(x) \wedge round(x)$	N/N $\lambda f.\lambda x.f(x) \wedge yellow(x)$	
N/N $\lambda f.\lambda x.f(x) \wedge square(x) \wedge blue(x)$			N/N $\lambda f.\lambda x.f(x) \wedge round(x) \wedge yellow(x)$		
$\lambda f.\lambda x.f(x) \wedge ((square(x) \wedge blue(x)) \vee (round(x) \wedge yellow(x)))$					

Coordinate composed adjectives

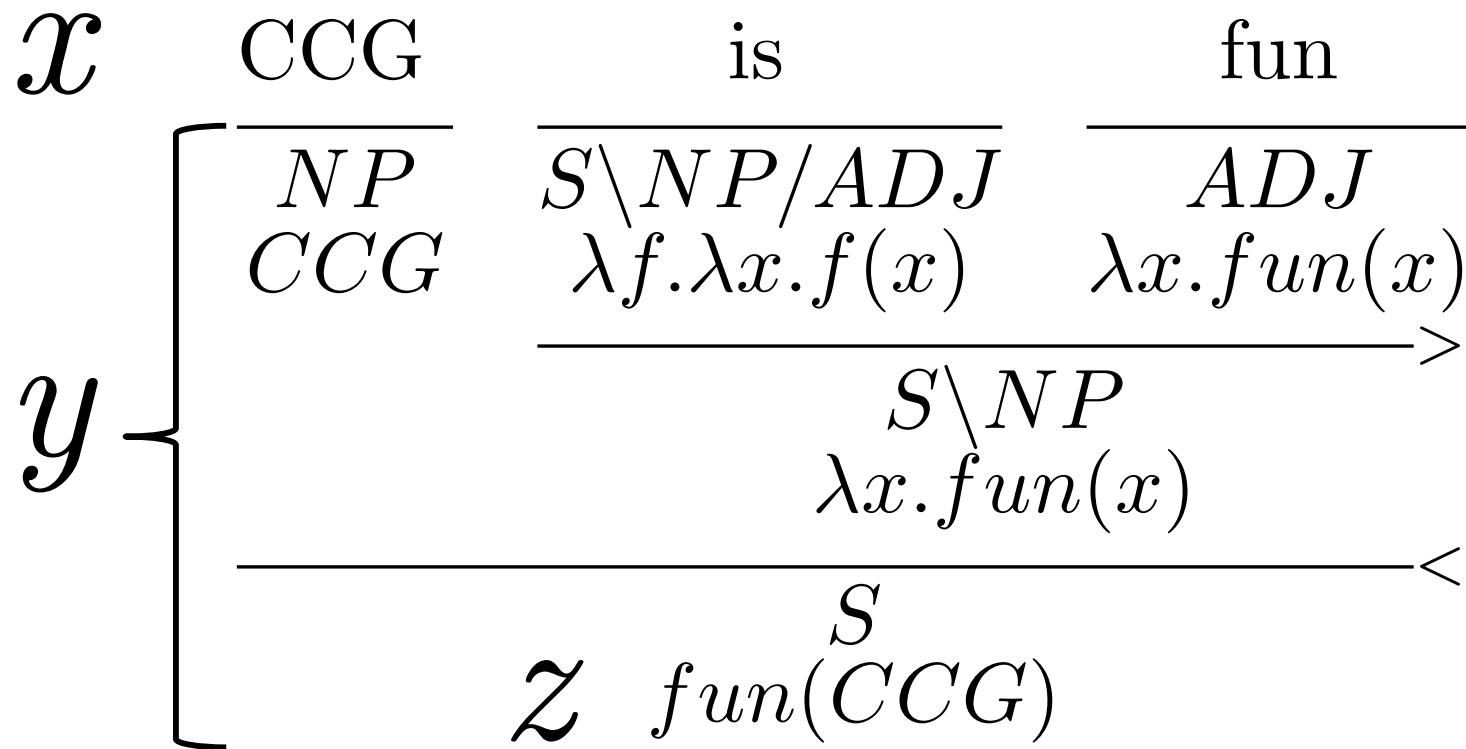
Parsing with CCGs

square	blue	or	round	yellow	pillow
$\frac{ADJ}{\lambda x.square(x)}$	$\frac{ADJ}{\lambda x.blue(x)}$	$\frac{C}{disj}$	$\frac{ADJ}{\lambda x.round(x)}$	$\frac{ADJ}{\lambda x.yellow(x)}$	$\frac{N}{\lambda x.pillow(x)}$
$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge square(x)}$	$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge blue(x)}$		$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge round(x)}$	$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge yellow(x)}$	
$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge square(x) \wedge blue(x)}$			$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge round(x) \wedge yellow(x)}$		
$\frac{N/N}{\lambda f.\lambda x.f(x) \wedge ((square(x) \wedge blue(x)) \vee (round(x) \wedge yellow(x)))} <\Phi>$					
$\frac{N}{\lambda x.pillow(x) \wedge ((square(x) \wedge blue(x)) \vee (round(x) \wedge yellow(x)))} >$					

Apply coordinated adjectives to noun

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

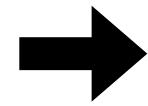
Parsing with CCGs



Lexical Ambiguity

十

Many parsing decisions



Many potential trees and LFs

Weighted Linear CCGs

- Given a weighted linear model:

- CCG lexicon Λ
 - Feature function $f : X \times Y \rightarrow \mathbb{R}^m$
 - Weights $w \in \mathbb{R}^m$

- The best parse is:

$$y^* = \arg \max_y w \cdot f(x, y)$$

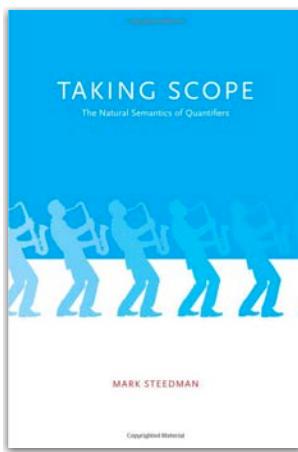
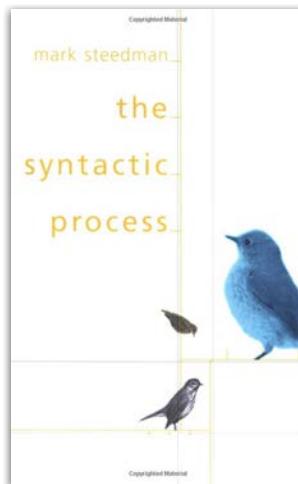
- We consider all possible parses y for sentence x given the lexicon Λ

Parsing Algorithms

- Syntax-only CCG parsing has polynomial time CKY-style algorithms
- Parsing with semantics requires entire category as chart signature
 - e.g., $ADJ : \lambda x. fun(x)$
- In practice, prune to top-N for each span
 - Approximate, but polynomial time



More on CCGs



- Generalized type-raising operations
- Cross composition operations for cross serial dependencies
- Compositional approaches to English intonation
- and a lot more ... even Jazz

[Steedman 1996; 2000; 2011; Granroth and Steedman 2012]

The Lexicon Problem

- Key component of CCG
- Same words often paired with many different categories
- Difficult to learn with limited data

Factored Lexicons

the house dog

the dog of the house

$$\iota x.\text{dog}(x) \wedge \text{of}(x, \iota y.\text{house}(y))$$

the garden dog

$$\iota x.\text{dog}(x) \wedge \text{of}(x, \iota y.\text{garden}(y))$$

- Lexical entries share information
- Decomposition of entries can lead to more compact lexicons

Factored Lexicons

the house dog house $\vdash ADJ : \lambda x.of(x, \iota y.house(y))$

the dog of the house $\text{house} \vdash N : \lambda x.\text{house}(x)$

$$\iota x.dog(x) \wedge of(x, \iota y.house(y))$$

the garden dog garden $\vdash ADJ : \lambda x.of(x, \iota y.garden(y))$

$$\iota x.\text{dog}(x) \wedge \text{of}(x, \iota y.\text{garden}(y))$$

- Lexical entries share information
 - Decomposition of entries can lead to more compact lexicons

Factored Lexicons

the house dog

house $\vdash ADJ : \lambda x.of(x, \iota y.house(y))$

the dog of the house

house $\vdash N : \lambda x.house(x)$

$\iota x.dog(x) \wedge of(x, \iota y.house(y))$

the garden dog

garden $\vdash ADJ : \lambda x.of(x, \iota y.garden(y))$

$\iota x.dog(x) \wedge of(x, \iota y.garden(y))$

- Lexical entries share information
- Decomposition of entries can lead to more compact lexicons

Factored Lexicons

the house dog

house $\vdash ADJ : \lambda x.of(x, \lambda y.house(y))$

the dog of the house

house $\vdash N : \lambda x.house(x)$

$\lambda x.dog(x) \wedge of(x, \lambda y.house(y))$

the garden dog

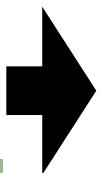
garden $\vdash ADJ : \lambda x.of(x, \lambda y.garden(y))$

$\lambda x.dog(x) \wedge of(x, \lambda y.garden(y))$

- Lexical entries share information
- Decomposition of entries can lead to more compact lexicons

Factored Lexicons

house $\vdash ADJ : \lambda x.of(x, \iota y.house(y))$
house $\vdash N : \lambda x.house(x)$
garden $\vdash ADJ : \lambda x.of(x, \iota y.garden(y))$



Lexemes

(garden, $\{garden\}$)
(house, $\{house\}$)

Templates

$\lambda(\omega, \{v_i\}_1^n).$
 $[\omega \vdash ADJ : \lambda x.of(x, \iota y.v_1(y))]$
 $\lambda(\omega, \{v_i\}_1^n).$
 $[\omega \vdash N : \lambda x.v_1(x)]$

Factored Lexicons

Templates

$$\lambda(\omega, \{v_i\}_1^n).$$
$$[\omega \vdash ADJ : \lambda x.of(x, \iota y.v_1(y))]$$
$$\lambda(\omega, \{v_i\}_1^n).$$
$$[\omega \vdash N : \lambda x.v_1(x)]$$

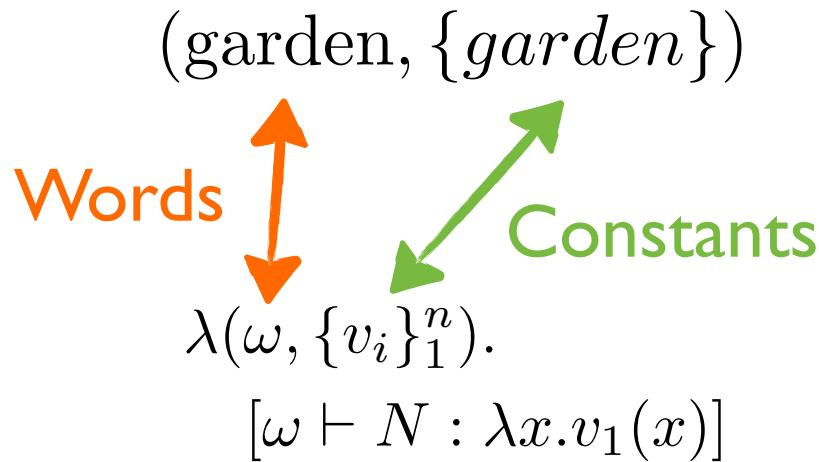
Lexemes

$$(garden, \{garden\})$$
$$(house, \{house\})$$

- Capture systematic variations in word usage
- Each variation can then be applied to compact units of lexical meaning

- Model word meaning
- Abstracts the compositional nature of the word

Factored Lexicons



$\omega \leftarrow \text{garden}$
 $v_1 \leftarrow \text{garden}$

A large blue downward-pointing arrow is positioned to the left of the assignment statements $\omega \leftarrow \text{garden}$ and $v_1 \leftarrow \text{garden}$.

$\text{garden} \vdash N : \lambda x.\text{garden}(x)$

Factored Lexicons

Original Lexicon

$$\begin{aligned}\vdash S|NP \quad \lambda x.\text{flight } x \\ \vdash S|NP/ \; S|NP \quad \lambda f.\lambda x.\text{flight } x \wedge f \; x \\ \vdash S|NP\backslash \; S|NP \quad \lambda f.\lambda x.\text{flight } x \wedge f \; x \\ \vdash S|NP \quad \lambda x.\text{trans } x \\ \vdash S|NP/ \; S|NP \quad \lambda f.\lambda x.\text{trans } x \wedge f \; x \\ \vdash S|NP\backslash \; S|NP \quad \lambda f.\lambda x.\text{trans } x \wedge f \; x\end{aligned}$$

Factored Lexicon

(flight, $\{\text{flight}\}$)
(ground transport, $\{\text{trans}\}$)

$$\begin{aligned}\lambda(\omega, \{v_i\}_1^n).[\omega \vdash S|NP : \lambda x.v_1(x)] \\ \lambda(\omega, \{v_i\}_1^n).[\omega \vdash S|NP/(S|NP) : \lambda f.\lambda x.v_1(x) \wedge f(x)] \\ \lambda(\omega, \{v_i\}_1^n).[\omega \vdash S|NP\backslash(S|NP) : \lambda f.\lambda x.v_1(x) \wedge f(x)]\end{aligned}$$

Factoring a Lexical Entry

house $\vdash ADJ : \lambda x.of(x, \iota y.house(y))$

Partial
factoring

(house, {*house*})
 $\lambda(\omega, \{v_i\}_1^n).[\omega \vdash ADJ : \lambda x.of(x, \iota y.v_1(y))]$

Partial
factoring

(house, {*of*})
 $\lambda(\omega, \{v_i\}_1^n).[\omega \vdash ADJ : \lambda x.v_1(x, \iota y.house(y))]$

Maximal
factoring

(house, {*of*, *house*})
 $\lambda(\omega, \{v_i\}_1^n).[\omega \vdash ADJ : \lambda x.v_1(x, \iota y.v_2(y))]$

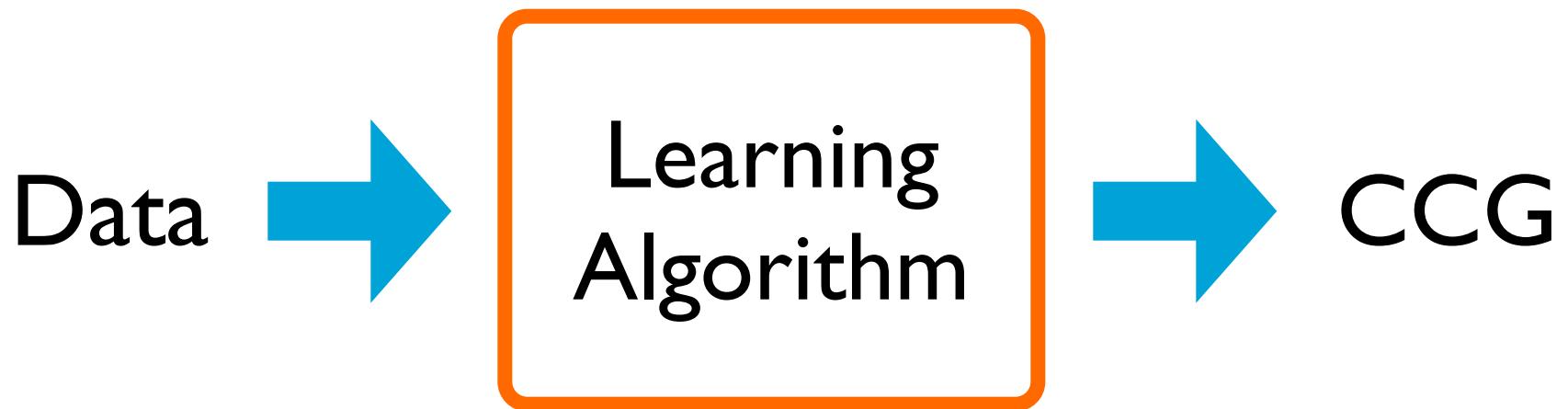
Parsing

Learning

Modeling

- Lambda calculus
- Parsing with Combinatory Categorial Grammars
- Linear CCGs
- Factored lexicons

Learning



- What kind of data/supervision we can use?
- What do we need to learn?

Parsing as Structure Prediction

$$\begin{array}{cccc} \text{show} & \text{me} & \text{flights} & \text{to} \\ \hline S/N & & N & PP/NP \\ \lambda f.f & & \lambda x.\text{flight}(x) & \lambda y.\lambda x.\text{to}(x, y) \\ & & & \hline & & & NP \\ & & & BOSTON \\ & & & \overrightarrow{PP} \\ & & & \lambda x.\text{to}(x, BOSTON) \\ & & & \hline & & N \setminus N & \\ & & \lambda f.\lambda x.f(x) \wedge \text{to}(x, BOSTON) & \leftarrow N \\ & & \hline & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & \leftarrow N \\ & & \hline & & \overrightarrow{S} & \\ & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & \end{array}$$

Learning CCG

$$\begin{array}{cccc}
 \text{show} & \text{me} & \text{flights} & \text{to} & \text{Boston} \\
 \hline
 S/N & & N & PP/NP & NP \\
 \lambda f.f & & \lambda x.\text{flight}(x) & \lambda y.\lambda x.\text{to}(x, y) & BOSTON \\
 & & & & \xrightarrow{\hspace{1cm}} \\
 & & & & \lambda x.\text{to}(x, BOSTON) \\
 & & & & PP \\
 & & & & \lambda f.\lambda x.f(x) \wedge \text{to}(x, BOSTON) \\
 & & & & N \setminus N \\
 & & & & \xleftarrow{\hspace{1cm}} \\
 & & & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) \\
 & & & & N \\
 & & & & \xrightarrow{\hspace{1cm}} \\
 & & & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) \\
 & & & & S
 \end{array}$$

Lexicon

Combinators

Predefined

w

Supervised Data

$$\begin{array}{cccc} \text{show} & \text{me} & \text{flights} & \text{to} \\ \hline S/N & & N & PP/NP \\ \lambda f.f & & \lambda x.\text{flight}(x) & \lambda y.\lambda x.\text{to}(x, y) \\ & & & \hline & & & NP \\ & & & BOSTON \\ & & & \overrightarrow{PP} \\ & & & \lambda x.\text{to}(x, BOSTON) \\ & & & \hline & & N \setminus N & \\ & & \lambda f.\lambda x.f(x) \wedge \text{to}(x, BOSTON) & \leftarrow N \\ & & \hline & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & \leftarrow N \\ & & \hline & & \overrightarrow{S} & \\ & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & \end{array}$$

Supervised Data

$$\begin{array}{cccc} \text{show} & \text{me} & \text{flights} & \text{to} & \text{Boston} \\ \hline S/N & & N & PP/NP & NP \\ \lambda f.f & & \lambda x.\text{flight}(x) & \lambda y.\lambda x.\text{to}(x, y) & BOSTON \\ & & & \hline & & & & \longrightarrow \\ & & & & \lambda x.\text{o}(x, BOSTON) \\ & & & & P \\ & & & & \hline & & & N \setminus N & \\ & & & \lambda f.\lambda x.f(x) \wedge \text{to}(x, BOSTON) & \\ & & & \hline & & N & & \\ & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & & \\ \hline & & & & \longrightarrow \\ & & & S & \\ & & & \lambda x.\text{flight}(x) \wedge \text{to}(x, BOSTON) & \end{array}$$

Latent

Supervised Data

Supervised learning is done from pairs
of sentences and logical forms

Show me flights to Boston

$\lambda x. flight(x) \wedge to(x, BOSTON)$

I need a flight from baltimore to seattle

$\lambda x. flight(x) \wedge from(x, BALTIMORE) \wedge to(x, SEATTLE)$

what ground transportation is available in san francisco

$\lambda x. ground_transport(x) \wedge to_city(x, SF)$

Weak Supervision

- Logical form is latent
- “Labeling” requires less expertise
- Labels don’t uniquely determine correct logical forms
- Learning requires executing logical forms within a system and evaluating the result

Weak Supervision

Learning from Query Answers

What is the largest state that borders Texas?

New Mexico

Weak Supervision

Learning from Query Answers

What is the largest state that borders Texas?

New Mexico

$$\begin{aligned} & \text{argmax}(\lambda x. \text{state}(x) \\ & \quad \wedge \text{border}(x, TX), \lambda y. \text{size}(y)) \end{aligned}$$
$$\begin{aligned} & \text{argmax}(\lambda x. \text{river}(x) \\ & \quad \wedge \text{in}(x, TX), \lambda y. \text{size}(y)) \end{aligned}$$

Weak Supervision

Learning from Query Answers

What is the largest state that borders Texas?

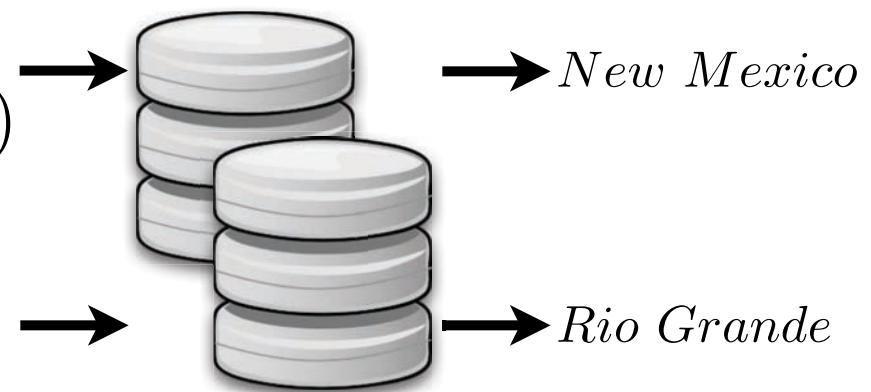
New Mexico

$$\operatorname{argmax}(\lambda x. \operatorname{state}(x)$$

$$\wedge \operatorname{border}(x, TX), \lambda y. \operatorname{size}(y))$$

$$\operatorname{argmax}(\lambda x. \operatorname{river}(x)$$

$$\wedge \operatorname{in}(x, TX), \lambda y. \operatorname{size}(y))$$



Weak Supervision

Learning from Query Answers

What is the largest state that borders Texas?

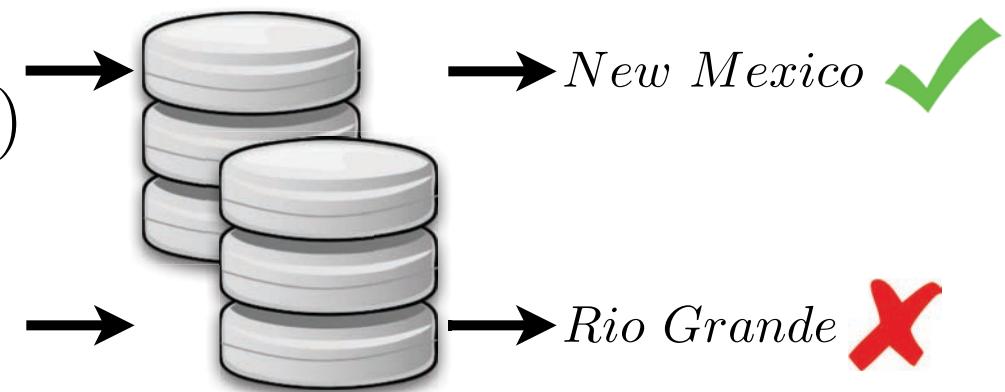
New Mexico

$\text{argmax}(\lambda x.\text{state}(x)$

$\wedge \text{border}(x, TX), \lambda y.\text{size}(y))$

$\text{argmax}(\lambda x.\text{river}(x)$

$\wedge \text{in}(x, TX), \lambda y.\text{size}(y))$



Weak Supervision

Learning from Demonstrations

at the chair, move forward three steps past the sofa



Weak Supervision

Learning from Demonstrations

at the chair, move forward three steps past the sofa



Some examples from other domains:

- Sentences and labeled game states [Goldwasser and Roth 2011]
- Sentences and sets of physical objects [Matuszek et al. 2012]

Weak Supervision

Learning from Conversation Logs

- SYSTEM how can I help you ? (OPEN_TASK)
- USER i ' d like to fly to new york
- SYSTEM flying to new york . (CONFIRM: $from(fl, ATL)$) leaving what city ?
(ASK: $\lambda x.from(fl,x)$)
- USER from boston on june seven with american airlines
- SYSTEM flying to new york . (CONFIRM: $to(fl, NYC)$) what date would you
like to depart boston ? (ASK: $\lambda x.date(fl,x) \wedge to(fl, BOS)$)
- USER june seventh
- [CONVERSATION CONTINUES]

Parsing

Learning

Modeling

- Structured perceptron
- A unified learning algorithm
- Supervised learning
- Weak supervision

Structured Perceptron

- Simple additive updates
 - Only requires efficient decoding (argmax)
 - Closely related to maxent and other feature rich models
 - Provably finds linear separator in finite updates, if one exists
- Challenge: learning with hidden variables

Structured Perceptron

Data: $\{(x_i, y_i) : i = 1 \dots n\}$

For $t = 1 \dots T$:

[iterate epochs]

For $i = 1 \dots n$:

[iterate examples]

$$y^* \leftarrow \arg \max_y \langle \theta, \Phi(x_i, y) \rangle$$

[predict]

If $y^* \neq y_i$:

[check]

$$\theta \leftarrow \theta + \Phi(x_i, y_i) - \Phi(x_i, y^*)$$

[update]

One Derivation of the Perceptron

Log-linear model: $p(y|x) = \frac{e^{w \cdot f(x,y)}}{\sum_{y'} e^{w \cdot f(x,y')}}$

Step 1: Differentiate, to maximize data log-likelihood

$$update = \sum_i f(x_i, y_i) - E_{p(y|x_i)} f(x_i, y)$$

Step 2: Use online, stochastic gradient updates, for example i :

$$update_i = f(x_i, y_i) - E_{p(y|x_i)} f(x_i, y)$$

Step 3: Replace expectations with maxes (Viterbi approx.)

$$update_i = f(x_i, y_i) - f(x_i, y^*) \text{ where } y^* = \arg \max_y w \cdot f(x_i, y)$$

The Perceptron with Hidden Variables

Log-linear model: $p(y|x) = \sum_h p(y, h|x)$ $p(y, h|x) = \frac{e^{w \cdot f(x, h, y)}}{\sum_{y', h'} e^{w \cdot f(x, h', y')}}$

Step 1: Differentiate marginal, to maximize data log-likelihood

$$update = \sum_i E_{p(h|y_i, x_i)}[f(x_i, h, y_i)] - E_{p(y, h|x_i)}[f(x_i, h, y)]$$

Step 2: Use online, stochastic gradient updates, for example i :

$$update_i = E_{p(y_i, h|x_i)}[f(x_i, h, y_i)] - E_{p(y, h|x_i)}[f(x_i, h, y)]$$

Step 3: Replace expectations with maxes (Viterbi approx.)

$$update_i = f(x_i, h', y_i) - f(x_i, h^*, y^*) \text{ where}$$

$$y^*, h^* = \arg \max_{y, h} w \cdot f(x_i, h, y) \quad \text{and} \quad h' = \arg \max_h w \cdot f(x_i, h, y_i)$$

Hidden Variable Perceptron

Data: $\{(x_i, y_i) : i = 1 \dots n\}$

For $t = 1 \dots T$: [iterate epochs]

For $i = 1 \dots n$: [iterate examples]

$y^*, h^* \leftarrow \arg \max_{y,h} \langle \theta, \Phi(x_i, h, y) \rangle$ [predict]

If $y^* \neq y_i$: [check]

$h' \leftarrow \arg \max_h \langle \theta, \Phi(x_i, h, y_i) \rangle$ [predict hidden]

$\theta \leftarrow \theta + \Phi(x_i, h', y_i) - \Phi(x_i, h^*, y^*)$ [update]

Hidden Variable Perceptron

- No known convergence guarantees
 - Log-linear version is non-convex
- Simple and easy to implement
 - Works well with careful initialization
- Modifications for semantic parsing
 - Lots of different hidden information
 - Can add a margin constraint, do probabilistic version, etc.

Unified Learning Algorithm

- Handle various learning signals
- Estimate parsing parameters
- Induce lexicon structure
- Related to loss-sensitive structured perceptron [Singh-Miller and Collins 2007]

Learning Choices

Validation Function

$$\mathcal{V} : \mathcal{Y} \rightarrow \{t, f\}$$

- Indicates correctness of a parse y
- Varying \mathcal{V} allows for differing forms of supervision

Lexical Generation Procedure

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

- Given:
 - sentence x
 - validation function \mathcal{V}
 - lexicon Λ
 - parameters θ
- Produce a overly general set of lexical entries

Unified Learning Algorithm

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n :$

Step 1: (Lexical generation)

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

- **Online**
- **2 steps:**
 - Lexical generation
 - Parameter update

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n :$

Step 1: (Lexical generation)

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Initialize parameters and lexicon

θ weights

Λ_0 initial lexicon

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n :$

Step 1: (Lexical generation)

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Iterate over data

T # iterations

n # samples

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Generate a large set of potential lexical entries

θ weights

x sentence

\mathcal{V} validation function

$GENLEX(x, \mathcal{V}; \lambda, \theta)$

lexical generation function

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Generate a large set of potential lexical entries

θ weights

x sentence

\mathcal{V} validation function

$GENLEX(x, \mathcal{V}; \lambda, \theta)$

lexical generation function

$$\mathcal{V} : \mathcal{Y} \rightarrow \{t, f\}$$

\mathcal{Y} all parses

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Generate a large set of potential lexical entries

θ weights

x sentence

\mathcal{V} validation function

$GENLEX(x, \mathcal{V}; \lambda, \theta)$

lexical generation function

Procedure to propose potential new lexical entries for a sentence

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Get top parses

x sentence

k beam size

$GEN(x; \lambda)$ set of all parses

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Get lexical entries from
highest scoring valid
parses

θ weights

\mathcal{V} validation function

$LEX(y)$ set of lexical entries

$\phi_i(y) = \phi(x_i, y)$

$MAXV_i(Y; \theta) =$

$$\{y | \forall y' \in Y, \langle \theta, \Phi_i(y') \rangle \leq \langle \theta, \Phi_i(y) \rangle \wedge \mathcal{V}_i(y)\}$$

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

- a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
 $\lambda \leftarrow \Lambda \cup \lambda_G$
- b. Let Y be the k highest scoring parses from
 $GEN(x_i; \lambda)$
- c. Select lexical entries from the highest scor-
ing valid parses:
$$\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$$
- d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

Update model's lexicon

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

Step 2: (Update parameters)

- a. Set $G_i \leftarrow MAXV_i(GEN(x_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i; \Lambda) \wedge \neg \mathcal{V}_i(y)\}$
- b. Construct sets of margin violating good and bad parses:

$$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

$$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

- c. Apply the additive update:

$$\begin{aligned} \theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) \\ - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e) \end{aligned}$$

Output: Parameters θ and lexicon Λ

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

Step 2: (Update parameters)

- a. Set $G_i \leftarrow MAXV_i(GEN(x_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i; \Lambda) \wedge \neg \mathcal{V}_i(y)\}$
- b. Construct sets of margin violating good and bad parses:

$$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

$$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

- c. Apply the additive update:

$$\begin{aligned} \theta &\leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) \\ &\quad - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e) \end{aligned}$$

Output: Parameters θ and lexicon Λ

Re-parse and group all parses into ‘good’ and ‘bad’ sets

θ weights

x sentence

\mathcal{V} validation function

$GEN(x; \lambda)$ set of all parses

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

Step 2: (Update parameters)

- Set $G_i \leftarrow MAXV_i(GEN(x_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i; \Lambda) \wedge \neg \mathcal{V}_i(y)\}$
- Construct sets of margin violating good and bad parses:

$$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

$$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

- Apply the additive update:

$$\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$$

Output: Parameters θ and lexicon Λ

For all pairs of ‘good’ and ‘bad’ parses, if their scores violate the margin, add each to ‘right’ and ‘error’ sets respectively

θ weights

γ margin

$\phi_i(y) = \phi(x_i, y)$

$\Delta_i(y, y') = |\Phi_i(y) - \Phi_i(y')|_1$

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Lexical generation)

Step 2: (Update parameters)

- a. Set $G_i \leftarrow MAXV_i(GEN(x_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i; \Lambda) \wedge \neg \mathcal{V}_i(y)\}$
- b. Construct sets of margin violating good and bad parses:

$$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

$$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \text{ s.t. } \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

- c. Apply the additive update:

$$\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$$

Output: Parameters θ and lexicon Λ

Update towards
violating ‘good’ parses
and against violating ‘bad’
parses

θ weights

$$\phi_i(y) = \phi(x_i, y)$$

Features and Initialization

Feature
Classes

- Parse: indicate lexical entry and combinator use
- Logical form: indicate local properties of logical forms, such as constant co-occurrence

Lexicon
Initialization

- Always use an NP list
- Sometimes include additional, domain independent entries for function words

Initial
Weights

- Positive weight for initial lexical indicator features

Unified Learning Algorithm

\mathcal{V} validation function

$GENLEX(x, \mathcal{V}; \lambda, \theta)$

lexical generation function

- Two parts of the algorithm we still need to define
- Depend on the task and supervision signal

Unified Learning Algorithm

Supervised

\mathcal{V}

Template-based *GENLEX*

Unification-based *GENLEX*

Weakly Supervised

\mathcal{V}

Template-based *GENLEX*

Supervised Learning

show me the afternoon flights from LA to boston

$$\lambda x. flight(x) \wedge during(x, AFTERNOON) \wedge from(x, LA) \wedge to(x, BOS)$$

Supervised Learning

show me the afternoon flights from LA to boston

$$\lambda x. flight(x) \wedge during(x, AFTERNOON) \wedge from(x, LA) \wedge to(x, BOS)$$

Parse structure is latent

Supervised Validation Function

- Validate logical form against gold label

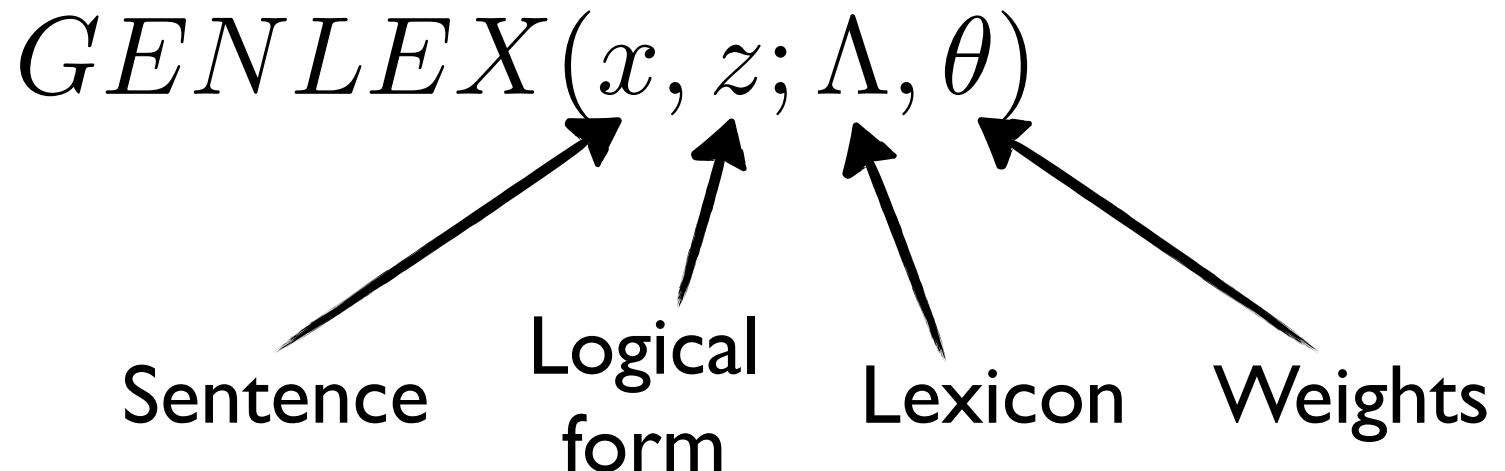
$$\mathcal{V}_i(y) = \begin{cases} \text{true} & \text{if } LF(y) = z_i \\ \text{false} & \text{else} \end{cases}$$

y parse

z_i labeled logical form

$LF(y)$ logical form at the root of y

Supervised Template-based



Small notation abuse:
take labeled logical
form instead of
validation function

Supervised Template-based

$$GENLEX(x, z; \Lambda, \theta)$$

I want a flight to new york

$\lambda x. flight(x) \wedge to(x, NYC)$

Supervised Template-based GENLEX

- Use templates to constrain lexical entries structure
- For example: from a small annotated dataset

$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash ADJ : \lambda x. v_1(x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash PP : \lambda x. \lambda y. v_1(y, x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash N : \lambda x. v_1(x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash S \setminus NP/NP : \lambda x. \lambda y. v_1(x, y)]$$

...

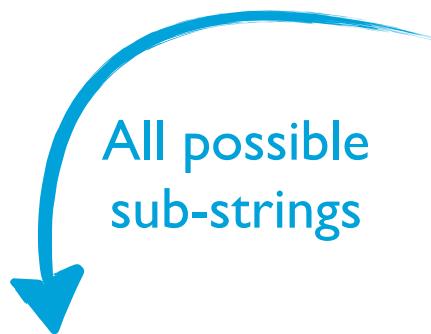
Supervised Template-based GENLEX

Need lexemes to instantiate templates

$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash ADJ : \lambda x. v_1(x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash PP : \lambda x. \lambda y. v_1(y, x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash N : \lambda x. v_1(x)]$$
$$\lambda(\omega, \{v_i\}_1^n).[\omega \vdash S \setminus NP/NP : \lambda x. \lambda y. v_1(x, y)]$$

...

Supervised Template-based

$$GENLEX(x, z; \Lambda, \theta)$$


I want
a flight
flight
flight to new
...

I want a flight to new york

$$\lambda x. flight(x) \wedge to(x, NYC)$$

Supervised Template-based

$$GENLEX(x, z; \Lambda, \theta)$$

I want
a flight
flight

flight to new

...

I want a flight to new york

$$\lambda x. flight(x) \wedge to(x, NYC)$$

All logical
constants from
labeled logical form

flight
to
NYC



Supervised Template-based

$$GENLEX(x, z; \Lambda, \theta)$$

I want a flight to new york

$$\lambda x. flight(x) \wedge to(x, NYC)$$

I want
a flight
flight
flight to new

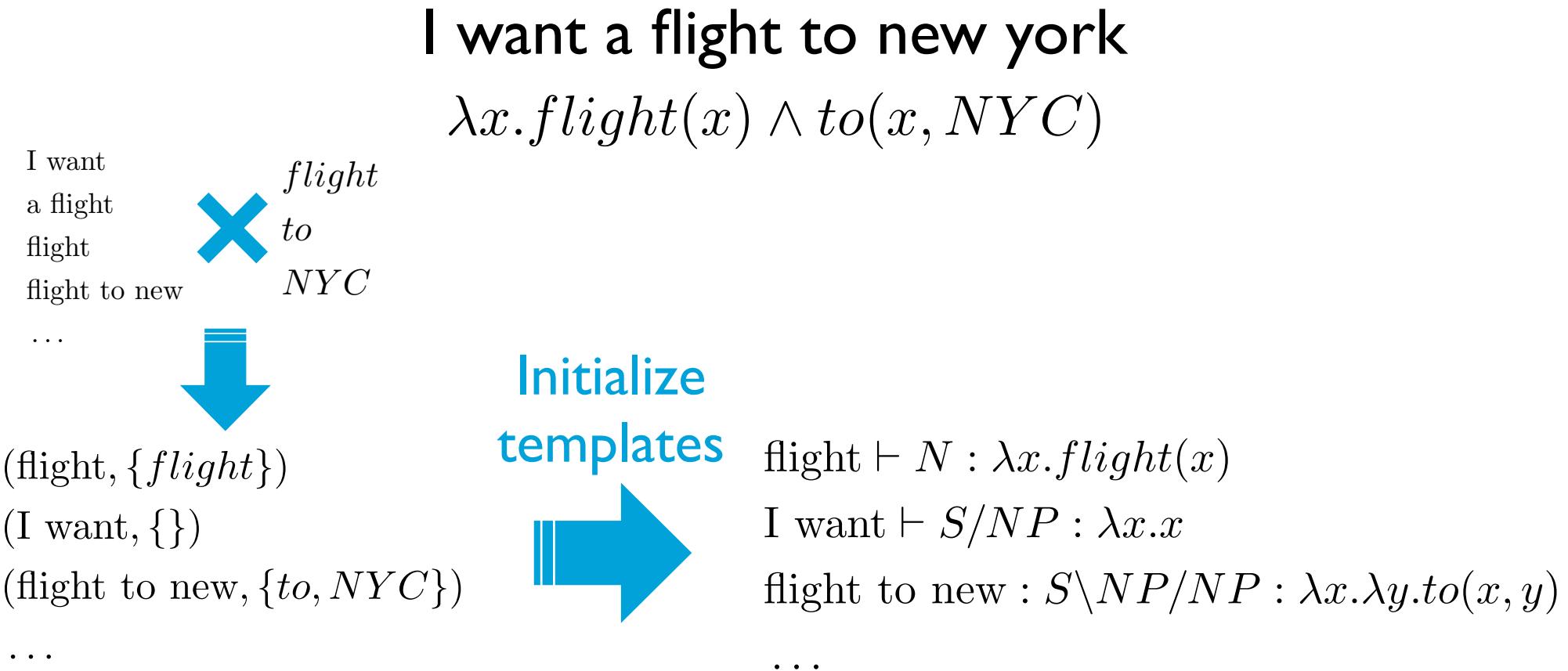


Create
lexemes

(flight, {flight})
(I want, {})
(flight to new, {to, NYC})

...

Supervised Template-based

$$GENLEX(x, z; \Lambda, \theta)$$


Fast Parsing with Pruning

- GENLEX outputs a large number of entries
- For fast parsing: use the labeled logical form to prune
- Prune partial logical forms can't lead to labeled form

I want a flight from New York to Boston on Delta

$\lambda x. from(x, NYC) \wedge to(x, BOS) \wedge carrier(x, DL)$

Fast Parsing with Pruning

I want a flight from New York to Boston on Delta

$$\lambda x. from(x, NYC) \wedge to(x, BOS) \wedge carrier(x, DL)$$

...	form	New York	to	Boston	...
	PP/NP	NP	PP/NP	NP	
	$\lambda x. \lambda y. to(y, x)$	NYC	$\lambda x. \lambda y. to(y, x)$	BOS	

Fast Parsing with Pruning

I want a flight from New York to Boston on Delta

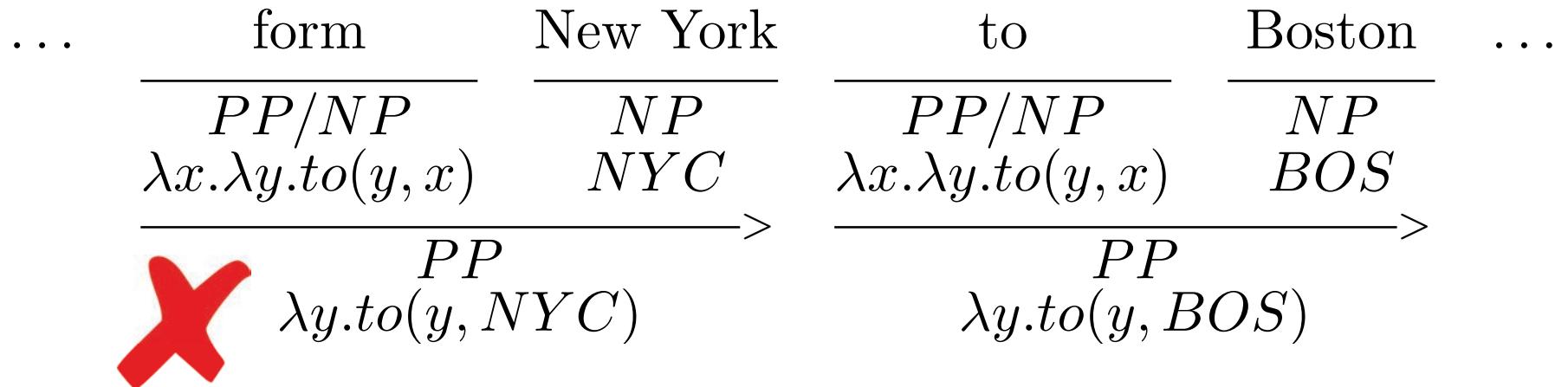
$\lambda x. from(x, NYC) \wedge to(x, BOS) \wedge carrier(x, DL)$

...	form	New York	to	Boston	...
	$\frac{}{PP/NP}$	$\frac{}{NP}$	$\frac{}{PP/NP}$	$\frac{}{NP}$	
	$\lambda x. \lambda y. to(y, x)$	NYC	$\lambda x. \lambda y. to(y, x)$	BOS	
	$\frac{\lambda x. \lambda y. to(y, x)}{PP}$		$\frac{\lambda x. \lambda y. to(y, x)}{PP}$		
		$\lambda y. to(y, NYC)$		$\lambda y. to(y, BOS)$	

Fast Parsing with Pruning

I want a flight from New York to Boston on Delta

$\lambda x.from(x, NYC) \wedge to(x, BOS) \wedge carrier(x, DL)$



Supervised Template-based GENLEX

Summary

No initial expert knowledge	
Creates compact lexicons	✓
Language independent	
Representation independent	
Easily inject linguistic knowledge	✓
Weakly supervised learning	✓

Unification-based GENLEX

- Automatically learns the templates
 - Can be applied to any language and many different approaches for semantic modeling
- Two step process
 - Initialize lexicon with labeled logical forms
 - “Reverse” parsing operations to split lexical entries

Unification-based GENLEX

- Initialize lexicon with labeled logical forms

For every labeled training example:

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

Initialize the lexicon with:

I want a flight to Boston $\vdash S : \lambda x. flight(x) \wedge to(x, BOS)$

Unification-based GENLEX

- Splitting lexical entries

I want a flight to Boston $\vdash S : \lambda x. flight(x) \wedge to(x, BOS)$



I want a flight $\vdash S/(S|NP) : \lambda f. \lambda x. flight(x) \wedge f(x)$

to Boston $\vdash S|NP : \lambda x. to(x, BOS)$

Unification-based GENLEX

- Splitting lexical entries

I want a flight to Boston $\vdash S : \lambda x. flight(x) \wedge to(x, BOS)$



I want a flight $\vdash S/(S|NP) : \lambda f. \lambda x. flight(x) \wedge f(x)$

to Boston $\vdash S|NP : \lambda x. to(x, BOS)$

Many possible phrase pairs 

Many possible category pairs

Unification-based GENLEX

- Splitting CCG categories:

I. Split logical form h to f and g s.t.

$$f(g) = h \text{ or } \lambda x. f(g(x)) = h$$

$$\begin{array}{c} S : \lambda x. flight(x) \wedge to(x, BOS) \quad \Rightarrow \\ \lambda f. \lambda x. flight(x) \wedge f(x) \\ \lambda x. to(x, BOS) \\ \dots \end{array}$$

Unification-based GENLEX

- Splitting CCG categories:
 - I. Split logical form h to f and g s.t.

$$f(g) = h \text{ or } \lambda x. f(g(x)) = h$$

2. Infer syntax from logical form type

$$S/(S|NP) : \lambda f. \lambda x. flight(x) \wedge f(x)$$

$$S|NP : \lambda x. to(x, BOS)$$

$$S : \lambda x. flight(x) \wedge to(x, BOS) \quad \Rightarrow$$

$$S/NP : \lambda y. \lambda x. flight(x) \wedge f(x, y)$$

$$NP : BOS$$

...

Unification-based GENLEX

- Split text and create all pairs

I want a flight to Boston $\vdash S : \lambda x. flight(x) \wedge to(x, BOS)$



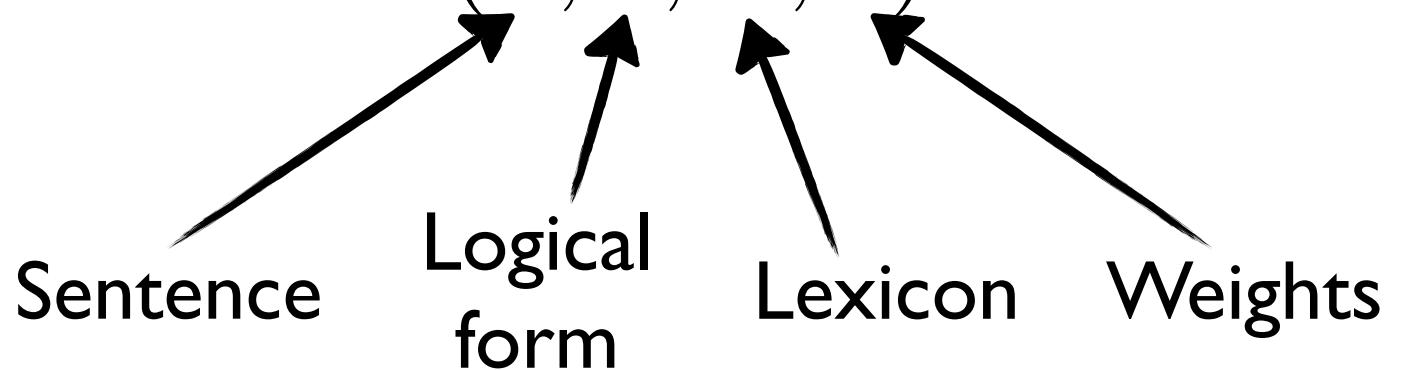
I want $S/(S|NP) : \lambda f. \lambda x. flight(x) \wedge f(x)$
a flight to Boston $S|NP : \lambda x. to(x, BOS)$

I want a flight $S/(S|NP) : \lambda f. \lambda x. flight(x) \wedge f(x)$
to Boston $S|NP : \lambda x. to(x, BOS)$

...

Unification-based

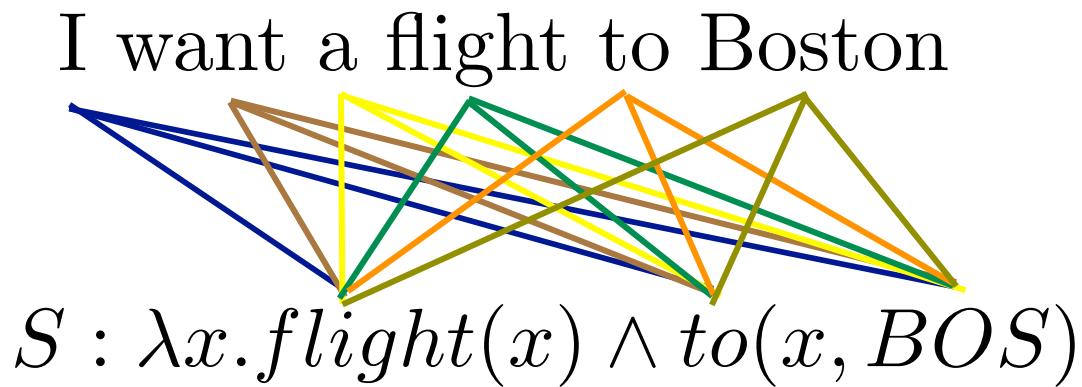
$$GENLEX(x, z; \Lambda, \theta)$$



1. Find highest scoring correct parse
2. Find split that most increases score
3. Return new lexical entries

Parameter Initialization

Compute co-occurrence (IBM Model I)
between words and logical constants



Initial score for new lexical entries: average
over pairwise weights

Unification-based

$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

Unification-based

$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

I want a flight to Boston

S

$\lambda x. flight(x) \wedge to(x, BOS)$

Unification-based

$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x.\text{flight}(x) \wedge \text{to}(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

I want a flight

 $S/(S|NP)$
 $\lambda f.\lambda x.\text{flight}(x) \wedge f(x)$

to Boston

 $S|NP$
 $\lambda x.\text{to}(x, BOS)$

I want a flight to Boston

 $\lambda x.\text{flight}(x) \wedge \overset{S}{\text{to}}(x, BOS)$

Unification-based

$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x.\text{flight}(x) \wedge \text{to}(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

I want a flight

$S/(S|NP)$

$$\lambda f.\lambda x.\text{flight}(x) \wedge f(x)$$

to Boston

$S|NP$

$$\lambda x.\text{to}(x, BOS)$$

I want a flight to Boston

S

$$\lambda x.\text{flight}(x) \wedge \text{to}(x, BOS)$$

Unification-based

$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

Iteration 2

$$\frac{\begin{array}{c} \text{I want a flight} \\ \hline S/(S|NP) \end{array}}{\lambda f. \lambda x. flight(x) \wedge f(x)} \quad \frac{\begin{array}{c} \text{to Boston} \\ \hline S|NP \end{array}}{\lambda x. to(x, BOS)} \rightarrow \frac{\begin{array}{c} S \\ \lambda x. flight(x) \wedge to(x, BOS) \end{array}}{} \quad \rightarrow$$

Unification-based

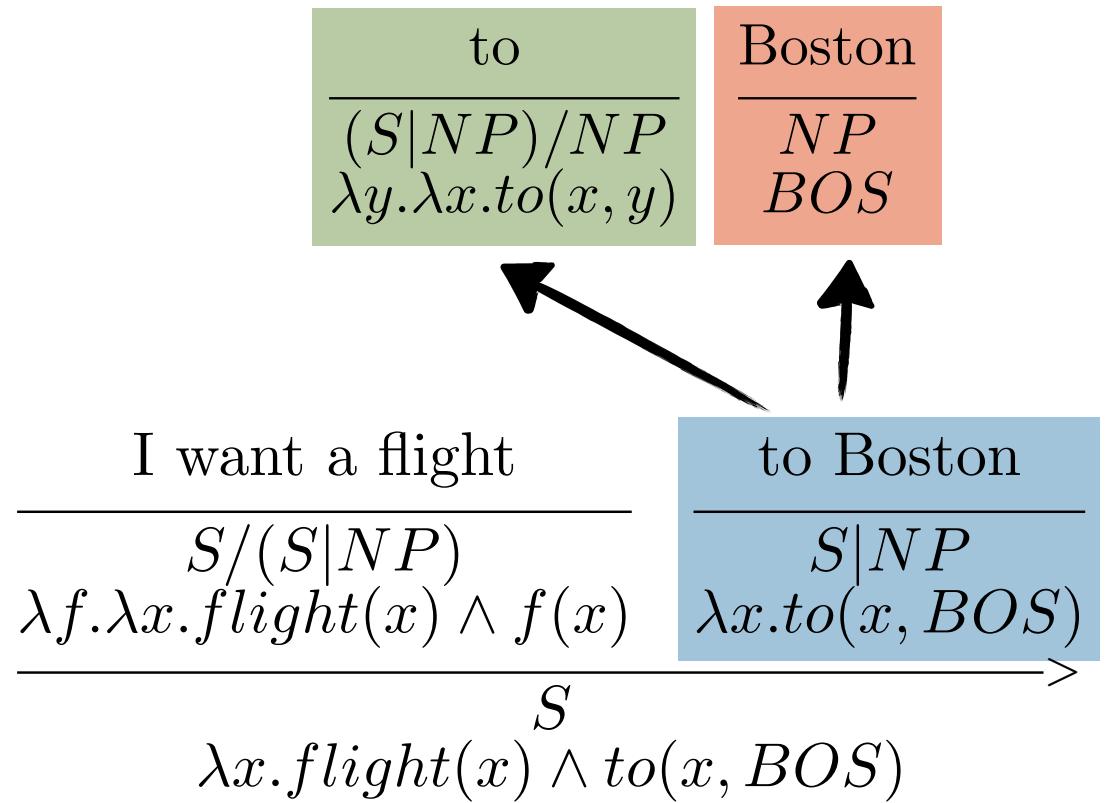
$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

Iteration 2



Unification-based

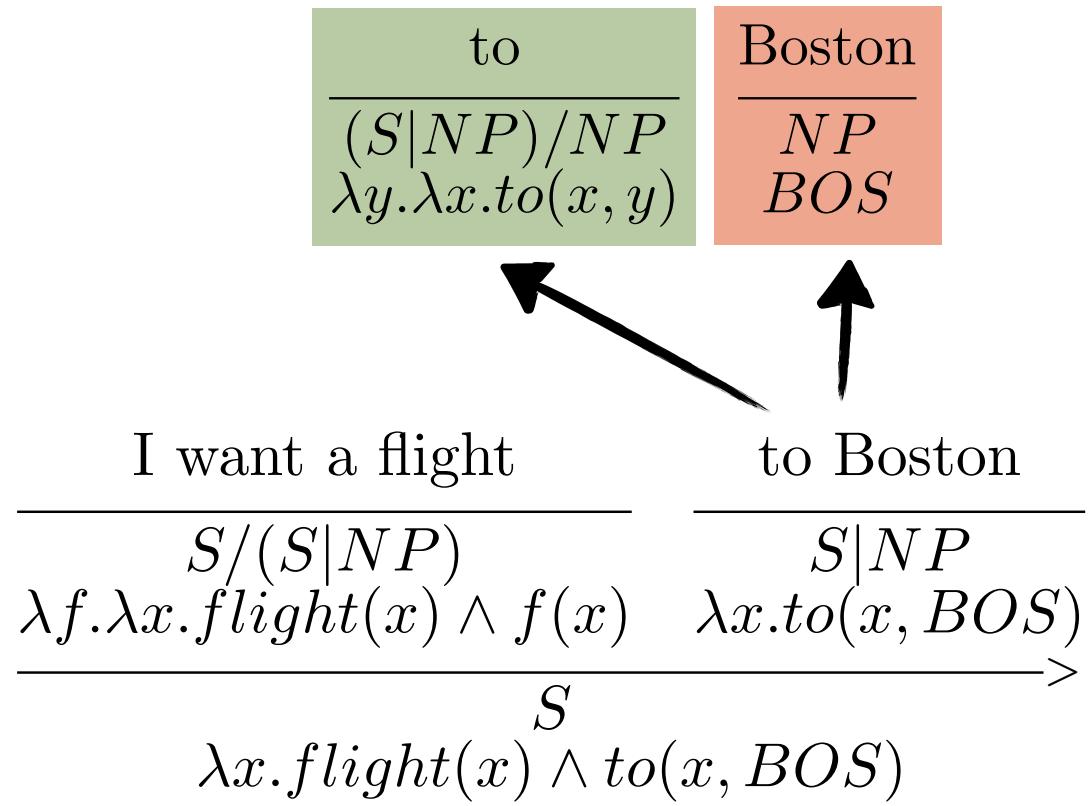
$GENLEX(x, z; \Lambda, \theta)$

I want a flight to Boston

$\lambda x. flight(x) \wedge to(x, BOS)$

1. Find highest scoring correct parse
2. Find splits that most increases score
3. Return new lexical entries

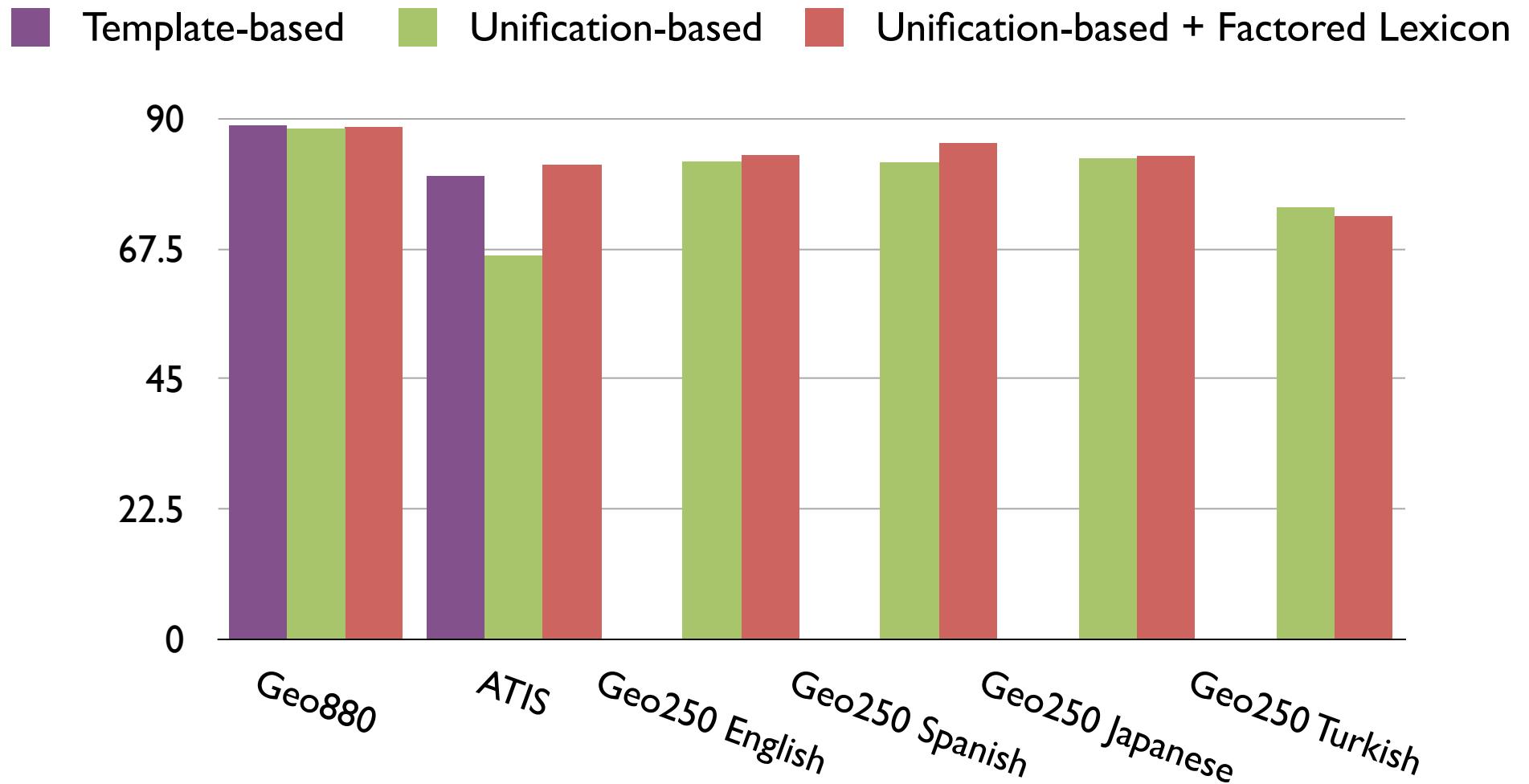
Iteration 2



Experiments

- Two database corpora:
 - Geo880/Geo250 [Zelle and Mooney 1996; Tang and Mooney 2001]
 - ATIS [Dahl et al. 1994]
- Learning from sentences paired with logical forms
- Comparing template-based and unification-based GENLEX methods

Results



[Zettlemoyer and Collins 2007; Kwiatkowski et al. 2010; 2011]

GENLEX Comparison

	Templates	Unification
No initial expert knowledge		✓
Creates compact lexicons	✓	
Language independent		✓
Representation independent		✓
Easily inject linguistic knowledge	✓	
Weakly supervised learning	✓	

GENLEX Comparison

	Templates	Unification
No initial expert knowledge		✓
Creates compact lexicons	✓	
Language independent		✓
Representation independent		✓
Easily inject linguistic knowledge	✓	
Weakly supervised learning	✓	?

Coffee Break

Recap CCGs

$$\frac{\text{CCG}}{\frac{\overline{NP}}{CCG} \quad \frac{\text{is}}{\overline{S \setminus NP / ADJ}} \quad \frac{\text{fun}}{\overline{ADJ}}} \quad \frac{\lambda f. \lambda x. f(x) \quad \lambda x. fun(x)}{\overrightarrow{S \setminus NP}} \\ \overline{\lambda x. fun(x)} \\ \overleftarrow{fun(CCG)}$$

Recap

Unified Learning Algorithm

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \dots T, i = 1 \dots n :$

Step 1: (Lexical generation)

Step 2: (Update parameters)

Output: Parameters θ and lexicon Λ

- Online
- 2 steps:
 - Lexical generation
 - Parameter update

Recap

Learning Choices

Validation Function

$$\mathcal{V} : \mathcal{Y} \rightarrow \{t, f\}$$

- Indicates correctness of a parse y
- Varying \mathcal{V} allows for differing forms of supervision

Lexical Generation Procedure

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

- Given:
 - sentence x
 - validation function \mathcal{V}
 - lexicon Λ
 - parameters θ
- Produce a overly general set of lexical entries

Unified Learning Algorithm

Supervised

\mathcal{V}

Template-based *GENLEX*

Unification-based *GENLEX*

Weakly Supervised

\mathcal{V}

Template-based *GENLEX*

Weak Supervision

What is the largest state that borders Texas?

New Mexico

Weak Supervision

What is the largest state that borders Texas?

New Mexico

at the chair, move forward three steps past the sofa



[Clarke et al. 2010; Liang et al. 2011; Chen and Mooney 2011; Artzi and Zettlemoyer 2013b]

Weak Supervision

What is the largest state that borders Texas?

New Mexico

at the chair, move forward three steps past the sofa



Execute the logical form and observe the result

Weakly Supervised Validation Function

$$\mathcal{V}_i(y) = \begin{cases} \text{true} & \text{if } EXEC(y) \approx e_i \\ \text{false} & \text{else} \end{cases}$$

$y \in \mathcal{Y}$ parse

$e_i \in \mathcal{E}$ available execution result

$EXEC(y) : \mathcal{Y} \rightarrow \mathcal{E}$

logical form at the root of y

Weakly Supervised Validation Function

$$\mathcal{V}_i(y) = \begin{cases} \text{true} & \text{if } EXEC(y) \approx e_i \\ \text{false} & \text{else} \end{cases}$$

Domain-specific execution function:
SQL query engine,
navigation robot

$y \in \mathcal{Y}$ parse

$e_i \in \mathcal{E}$ available execution result

$EXEC(y) : \mathcal{Y} \rightarrow \mathcal{E}$

logical form at the root of y

Weakly Supervised Validation Function

$$\mathcal{V}_i(y) = \begin{cases} \text{true} & \text{if } EXEC(y) \approx e_i \\ \text{false} & \text{else} \end{cases}$$

Domain-specific execution function:
SQL query engine,
navigation robot

$y \in \mathcal{Y}$ parse

$e_i \in \mathcal{E}$ available execution result

$EXEC(y) : \mathcal{Y} \rightarrow \mathcal{E}$

logical form at the root of y

Depends on supervision

Weakly Supervised Validation Function

$$\mathcal{V}_i(y) = \begin{cases} \text{true} & \text{if } EXEC(y) \approx e_i \\ \text{false} & \text{else} \end{cases}$$

Domain-specific execution function:
SQL query engine,
navigation robot

$y \in \mathcal{Y}$ parse

$e_i \in \mathcal{E}$ available execution result

$EXEC(y) : \mathcal{Y} \rightarrow \mathcal{E}$

logical form at the root of y

Depends on supervision

In general: execution function is a natural part of a complete system

Weakly Supervised Validation Function

Example $EXEC(y)$:

Robot moving in an environment

Weakly Supervised Validation Function

Example $EXEC(y)$:

Robot moving in an environment

Example supervision:

Complete
Demonstration



Weakly Supervised Validation Function

Example $EXEC(y)$:

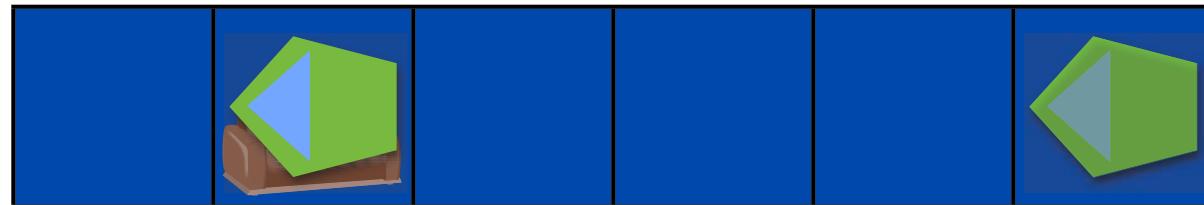
Robot moving in an environment

Example supervision:

Complete
Demonstration



Validate all steps



Weakly Supervised Validation Function

Example $EXEC(y)$:

Robot moving in an environment

Example supervision:

Final State

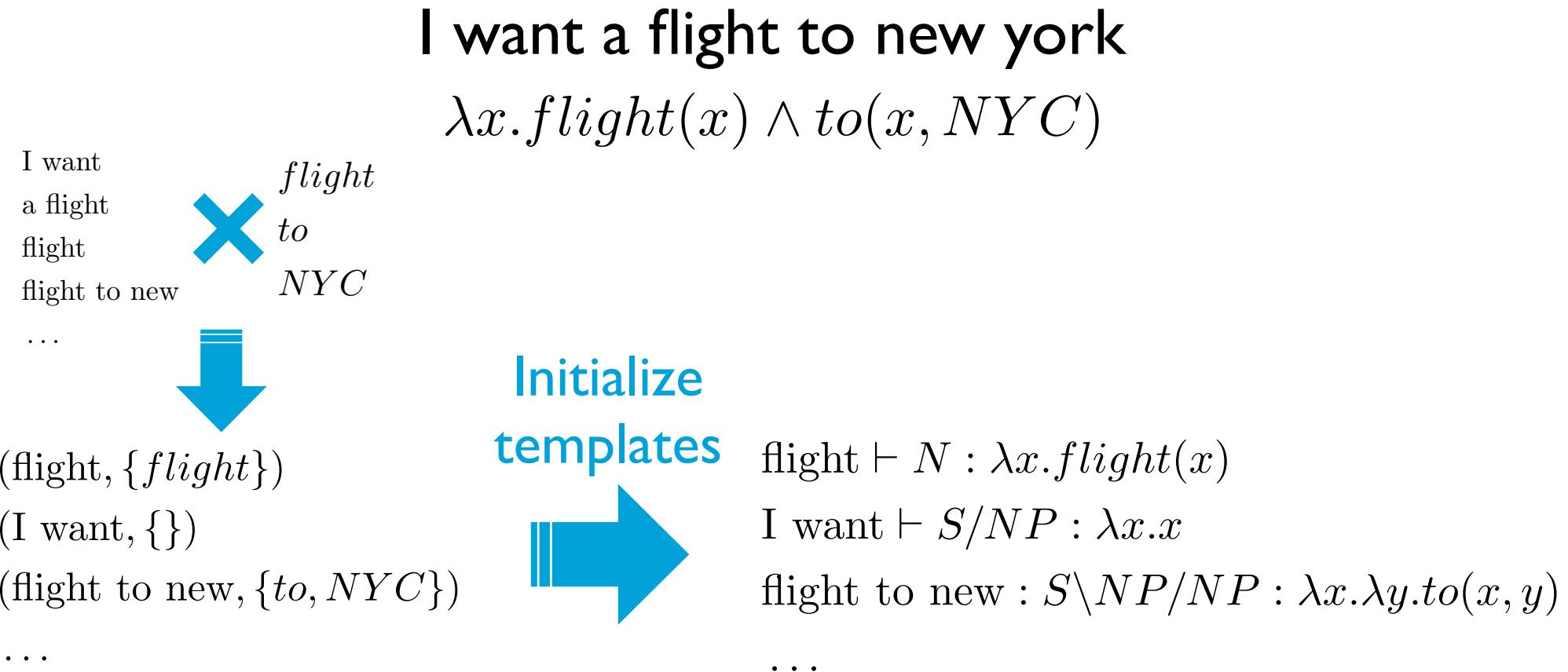


Validate only last
position



Weakly Supervised

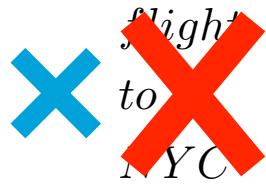
$GENLEX(x, \mathcal{V}; \Lambda, \theta)$



Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

I want
a flight
flight
flight to new



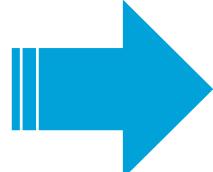
...
(flight, {flight})
(I want, {})
(flight to new, {to, NYC})
...

I want a flight to new york

~~$\lambda x.\text{flight}(x) \wedge \text{to}(x, \text{NYC})$~~

No access to
labeled logical form

Initialize
templates



flight $\vdash N : \lambda x.\text{flight}(x)$
I want $\vdash S/NP : \lambda x.x$
flight to new : $S \setminus NP/NP : \lambda x.\lambda y.\text{to}(x, y)$
...

Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

I want a flight to new york

I want
a flight
flight
flight to new
...



flight, from, to,
ground_transport, dtime, atime,
NYC, BOS, LA, SEA, ...

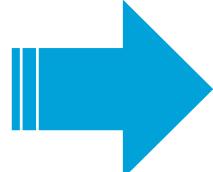


(flight, {*flight*})

(I want, {})

(flight to new, {*to*, NYC})

Initialize
templates



flight $\vdash N : \lambda x. flight(x)$

I want $\vdash S/NP : \lambda x. x$

flight to new : $S \setminus NP/NP : \lambda x. \lambda y. to(x, y)$

...

Use all logical
constants in the
system instead

Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

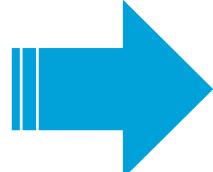
I want a flight to new york

I want
a flight
flight
flight to new
...



*flight, from, to,
ground_transport, dtime, atime,
NYC, BOS, LA, SEA, ...*

Initialize
templates



(flight, {*flight*})
(I want, {})
(flight to new, {*to*, *NYC*})
...

Many more
lexemes

Use all logical
constants in the
system instead

flight $\vdash N : \lambda x. flight(x)$
I want $\vdash S/NP : \lambda x. x$
flight to new : $S \setminus NP/NP : \lambda x. \lambda y. to(x, y)$
...

Huge number of
lexical entries

Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

I want a flight to new york

I want
a flight
flight
flight to new
...
 flight, from, to,
ground_transport, dtime, atime,
NYC, BOS, LA, SEA, ...

(flight, {flight})
(I want, {})
(flight to new, {to, NYC})
...

flight $\vdash N : \lambda x. flight(x)$

I want $\vdash S/NP : \lambda x. x$

flight to new : $S \setminus NP/NP : \lambda x. \lambda y. to(x, y)$

...

Parse to prune
generated lexicon



Huge number of
lexical entries

Model

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

I want a flight to new york

I want
a flight
flight
flight to new
...
 flight, from, to,
ground_transport, dtime, atime,
NYC, BOS, LA, SEA, ...

(flight, {flight})
(I want, {})
(flight to new, {to, NYC})
...

flight $\vdash N : \lambda x. flight(x)$

I want $\vdash S/NP : \lambda x. x$

flight to new : $S \setminus NP/NP : \lambda x. \lambda y. to(x, y)$

...

Parse
generated lexicon




Huge number of
lexical entries

Intractable

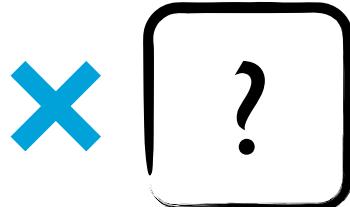
Model

Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

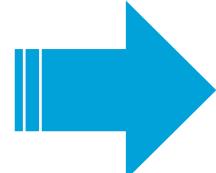
I want a flight to new york

I want
a flight
flight
flight to new
...



(flight, {*flight*})
(I want, {})
(flight to new, {*to*, *NYC*})
...

Initialize
templates



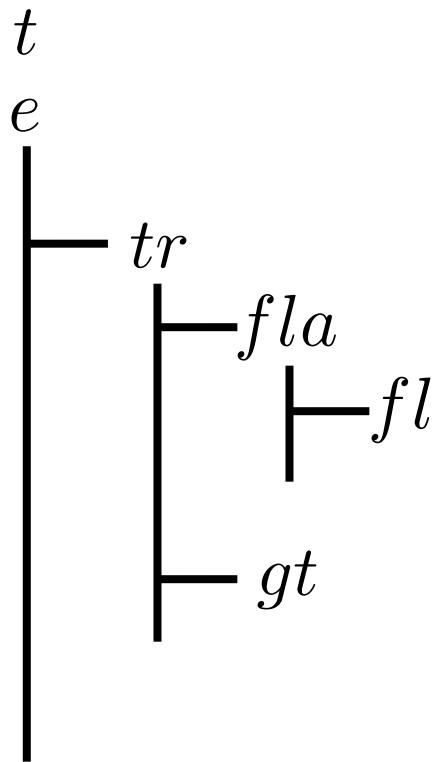
flight $\vdash N : \lambda x. flight(x)$
I want $\vdash S/NP : \lambda x. x$
flight to new : $S \setminus NP/NP : \lambda x. \lambda y. to(x, y)$
...

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

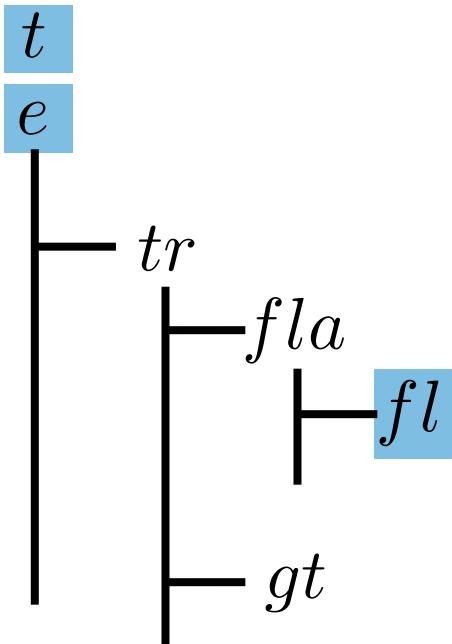
- Gradually prune lexical entries using a coarse-to-fine semantic parsing algorithm
- Transition from coarse to fine defined by typing system

Coarse Ontology

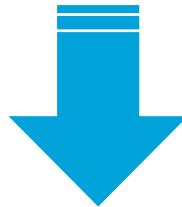


$flight_{<fl,t>} , from_{<fl,<loc,t>>} , to_{<fl,<loc,t>>} ,$
 $ground_transport_{<gt,t>} , dtime_{<tr,<ti,t>>} , atime_{<tr,<ti,t>>} ,$
 $NYC_{ci} , BOS_{ci} , JFK_{ap} , LAS_{ap} , \dots$

Coarse Ontology



$flight_{fl,t}, from_{fl,<loc,t>>}, to_{fl,<loc,t>>},$
 $ground_transport_{gt,t}, dtime_{tr,<ti,t>>}, atime_{tr,<ti,t>>},$
 $NYC_{ci}, BOS_{ci}, JFK_{ap}, LAS_{ap}, \dots$



Generalize types

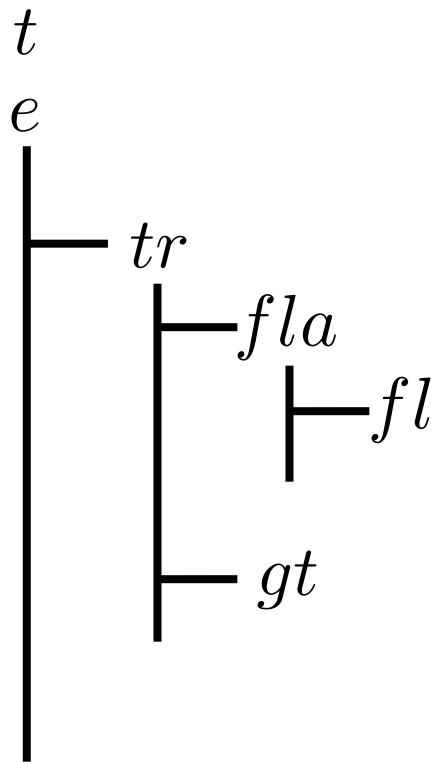
$flight_{fl,t}$

$fl \leftarrow e$
 $t \leftarrow t$

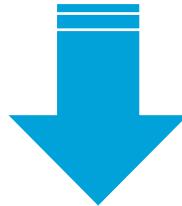
$flight_{e,t}$

$flight_{e,t}, from_{e,<e,t>>}, to_{e,<e,t>>},$
 $ground_transport_{e,t}, dtime_{e,<e,t>>}, atime_{e,<e,t>>},$
 $NYC_e, BOS_e, LA_e, SEA_e, \dots$

Coarse Ontology

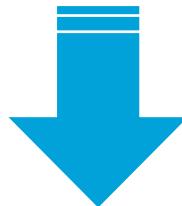


flight_{<fl,t>}, from_{<fl,<loc,t>>}, to_{<fl,<loc,t>>},
ground_transport_{<gt,t>}, dtime_{<tr,<ti,t>>}, atime_{<tr,<ti,t>>},
NYC_{ci}, BOS_{ci}, JFK_{ap}, LAS_{ap}, ...



Generalize types

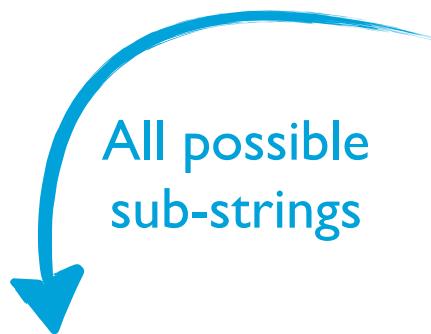
flight_{<e,t>}, from_{<e,<e,t>>}, to_{<e,<e,t>>},
ground_transport_{<e,t>}, dtime_{<e,<e,t>>}, atime_{<e,<e,t>>},
NYC_e, BOS_e, LA_e, SEA_e, ...



**Merge identically
typed constants**

c1_{<e,t>}, c2_{<e,<e,t>>}, c3_e, ...

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$


I want
a flight
flight
flight to new
...

I want a flight to new york

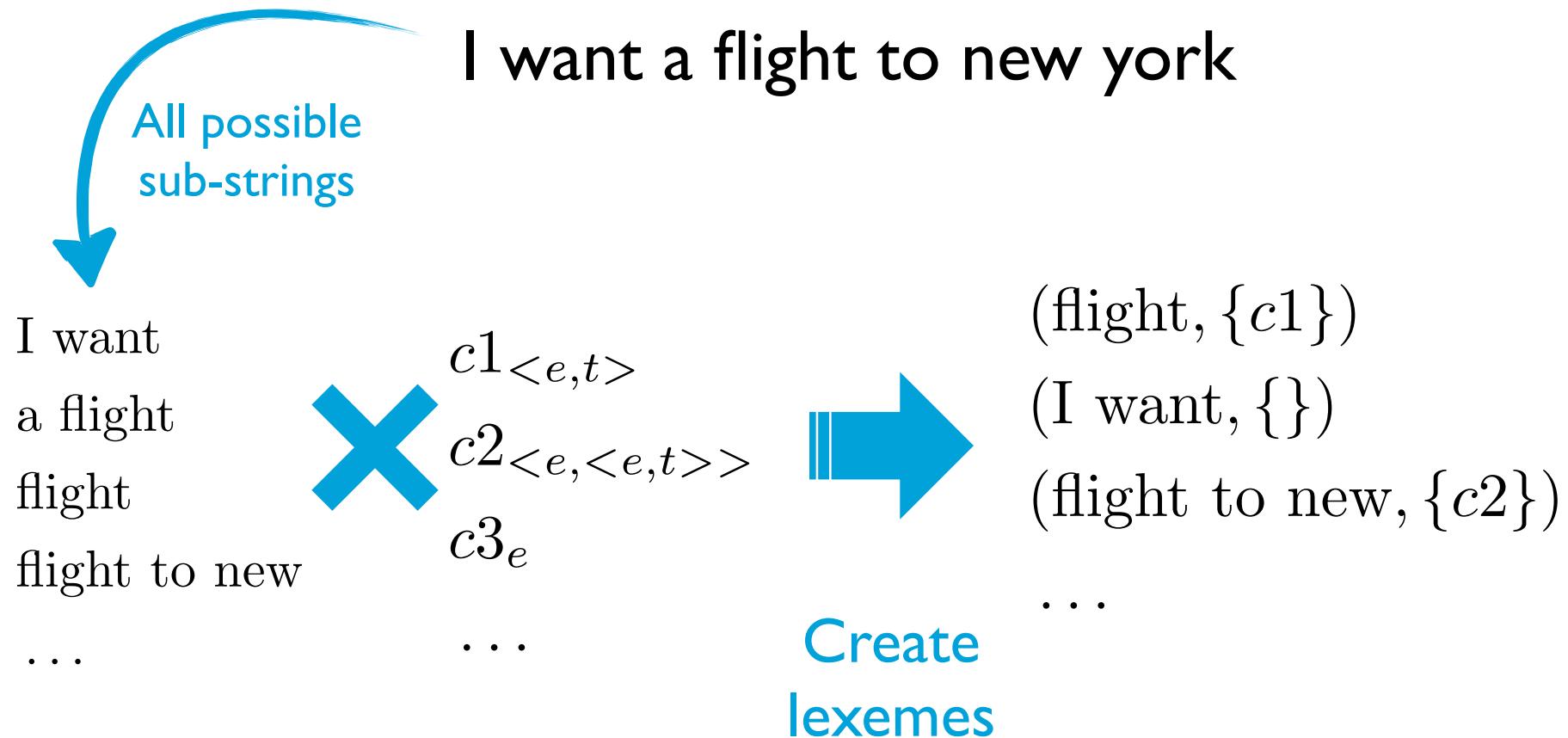
$c1_{\langle e, t \rangle}$

$c2_{\langle e, \langle e, t \rangle \rangle}$

$c3_e$

...

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$


Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

I want a flight to new york

I want
a flight
flight
flight to new

$c1_{<e,t>}$
 ~~$c2_{<e,<e,t>>}$~~
 $c3_e$

... ...



(flight, {c1})

(I want, {})

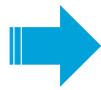
(flight to new, {c2})

...

flight $\vdash N : \lambda x.c1(x)$

I want $\vdash S/NP : \lambda x.x$

flight to new $\vdash S\backslash NP/NP : \lambda x.\lambda y.c2(x, y)$



Initialize ...
templates

Weakly Supervised

$GENLEX(x, \mathcal{V}; \Lambda, \theta)$

I want a flight to new york

I want
a flight
flight
flight to new

$c1_{<e,t>}$

$\times c2_{<e,<e,t>>}$

$c3_e$

...

...

(flight, $\{c1\}$)

(I want, $\{\}$)

(flight to new, $\{c2\}$)

...

Coarse
constants

flight $\vdash N : \lambda x.c1(x)$

I want $\vdash S/NP : \lambda x.x$

flight to new $\vdash S\backslash NP/NP : \lambda x.\lambda y.c2(x, y)$

Initialize ...
templates

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

I want a flight to new york

flight $\vdash N : \lambda x.c1(x)$

Prune by
parsing

I want $\vdash S/NP : \lambda x.x$

flight to new $\vdash S\backslash NP/NP : \lambda x.\lambda y.c2(x, y)$

...

Keep only lexical entries that **participate in complete parses**, which **score higher** than the current best valid parse by a margin

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

I want a flight to new york

flight $\vdash N : \lambda x.c1(x)$

Prune by
parsing

~~I want $\vdash S/NP : \lambda x.x$~~

~~flight to now $\vdash S\backslash NP/NP : \lambda x.\lambda y.c2(x, y)$~~

...

Keep only lexical entries that **participate in complete parses**, which **score higher** than the current best valid parse by a margin

Weakly Supervised

$$GENLEX(x, \mathcal{V}; \Lambda, \theta)$$

I want a flight to new york

flight $\vdash N : \lambda x.c1(x)$

...

Replace all coarse constants with
all similarly typed constants



flight $\vdash N : \lambda x.flight(x)$

flight $\vdash N : \lambda x.ground_transport(x)$

flight $\vdash N : \lambda x.nonstop(x)$

flight $\vdash N : \lambda x.connecting(x)$

...

Weak Supervision Requirements

- Know how to act given a logical form
- A validation function
- Templates for lexical induction

Experiments

Instruction:

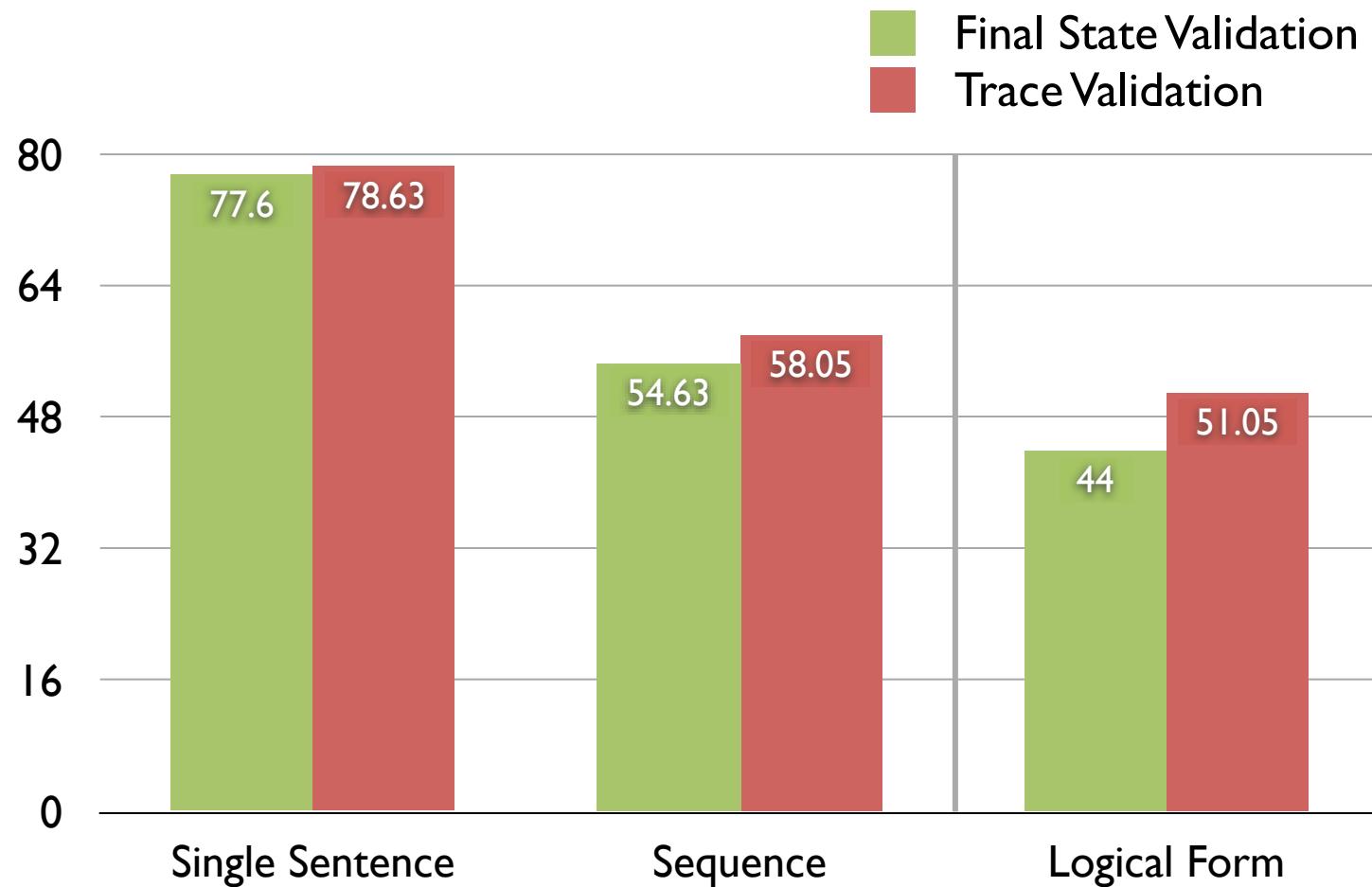
at the chair, move forward three steps past the sofa

Demonstration:



- Situated learning with joint inference
- Two forms of validation
- Template-based $GENLEX(x, \mathcal{V}; \Lambda, \theta)$

Results



Unified Learning Algorithm Extensions

- Loss-sensitive learning
 - Applied to learning from conversations
- Stochastic gradient descent
 - Approximate expectation computation

Parsing

Learning

Modeling

- Structured perceptron
- A unified learning algorithm
- Supervised learning
- Weak supervision

Modeling

Show me all papers about semantic parsing



Parsing with CCG

$$\lambda x. paper(x) \wedge topic(x, SEMPAR)$$

Modeling

Show me all papers about semantic parsing



Parsing with CCG

$$\lambda x. paper(x) \wedge topic(x, SEMPAR)$$

What should these logical forms look like?

But why should we care?

Modeling Considerations

Modeling is key to learning compact lexicons and high performing models

- Capture language complexity
- Satisfy system requirements
- Align with language units of meaning

Parsing

Learning

Modeling

- Semantic modeling for:
 - Querying databases
 - Referring to physical objects
 - Executing instructions

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7
WA	Olympia	4.1
NY	Albany	17.5
IL	Springfield	11.4

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV
CA	AZ

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA
Wrangell	AK
Sill	CA
Bonanza	AK
Elbe	AK



[Zettlemoyer and Collins 2005]

Querying Databases

States			Borders		Mountains	
Abbr.	Capital	Pop.	State1	State2	Name	State
AL	Montgomery	3.9	WA	OR	Bianca	CO
AK	Juneau	0.4	WA	ID	Antero	CO
AZ	Phoenix	2.7	CA	OR	Rainier	WA
			CA	NV	Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Noun Phrases

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Verbs

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the **capital** of Arizona?

How many **states** border California?

What is the largest **state**?

Nouns

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Prepositions

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Superlatives

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Determiners

Querying Databases

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7

Borders

State1	State2
WA	OR
WA	ID
CA	OR
CA	NV

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

What is the capital of Arizona?

How many states border California?

What is the largest state?

Questions

Referring to DB Entities

Noun phrases

Select single DB entities

Prepositions
Verbs

Relations between entities

Nouns

Typing (i.e., column headers)

Superlatives

Ordering queries

Noun Phrases

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Noun phrases name specific entities

Washington

WA

Florida

The Sunshine State

FL

Noun Phrases

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

e-typed
entities

Noun phrases name
specific entities

Washington

WA

WA

Florida

The Sunshine State

FL

FL

Noun Phrases

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Noun phrases name specific entities

Washington

NP
WA

The Sunshine State

NP
FL

Verb Relations

States		Borders	
Abbr.	Capital	State1	State2
AL	Montgomery	WA	OR
AK	Juneau	WA	ID
AZ	Phoenix	CA	OR
WA	Olympia	CA	NV
NY	Albany		
IL	Springfield		

Verbs express relations between entities

Nevada borders California
border(NV, CA)

Verb Relations

States		Borders	
Abbr.	Capital	State1	State2
AL	Montgomery	WA	OR
AK	Juneau	WA	ID
AZ	Phoenix	CA	OR
WA	Olympia	CA	NV
NY	Albany		
IL	Springfield		

Verbs express relations between entities

Nevada borders California
border(NV, CA)

true

Verb Relations

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Nevada
 $\frac{}{NP}$
 $\frac{}{NV}$

borders
 $\frac{}{S \setminus NP / NP}$
 $\lambda x. \lambda y. border(y, x)$

California
 $\frac{}{NP}$
 $\frac{}{CA}$

$\frac{}{S \setminus NP}$
 $\lambda y. border(y, CA)$

$\frac{}{S}$
 $border(NV, CA)$

Nouns

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Nouns are functions
that define entity type

state

$\lambda x.state(x)$

mountain

$\lambda x.mountain(x)$

Nouns

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

$e \rightarrow t$
functions
define sets

Nouns are functions
that define entity type

state

$\lambda x.state(x)$

$\{ \text{WA}, \text{AL}, \text{AK}, \dots \}$

mountain

$\lambda x.mountain(x)$

$\{ \text{BIANCA}, \text{ANTERO}, \dots \}$

Nouns

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Nouns are functions
that define entity type

state

N

$\lambda x.state(x)$

mountain

N

$\lambda x.mountain(x)$

Prepositions

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Prepositional phrases are conjunctive modifiers

mountain in Colorado

Prepositions

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Prepositional phrases are conjunctive modifiers

mountain

$\lambda x.\text{mountain}(x)$

{ BIANCA , ANTERO , RAINIER , ... }

Prepositions

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Prepositional phrases are conjunctive modifiers

mountain in Colorado

$\lambda x.\text{mountain}(x) \wedge$
 $in(x, CO)$

$\{ \text{BIANCA}, \text{ANTERO} \}$

Prepositions

States

Abbr.	Capital	mountain	in	Colorado
AL	Montgomery	N	PP/NP	NP
AK	Juneau	$\lambda x.mountain(x)$	$\lambda y.\lambda x.in(x, y)$	CO
AZ	Phoenix			PP
WA	Olympia		$\lambda x.in(x, CO)$	
NY	Albany			$N \setminus N$
IL	Springfield		$\lambda f.\lambda x.f(x) \wedge in(x, CO)$	f
			N	
		$\lambda x.mountain(x) \wedge in(x, CO)$		

Function Words

States		Borders	
Abbr.	Capital	State1	State2
AL	Montgomery	WA	OR
AK	Juneau	WA	ID
AZ	Phoenix	CA	OR
WA	Olympia	CA	NV
NY	Albany		
IL	Springfield		

Certain words are used to modify syntactic roles

state **that** borders California

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

$\left\{ \text{OR}, \text{NV}, \text{AZ} \right\}$

Function Words

States

Abbr.	Capital	state	that	borders	California
AL	Montgomery	$\frac{N}{NV}$	$\frac{PP/(S\setminus NP)}{\lambda f.f}$	$\frac{S\setminus NP/NP}{\lambda x.\lambda y.border(y,x)}$	$\frac{NP}{CA}$
AK	Juneau				\rightarrow
AZ	Phoenix			$\frac{S\setminus NP}{\lambda y.border(y, CA)}$	\rightarrow
WA	Olympia			$\frac{PP}{\lambda y.border(y, CA)}$	
NY	Albany			$\frac{N\setminus N}{\lambda f.\lambda y.f(y) \wedge border(y, CA)}$	\leftarrow
IL	Springfield			$\frac{N}{\lambda x.state(x) \wedge (x, CA)}$	

Function Words

States		Borders	
Abbr.	Capital	State1	State2
AL	Montgomery	WA	OR
AK	Juneau	WA	ID
AZ	Phoenix	CA	OR
WA	Olympia	CA	NV
NY	Albany		
IL	Springfield		

Certain words are used to modify syntactic roles

- May have other senses with semantic meaning
- May carry content in other domains

Other common function words: which, of, for, are, is, does, please

Definite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Definite determiner
selects the single members
of a set when such exists

$$\iota : (e \rightarrow t) \rightarrow e$$

the mountain in Washington

Definite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Definite determiner
selects the single members
of a set when such exists

$$\iota : (e \rightarrow t) \rightarrow e$$

mountain in Washington

$\lambda x. mountain(x) \wedge in(x, WA)$

{ RAINIER }

Definite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Definite determiner
selects the single members
of a set when such exists

$$\iota : (e \rightarrow t) \rightarrow e$$

the mountain in Washington

$\iota x. mountain(x) \wedge in(x, WA)$



Definite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Definite determiner
selects the single members
of a set when such exists

$$\iota : (e \rightarrow t) \rightarrow e$$

the mountain in Colorado

$\iota x.\text{mountain}(x) \wedge \text{in}(x, CO)$

$\left\{ \text{BIANCA}, \text{ANTERO} \right\} \rightarrow ?$

Definite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Definite determiner
selects the single members
of a set when such exists

$$\iota : (e \rightarrow t) \rightarrow e$$

the mountain in Colorado

$\iota x.\text{mountain}(x) \wedge \text{in}(x, CO)$



No information to disambiguate

Definite Determiners

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

$$\frac{\text{the}}{NP/N}
 \\ \lambda f. \iota x. f(x)$$

mountain in Colorado

.

.

.

$$\frac{}{\lambda x. mountain(x) \wedge in(x, CO)}^N$$

$$\frac{}{\iota x. mountain(x) \wedge in(x, CO)}^{NP}$$

Indefinite Determiners

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Indefinite determiners are select any entity from a set without a preference

$$\mathcal{A} : (e \rightarrow t) \rightarrow e$$

state with a mountain

$$\lambda x. state(x) \wedge \text{in}(\mathcal{A}y. \text{mountain}(y), x)$$

Indefinite Determiners

States	
Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA
Shasta	CA

Indefinite determiners are select any entity from a set without a preference

$$\mathcal{A} : (e \rightarrow t) \rightarrow e$$

state with a mountain

$$\lambda x. state(x) \wedge in(\mathcal{A}y. mountain(y), x)$$



$$\lambda x. state(x) \wedge \exists y. mountain(y) \wedge in(y, x)$$

Exists

Indefinite Determiners

$$\frac{\text{state} \quad \text{with} \quad \begin{array}{c} \text{a} \\ \text{mountain} \end{array}}{\lambda x.state(x) \quad \lambda x.\lambda y.in(x,y) \quad \begin{array}{c} NP/N \\ \lambda f.\mathcal{A}x.f(x) \quad \lambda x.mountain(x) \\ \hline NP \\ \mathcal{A}x.mountain(x) \end{array}}$$

$$\frac{}{\lambda y.(Ax.mountain(x),y) \quad \begin{array}{c} PP \\ \lambda f.\lambda y.f(y) \wedge (\mathcal{A}x.mountain(x),y) \end{array}}$$

$$\frac{}{\lambda y.state(y) \wedge (\mathcal{A}x.mountain(x),y) \quad N}$$

Indefinite Determiners

$$\frac{\frac{\frac{a}{PP \setminus (PP/NP)/N}}{\lambda f. \lambda g. \lambda y. \exists x. g(x, y) \wedge f(x)}}{\frac{\frac{a}{S \setminus NP \setminus (S \setminus NP/NP)/N}}{\lambda f. \lambda g. \lambda y. \exists x. g(x, y) \wedge f(x)}} \rightarrow \frac{\frac{a}{S \setminus (S \setminus NP)/N}}{\lambda f. \lambda g. \lambda y. \exists x. g(x, y) \wedge f(x)}$$

Using the indefinite quantifier simplifies CCG
handling of the indefinite determiner

Superlatives

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7
WA	Olympia	4.1
NY	Albany	17.5
IL	Springfield	11.4

Superlatives select optimal entities according to a measure

the largest state

$\text{argmax}(\lambda x.\text{state}(x), \lambda y.\text{pop}(y))$

Min or max ... over this set ... according to this measure

{ WA, AL,
AK, ... }

AL	3.9
AK	0.4
Seattle	2.7
San Francisco	4.1
NY	17.5
IL	11.4

Superlatives

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4
AZ	Phoenix	2.7
WA	Olympia	4.1
NY	Albany	17.5
IL	Springfield	11.4

Superlatives select optimal entities according to a measure

the largest state

$\text{argmax}(\lambda x.\text{state}(x), \lambda y.\text{pop}(y))$

Min or max ... over this set ... according to this measure

CA

AL	3.9
AK	0.4
Seattle	2.7
San Francisco	4.1
NY	17.5
IL	11.4

Superlatives

States

Abbr.	Capital
AL	Montgomery
AK	Juneau
AZ	Phoenix
WA	Olympia
NY	Albany
IL	Springfield

$$\frac{\text{the largest } NP/N}{\lambda f.\text{argmax}(\lambda x.f(x), \lambda y.\text{pop}(y))} \xrightarrow{\lambda x.\text{state}(x)} NP \text{ argmax}(\lambda x.\text{state}(x), \lambda y.\text{pop}(y))$$

Superlatives

States

Abbr.	Capital			
AL	Montgomery	the most $\lambda g.\lambda f.argmax(\lambda x.f(x), \lambda y.g(y))$	populated $\lambda x.pop(x)$	state $\lambda x.state(x)$
AK	Juneau	$NP/N/N$	N	N
AZ	Phoenix			
WA	Olympia	NP/N $\lambda f.argmax(\lambda x.f(x), \lambda y.pop(y))$		
NY	Albany		NP $argmax(\lambda x.state(x), \lambda y.pop(y))$	
IL	Springfield			

Representing Questions

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4

Borders

State1	State2
WA	OR
WA	ID
CA	OR

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA

Which mountains are in Arizona?

Represent questions as the queries that generate their answers

Representing Questions

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4

Borders

State1	State2
WA	OR
WA	ID
CA	OR

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA

Which mountains are in Arizona?

```
SELECT Name FROM Mountains  
WHERE State == AZ
```

Represent questions as the queries that generate their answers

Reflects the query SQL

Representing Questions

States

Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4

Borders

State1	State2
WA	OR
WA	ID
CA	OR

Mountains

Name	State
Bianca	CO
Antero	CO
Rainier	WA

Which mountains are in Arizona?

$$\lambda x. \text{mountain}(x) \wedge \text{in}(x, AZ)$$

Represent questions as the queries that generate their answers

Reflects the query SQL

Representing Questions

States		
Abbr.	Capital	Pop.
AL	Montgomery	3.9
AK	Juneau	0.4

Borders	
State1	State2
WA	OR
WA	ID
CA	OR

Mountains	
Name	State
Bianca	CO
Antero	CO
Rainier	WA

How many states border California?

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

Represent questions as the queries that generate their answers

Reflects the query SQL

DB Queries

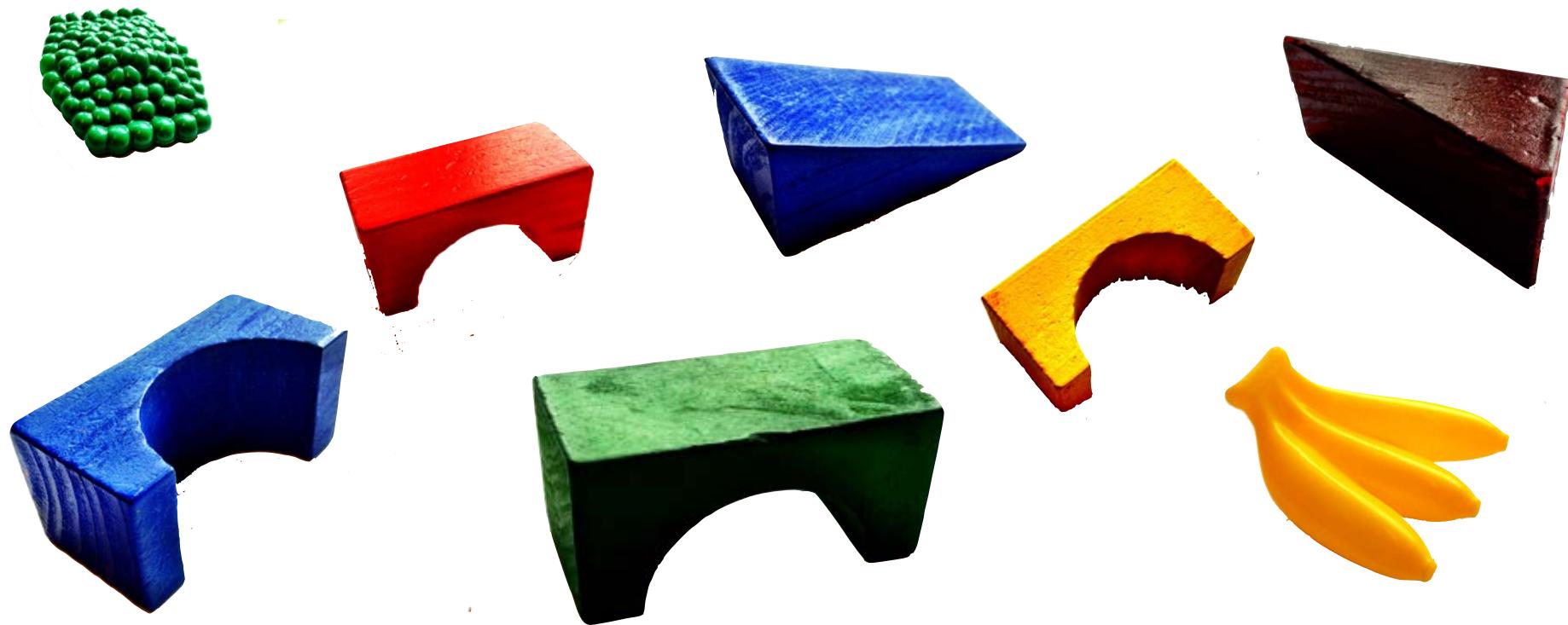
So Far

- Refer to entities in a database
- Query over type of entities, order and other database properties

Next

- How does this approach hold for physical objects?
- What do we need to change? Add?

Referring to Real World Objects



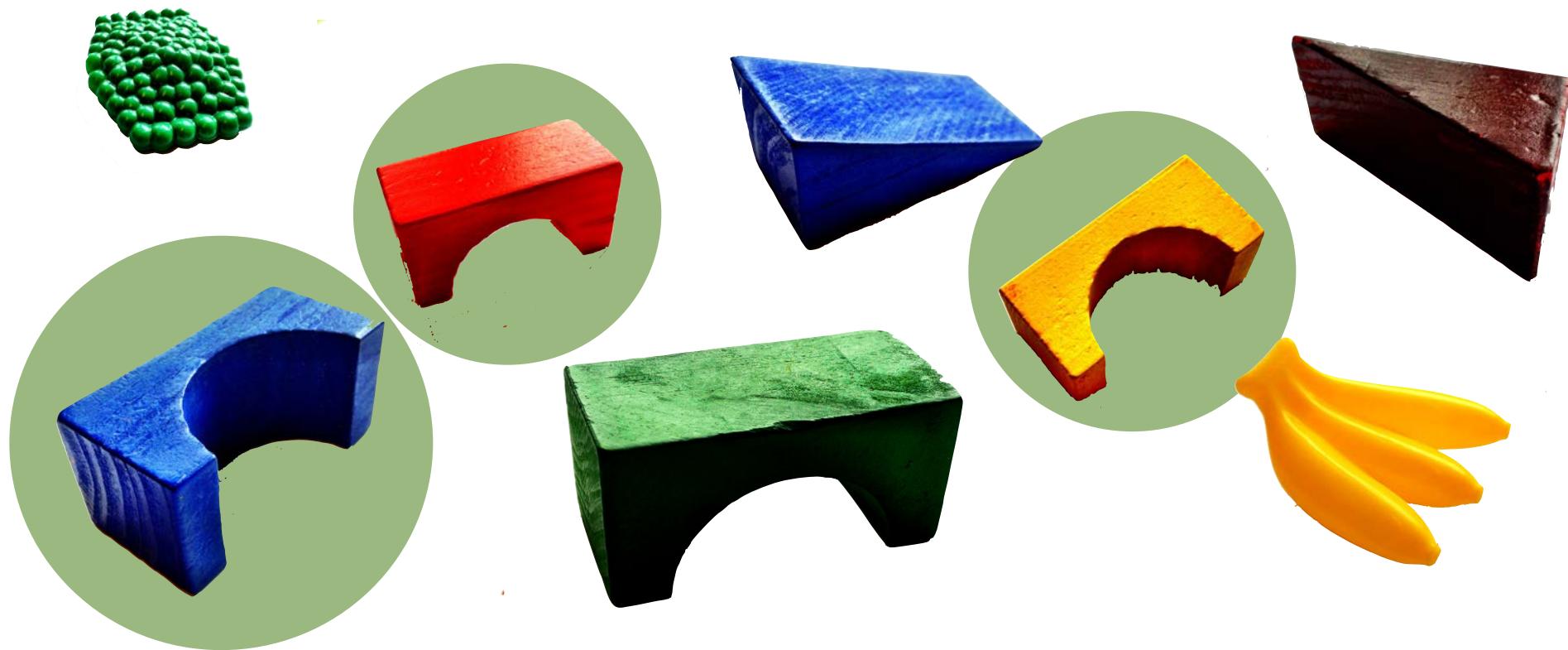
[Matuszek et al. 2012a]

Referring to Real World Objects



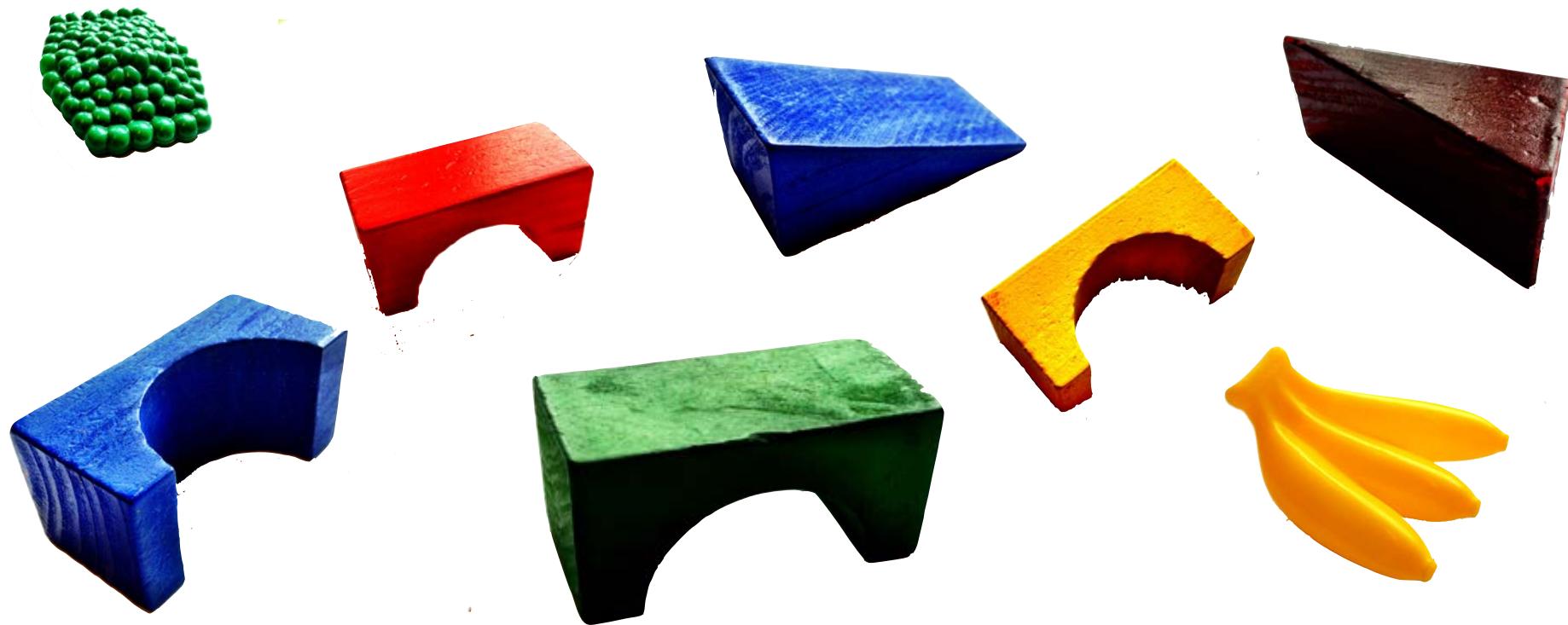
all the arches except the green arch

Referring to Real World Objects



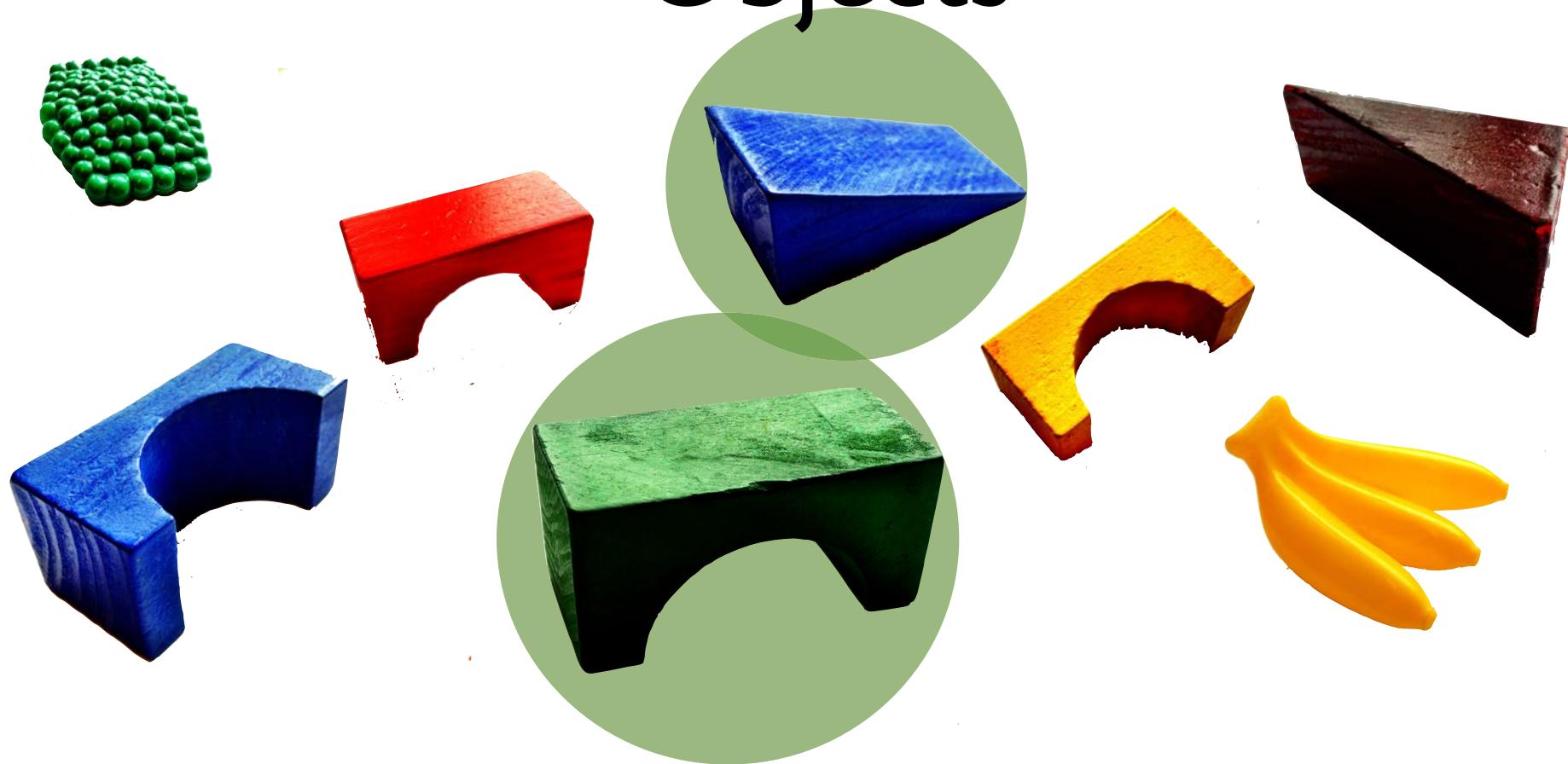
all the arches except the green arch

Referring to Real World Objects



the blue triangle and the green arch

Referring to Real World Objects



the blue triangle and the green arch

Plurality



arches

$\lambda x.\text{arch}(x)$

$\{\textcolor{red}{\text{arch}}, \textcolor{yellow}{\text{arch}}, \textcolor{blue}{\text{arch}}, \textcolor{green}{\text{arch}}\}$

Plurality



arches

$\lambda x.\text{arch}(x)$



the arches

$\iota x.\text{arch}(x)$



Plurality



blue blocks

$\lambda x.\text{blue}(x) \wedge \text{block}(x)$



brown block

$\lambda x.\text{brown}(x) \wedge \text{block}(x)$



Plurality



- All entities are sets
- Space of entities includes singletons and sets of multiple objects

Plurality



- All entities are sets
- Space of entities includes singletons and sets of multiple objects

Cognitive evidence
for sets being a
primitive type

[Scontras et al. 2012]

Plurality



Plurality is a modifier and entities are defined to be sets.

Plurality



Plurality is a modifier and entities are defined to be sets.

arch

$\lambda x. arch(x) \wedge sg(x)$

Plurality



Plurality is a modifier and entities are defined to be sets.

arch

$\lambda x. arch(x) \wedge sg(x)$

$\left\{ \left\{ \textcolor{red}{\text{arch}} \right\}, \left\{ \textcolor{yellow}{\text{arch}} \right\}, \right.$
 $\left. \left\{ \textcolor{green}{\text{arch}} \right\}, \left\{ \textcolor{blue}{\text{arch}} \right\} \right\}$

Plurality



Plurality is a modifier and entities are defined to be sets.

arches

$\lambda x. arch(x) \wedge plu(x)$

$$\left\{ \left\{ \textcolor{red}{\text{arch}}, \textcolor{yellow}{\text{arch}}, \textcolor{blue}{\text{arch}}, \textcolor{green}{\text{arch}} \right\}, \left\{ \textcolor{red}{\text{arch}}, \textcolor{blue}{\text{arch}} \right\}, \dots \right\}$$

Plurality and Determiners



Definite determiner must select a single set. E.g., heuristically select the maximal set.

the arches

$\iota x. arch(x) \wedge plu(x)$

$\left\{ \left\{ \textcolor{red}{\text{r}}, \textcolor{yellow}{\text{r}}, \textcolor{blue}{\text{r}}, \textcolor{green}{\text{r}} \right\} \right\}$

Adjectives



Adjectives are conjunctive
modifiers

blue objects

$\lambda x. blue(x) \wedge obj(x) \wedge plu(x)$

Adjectives



Adjectives are conjunctive
modifiers

blue objects

$\lambda x. blue(x) \wedge obj(x) \wedge plu(x)$

$\left\{ \left\{ \text{blue}, \text{U-block} \right\} \right\}$

DBs and Physical Objects

- Describe and refer to entities
- Ask about objects and relations between them
- Next: move into more dynamic scenarios

States		Borders	
Abbr.	Capital	State1	State2
AL	Montgomery	WA	OR
AK	Juneau	WA	ID
		CA	OR



Beyond Queries

Noun phrases

Specific entities

Nouns

Sets of entities

Prepositional phrases
Adjectives

Constrain sets

Questions

Queries to generate response

Beyond Queries

Noun phrases

Specific entities

Nouns

Sets of entities

Prepositional phrases
Adjectives

Constrain sets

Questions

Queries to generate response

Works well for natural language interfaces for DBs

How can we use this approach for other domains?

Procedural Representations

- Common approach to represent instructional language
- Natural for executing commands

go forward along the stone hall to the intersection with a bare concrete hall

Verify(front : GRAVEL_HALL)

Travel()

Verify(side : CONCRETE_HALL)

Procedural Representations

- Common approach to represent instructional language
- Natural for executing commands

leave the room and go right

*do_seq(verify(room(current_loc)),
move_to(unique_thing($\lambda x. equals(distance(x), 1)$)),
move_to(right_loc))*

Procedural Representations

- Common approach to represent instructional language
- Natural for executing commands

Click Start, point to Search, and the click For Files and Folders. In the Search for box, type “msdownld.tmp”.

LEFT_CLICK(Start)

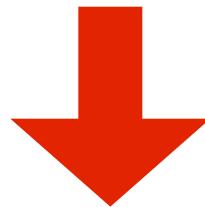
LEFT_CLICK(Search)

...

TYPE_INFO(Search for:, “msdownld.tmp”)

Procedural Representations

Dissonance between structure of
semantics and language



- Poor generalization of learned models
- Difficult to capture complex language

Spatial and Instructional Language

Name objects

Noun phrases

Specific entities

Nouns

Sets of entities

Prepositional phrases

Adjectives

Constrain sets

Instructions to execute

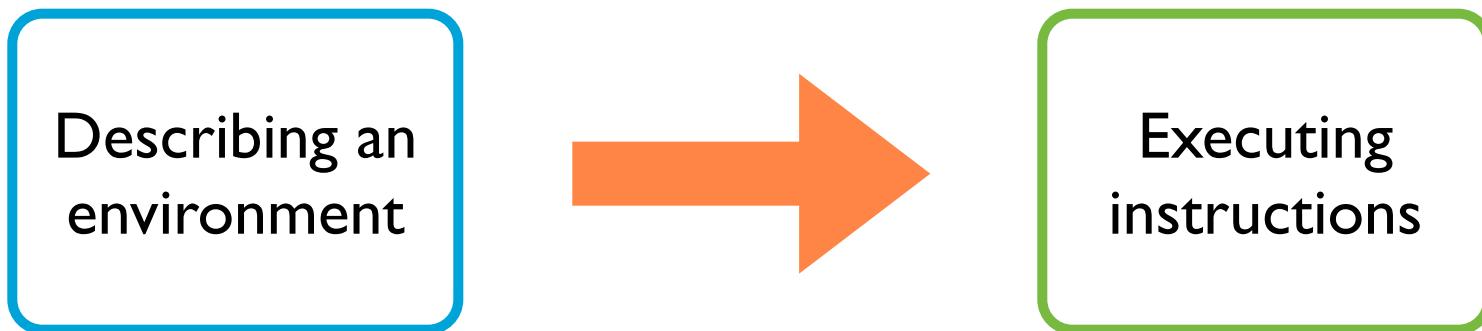
Verbs

Davidsonian events

Imperatives

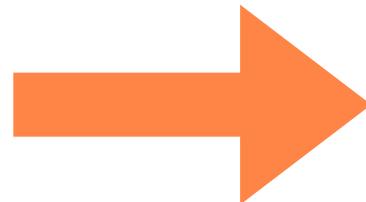
Sets of events

Modeling Instructions



Modeling Instructions

Describing an environment



Executing instructions

Agent



Modeling Instructions



- Model actions and imperatives
- Consider how the state of the agent influences its understanding of language

Modeling Instructions

place your back against the wall of the t intersection

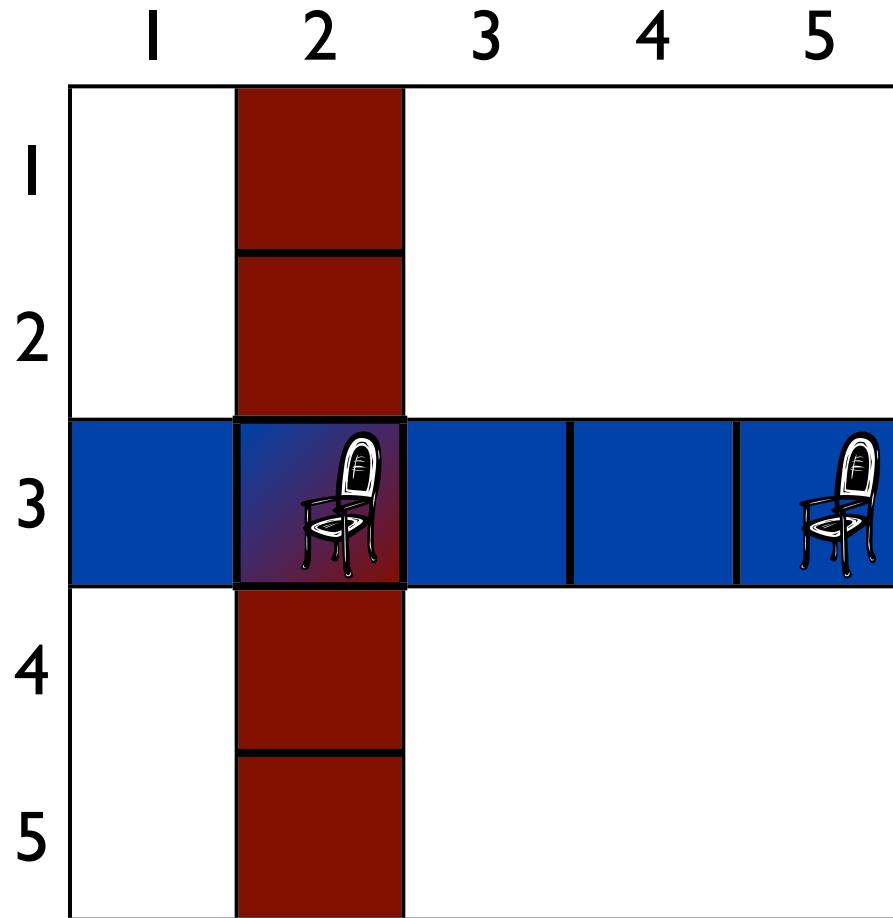
turn left

go forward along the pink flowered carpet hall two segments to the intersection with the brick hall

⋮

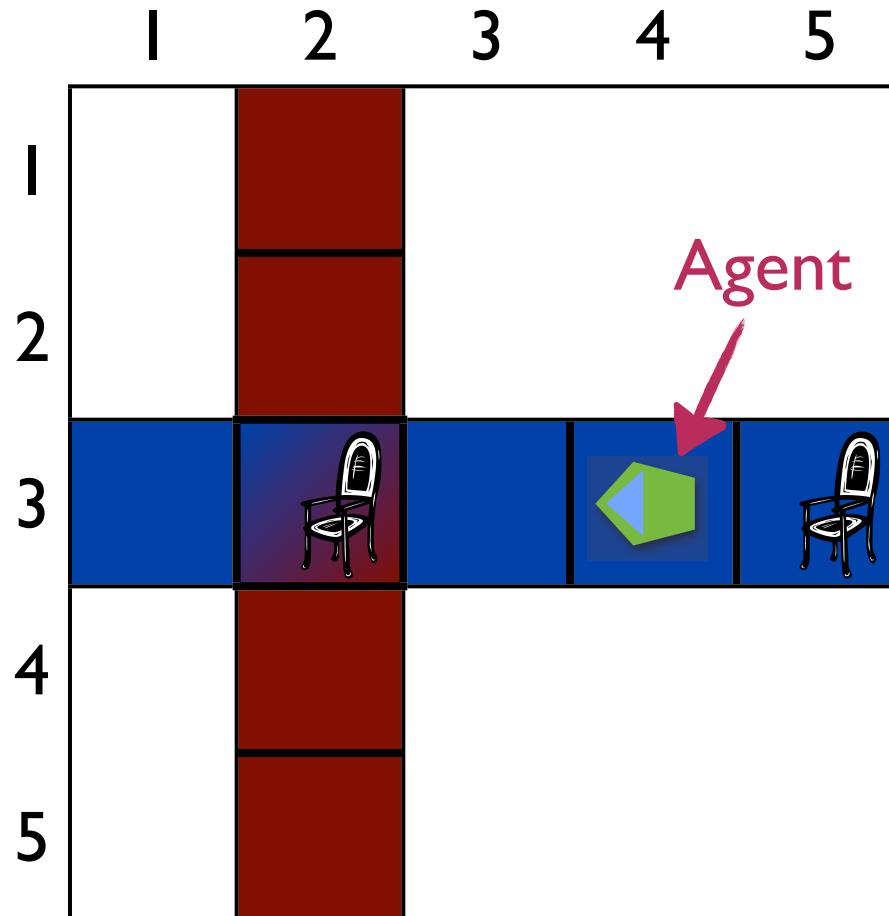


Instructional Environment



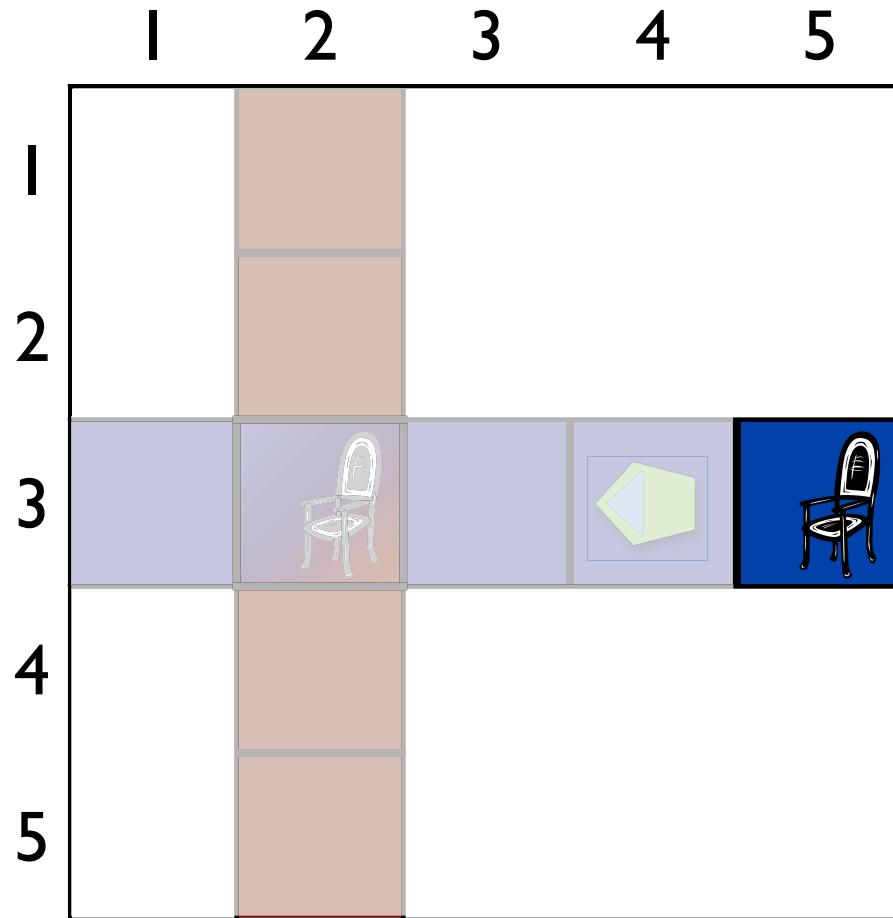
- Maps are graphs of connected positions
- Positions have properties and contain objects

Instructional Environment



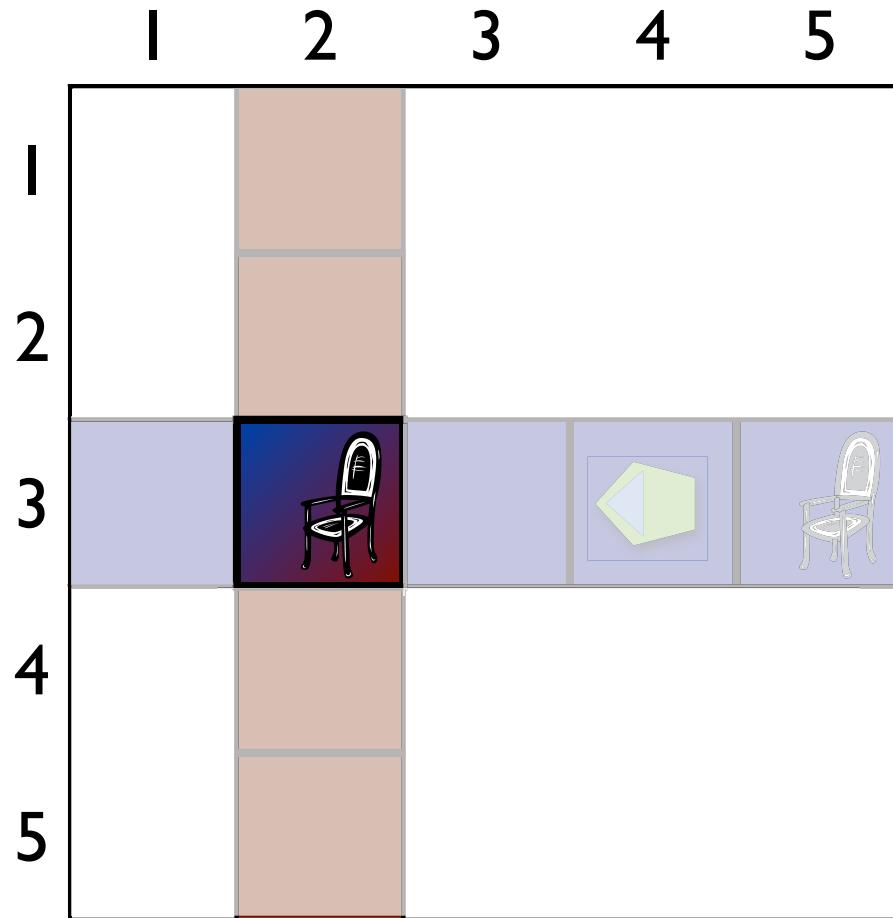
- Agent can move forward, turn right and turn left
- Agent perceives clusters of positions
- Clusters capture objects

Instructional Environment



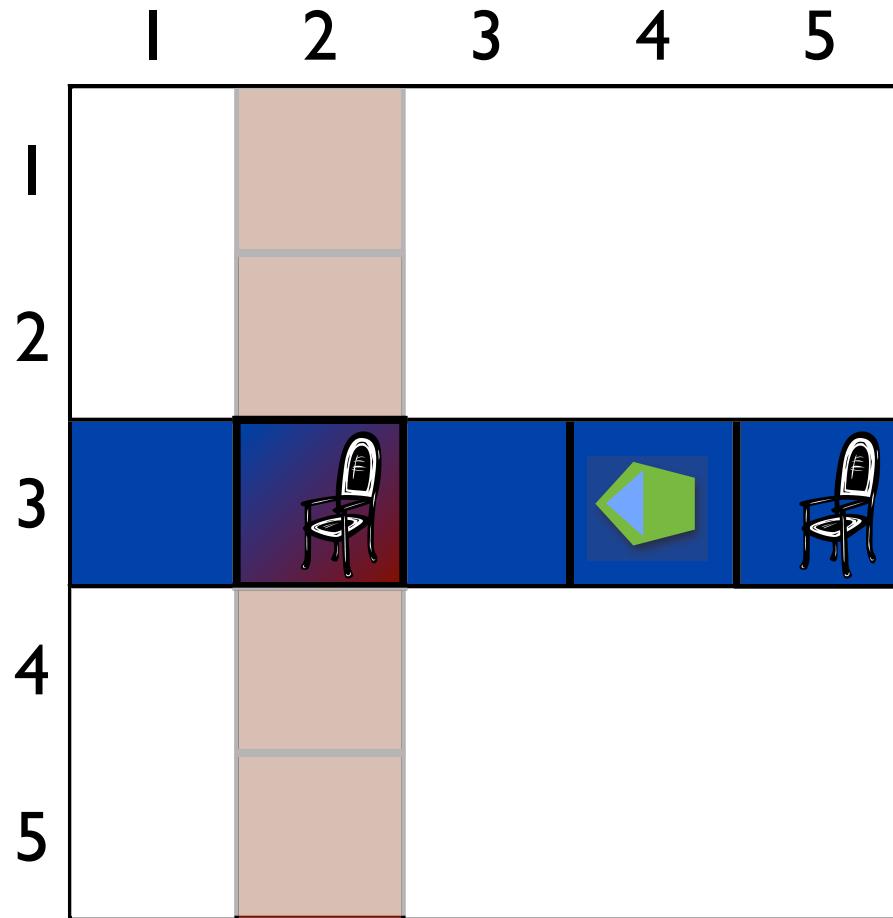
- Agent can move forward, turn right and turn left
- Agent perceives clusters of positions
- Clusters capture objects

Instructional Environment



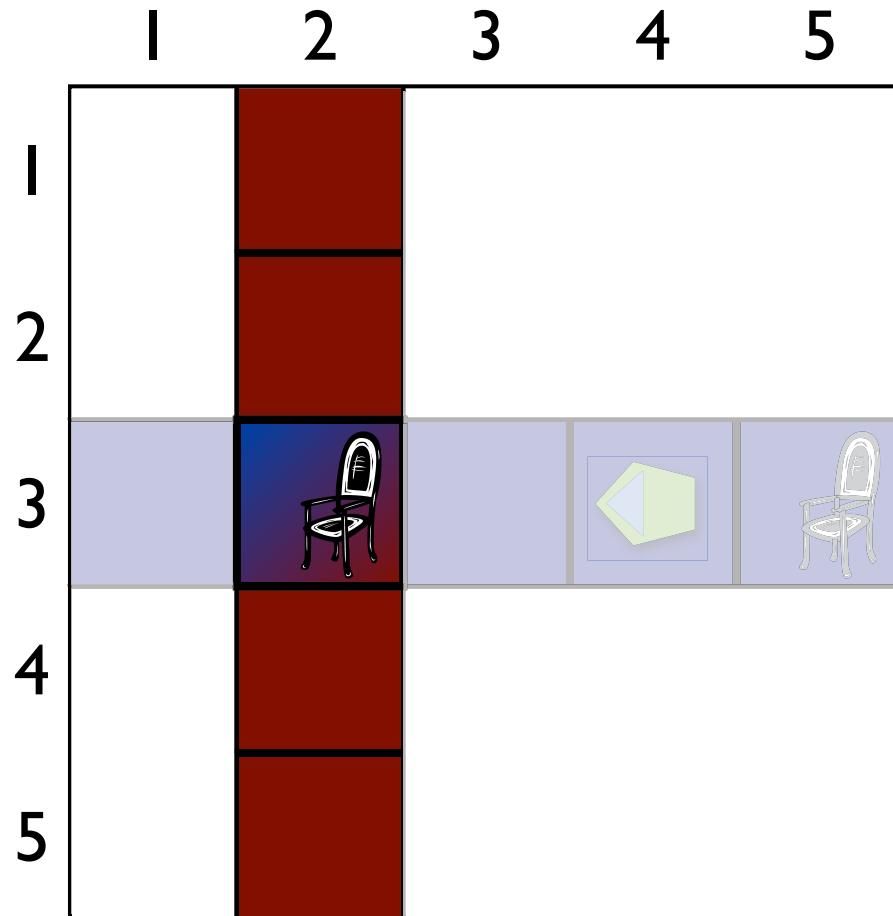
- Agent can move forward, turn right and turn left
- Agent perceives clusters of positions
- Clusters capture objects

Instructional Environment



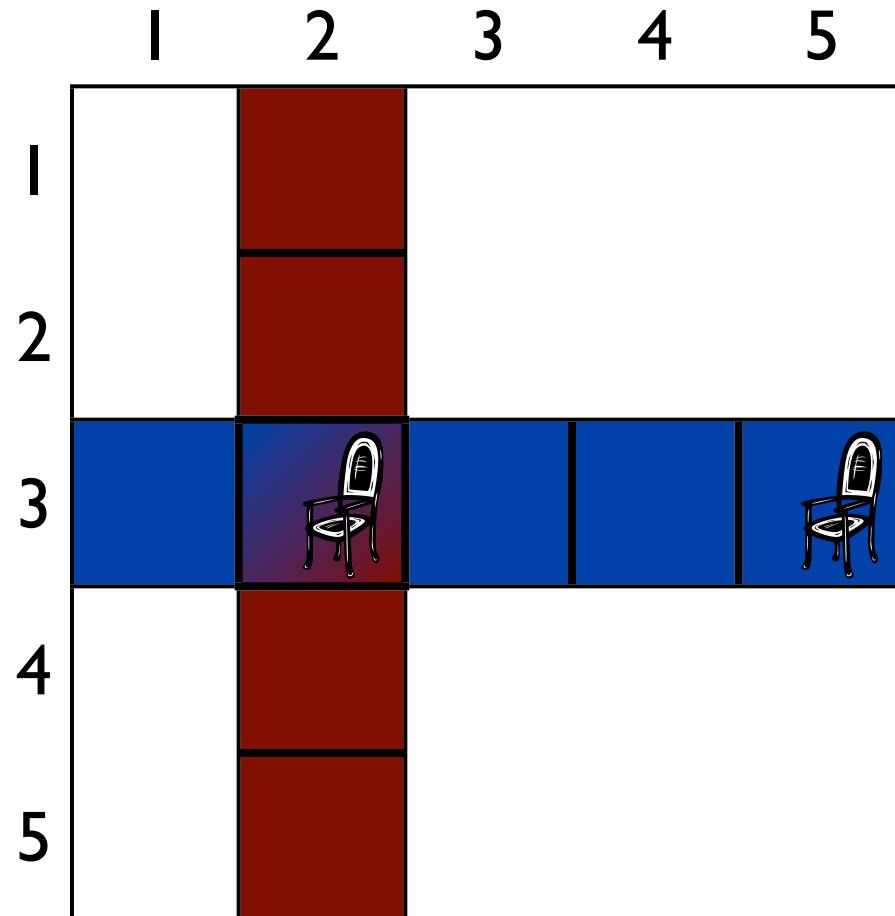
- Agent can move forward, turn right and turn left
- Agent perceives clusters of positions
- Clusters capture objects

Instructional Environment



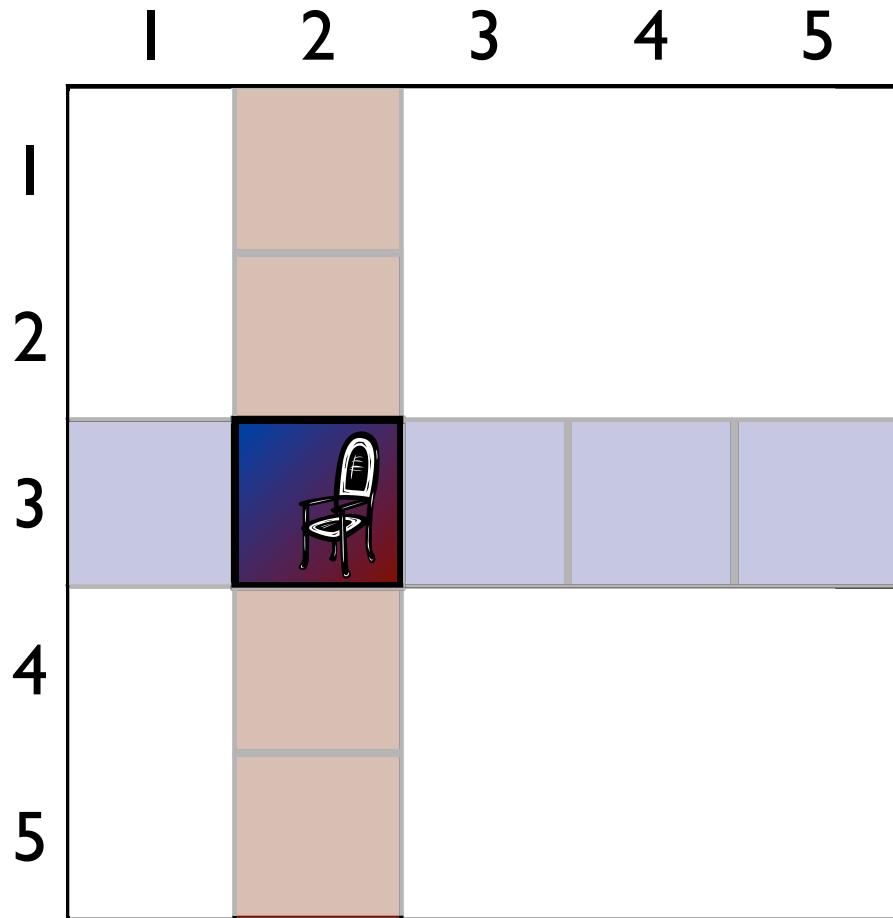
- Agent can move forward, turn right and turn left
- Agent perceives clusters of positions
- Clusters capture objects

Instructional Environment



- Refer to objects similarly to our previous domains
- “Query” the world

Grounded Resolution of Determiners



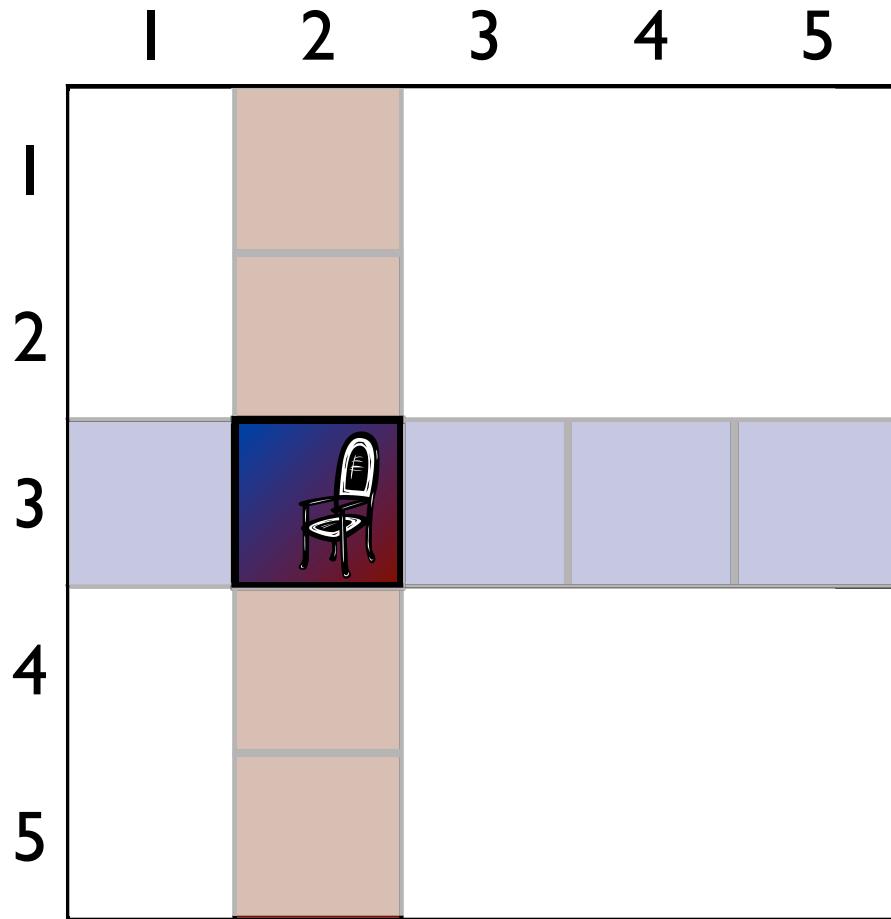
Nouns denote sets of objects

chair

$\lambda x.\text{chair}(x)$

$\left\{ \begin{array}{c} \text{blue} \\ \text{icon} \\ \text{of} \\ \text{a} \\ \text{chair} \end{array} \right\}$

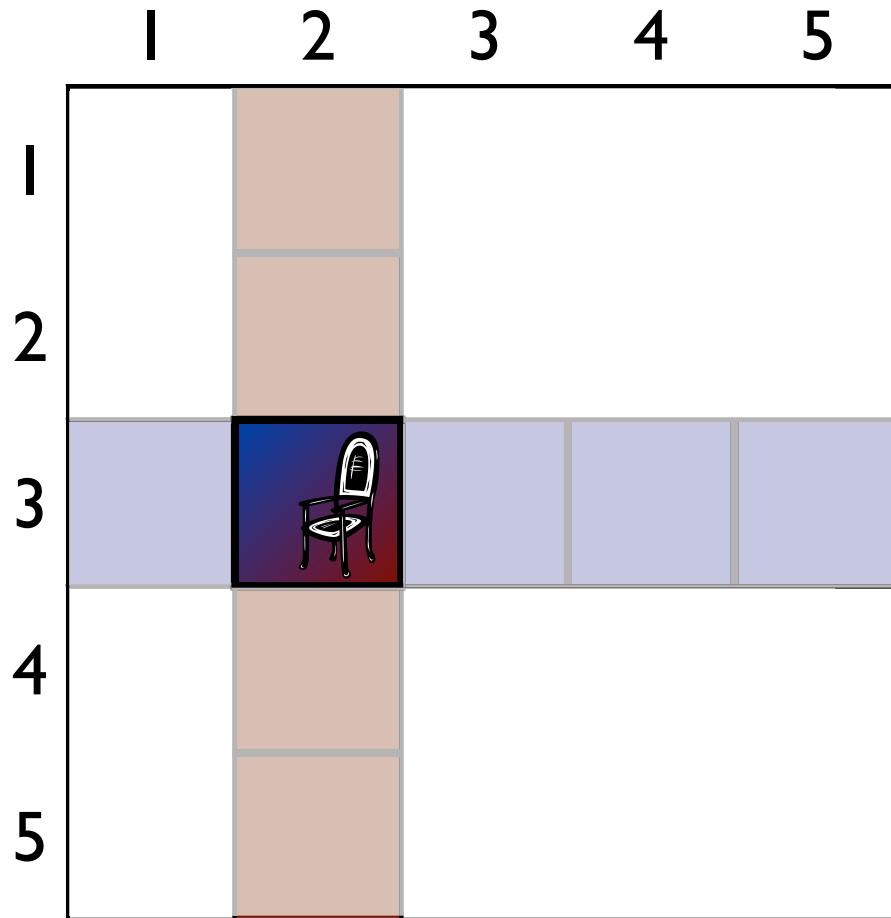
Grounded Resolution of Determiners



Definite determiner
selects a single entity

the chair
 $\iota x.chair(x)$

Grounded Resolution of Determiners

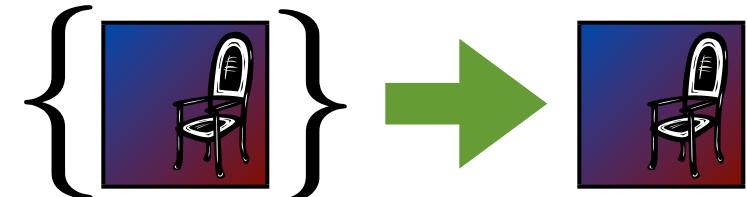


Definite determiner
selects a single entity

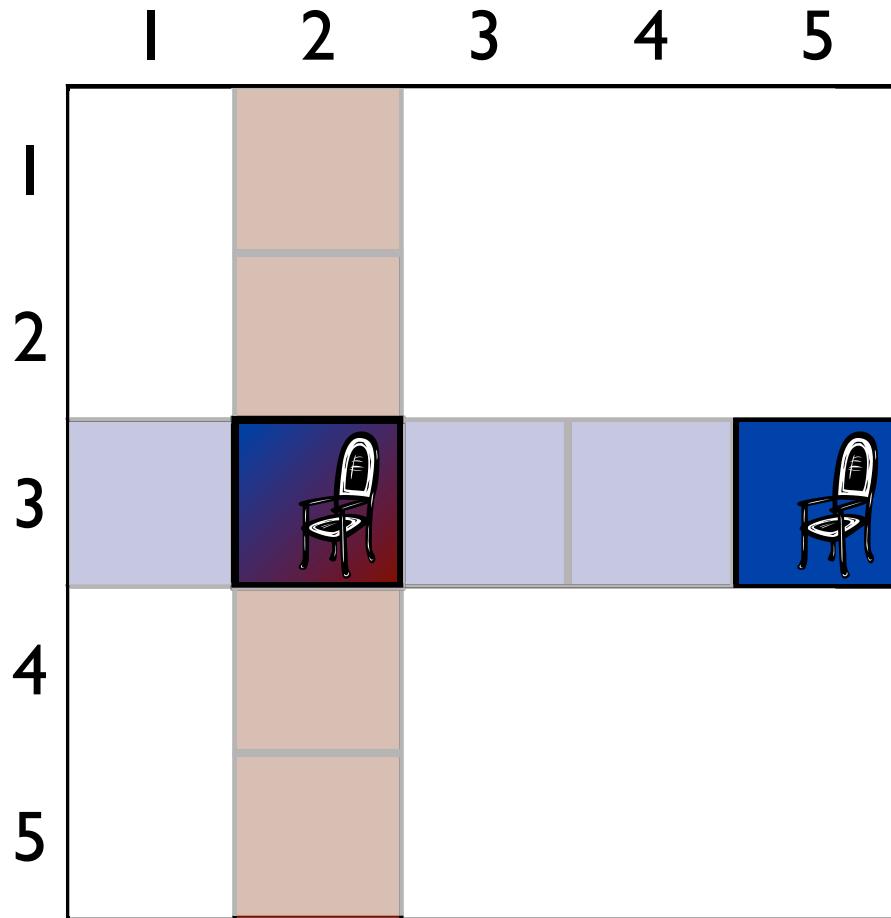
the chair

$\iota x.chair(x)$

$$\iota : (e \rightarrow t) \rightarrow e$$



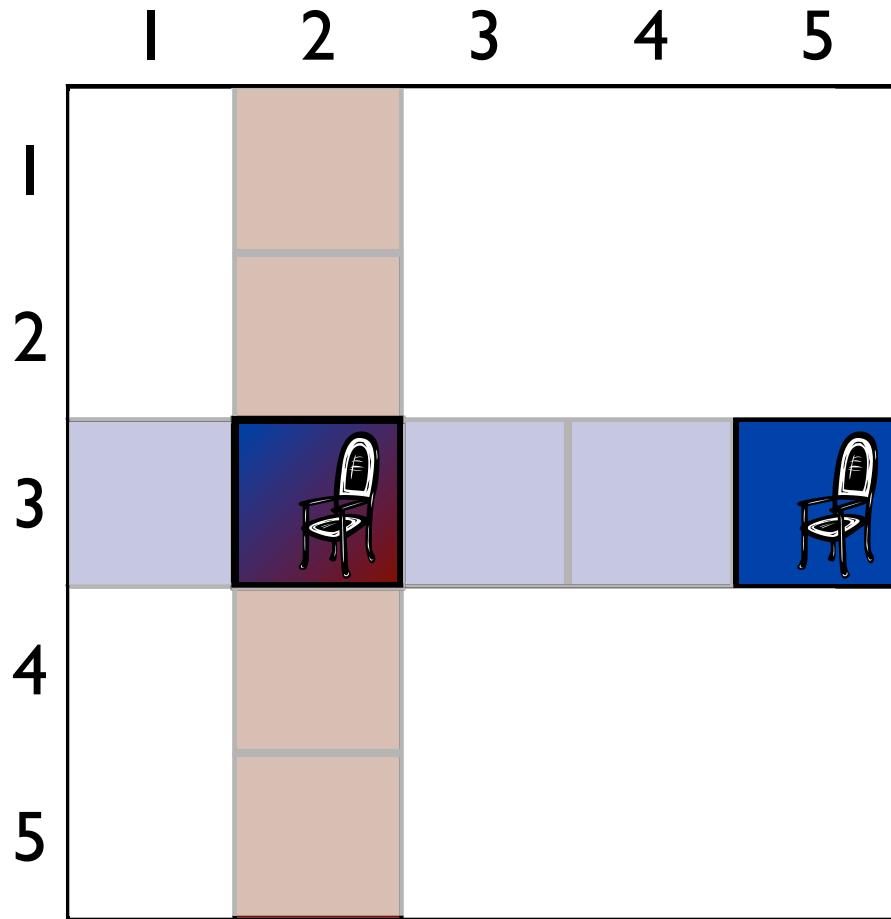
Grounded Resolution of Determiners



Definite determiner
selects a single entity

the chair
 $\iota x.chair(x)$

Grounded Resolution of Determiners

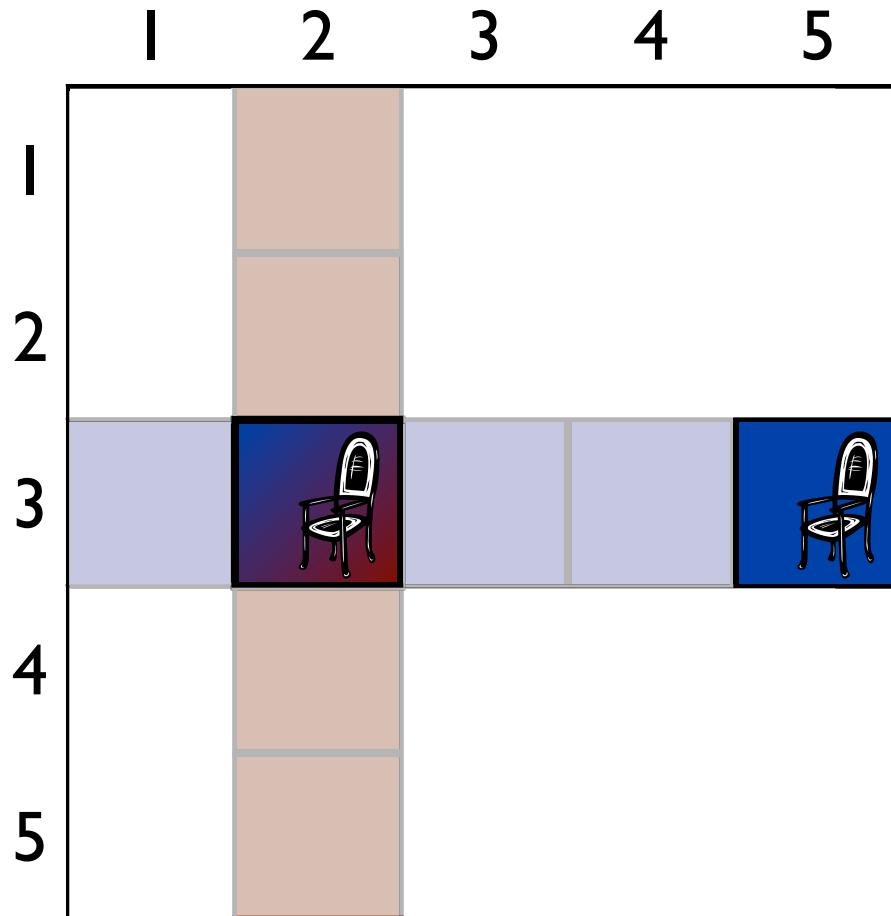


Definite determiner
selects a single entity

the chair
 $\iota x.chair(x)$

Fail?

Grounded Resolution of Determiners

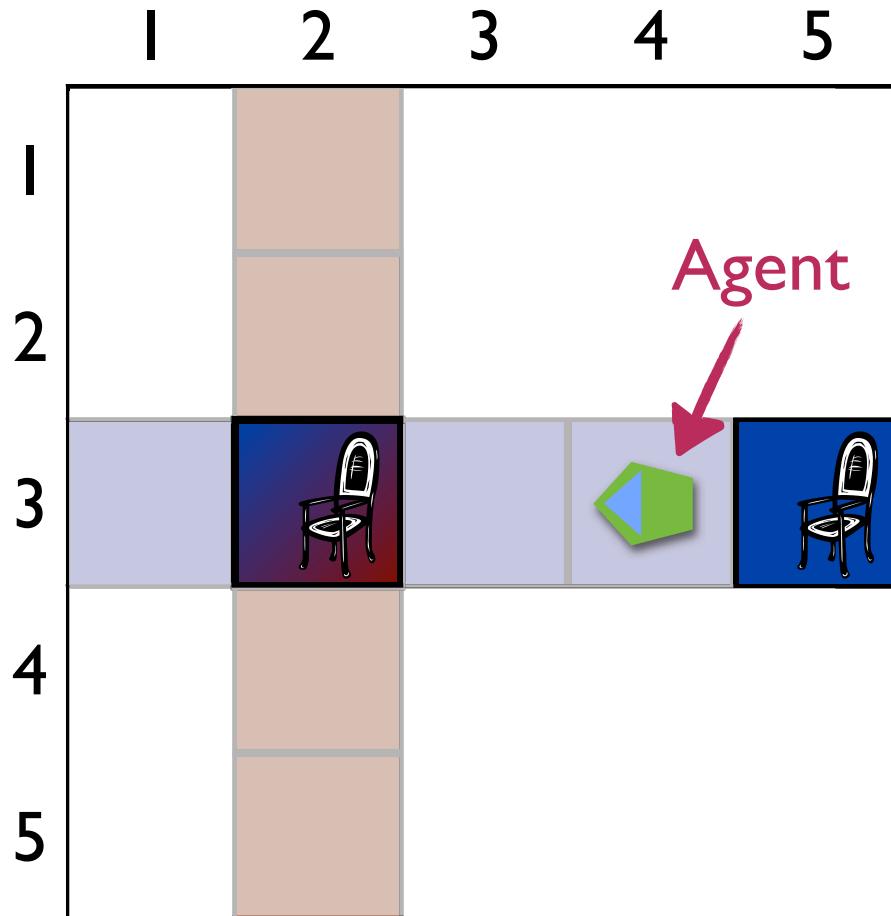


Definite determiner
selects a single entity

the chair
 $\iota x.\text{chair}(x)$

Must disambiguate to
select a single entity

Grounded Resolution of Determiners



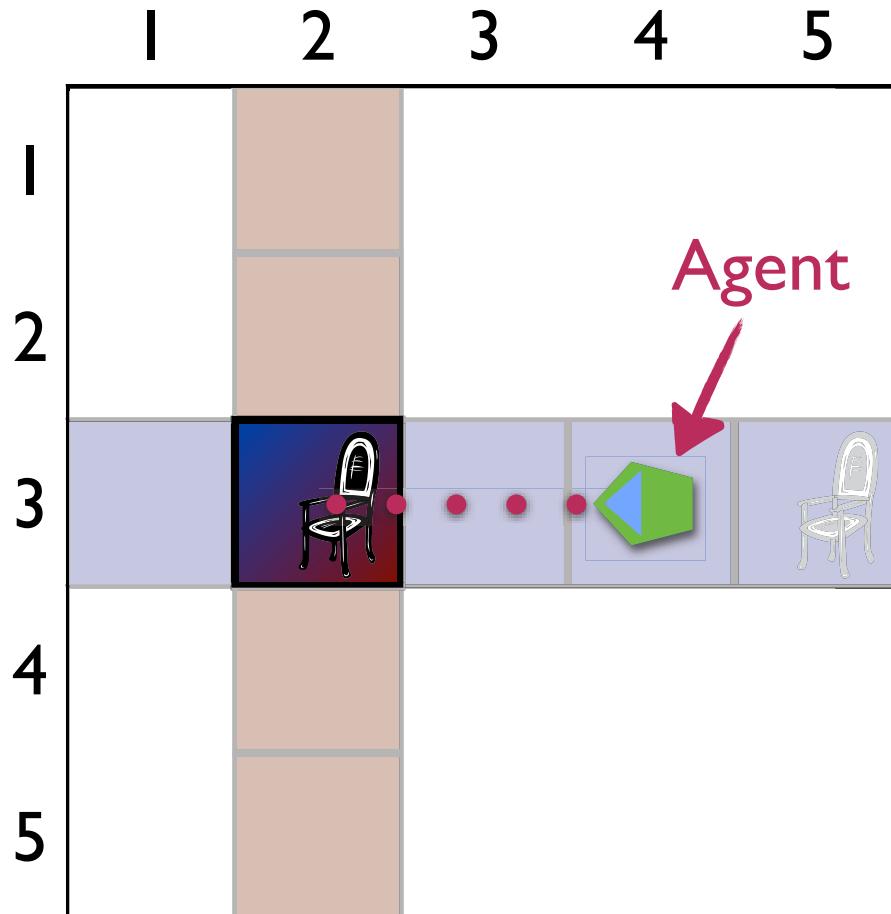
Definite determiner
selects a single entity

the chair

$\iota x.\text{chair}(x)$

Definite determiner
depends on agent state

Grounded Resolution of Determiners

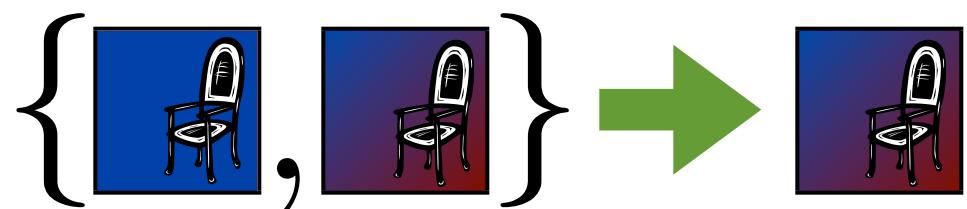


Definite determiner
selects a single entity

the chair

$\iota x.chair(x)$

Definite determiner
depends on agent state



Modeling Instructions

Events taking
place in the
world

Events refer to
environment

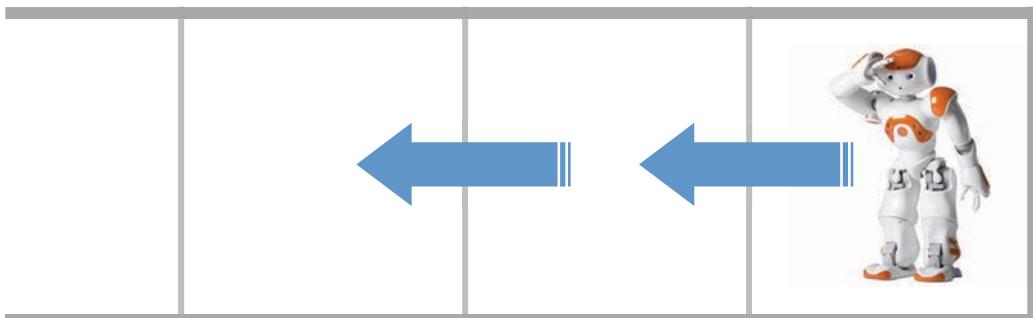
Implicit
requests

Modeling Instructions

Events taking place in the world

Events refer to environment

Implicit requests



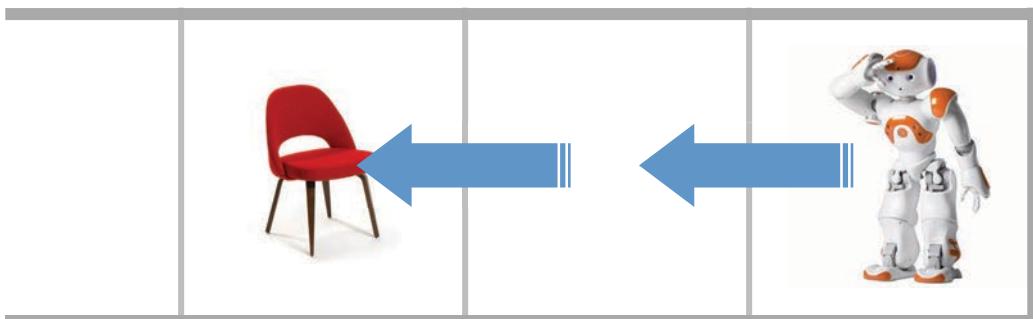
walk forward twice

Modeling Instructions

Events taking place in the world

Events refer to environment

Implicit requests



move twice to the chair

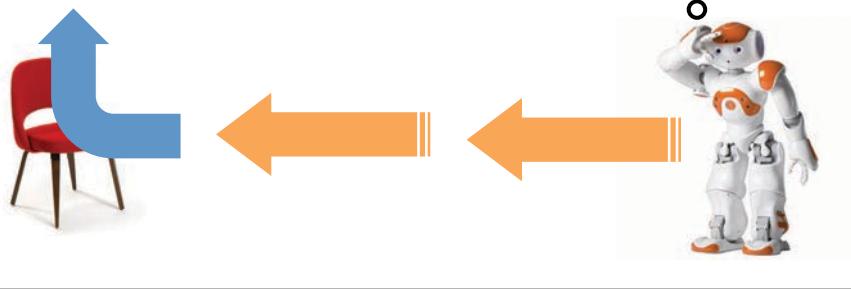
Modeling Instructions

Events taking place in the world

Events refer to environment

Implicit requests

need to move first



at the chair, turn right

Davidsonian Event Semantics

- Actions in the world are constrained by adverbial modifiers
- The number of such modifiers is flexible

Adverbial modification is thus seen to be logically on a par with adjectival modification: what adverbial clauses modify is not verbs, but the events that certain verbs introduce.

Davidson 1969 (quoted in Maienborn et al. 2010)

[Davidson 1967]

Davidsonian Event Semantics

- Use event variable to represent events
- Verbs describe events like nouns describe entities
- Adverbials are conjunctive modifiers

Vincent shot Marvin in the car accidentally

$$\exists a. shot(a, VINCENT, MARVIN) \wedge \\ in(a, \iota x. car(x)) \wedge \neg intentional(a)$$

Neo-Davidsonian Event Semantics

Active

Vincent shot Marvin

$$\exists a. shot(a, VINCENT, MARVIN)$$

Neo-Davidsonian Event Semantics

Active

Vincent shot Marvin

$$\exists a. shot(a, VINCENT, MARVIN)$$

Passive

Marvin was shot by Vincent

Neo-Davidsonian Event Semantics

Active

Vincent shot Marvin

$\exists a. shot(a, VINCENT, MARVIN)$

Passive

Marvin was shot (~~by Vincent~~)

Agent
optional in
passive

Neo-Davidsonian Event Semantics

Active

Vincent shot Marvin

$$\exists a. shot(a, VINCENT, MARVIN)$$

Passive

Marvin was shot (~~by Vincent~~)

$$\exists a. shot(a, MARVIN)$$

Agent
optional in
passive

Neo-Davidsonian Event Semantics

Active

Vincent shot Marvin

$$\exists a. shot(a, VINCENT, MARVIN)$$

Passive

Marvin was shot (~~by Vincent~~)

$$\exists a. shot(a, MARVIN)$$

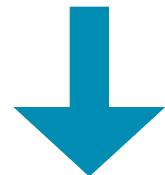
Agent
optional in
passive

Can we represent such distinctions without requiring different arity predicates?

Neo-Davidsonian Event Semantics

- Separation between semantic and syntactic roles
- Thematic roles captured by conjunctive predicates

Vincent shot Marvin

$$\exists a. shot(a, VINCENT, MARVIN)$$

$$\exists a. shot(a) \wedge agent(a, VINCENT) \wedge patient(a, MARVIN)$$

Neo-Davidsonian Event Semantics

Vincent shot Marvin in the car accidentally

$$\exists a. shot(a) \wedge agent(a, VINCENT) \wedge \\ patient(a, MARVIN) \wedge in(a, \iota x. car(x)) \wedge \neg intentional(a)$$

- Decomposition to conjunctive modifiers makes incremental interpretation simpler
- Shallow semantic structures: no need to modify deeply embedded variables

Neo-Davidsonian Event Semantics

$$\exists a. shot(a) \wedge agent(a, VINCENT) \wedge \\ patient(a, MARVIN) \wedge in(a, \iota x. car(x)) \wedge \neg intentional(a)$$

Without events:

$$shot(VINCENT, MARVIN, \iota x. car(x), INTENTIONAL)$$

- Decomposition to conjunctive modifiers makes incremental interpretation simpler
- Shallow semantic structures: no need to modify deeply embedded variables

Representing Imperatives

move forward past the sofa to the chair

Representing Imperatives

move forward past the sofa to the chair

Representing Imperatives



Representing Imperatives



- Imperatives define actions to be executed
- Constrained by adverbials
- Similar to how nouns are defined

Representing Imperatives



- Imperatives are sets of actions
- Just like nouns: functions from events to truth

$$f : ev \rightarrow t$$

Representing Imperatives



Given a set, what do we **actually** execute?

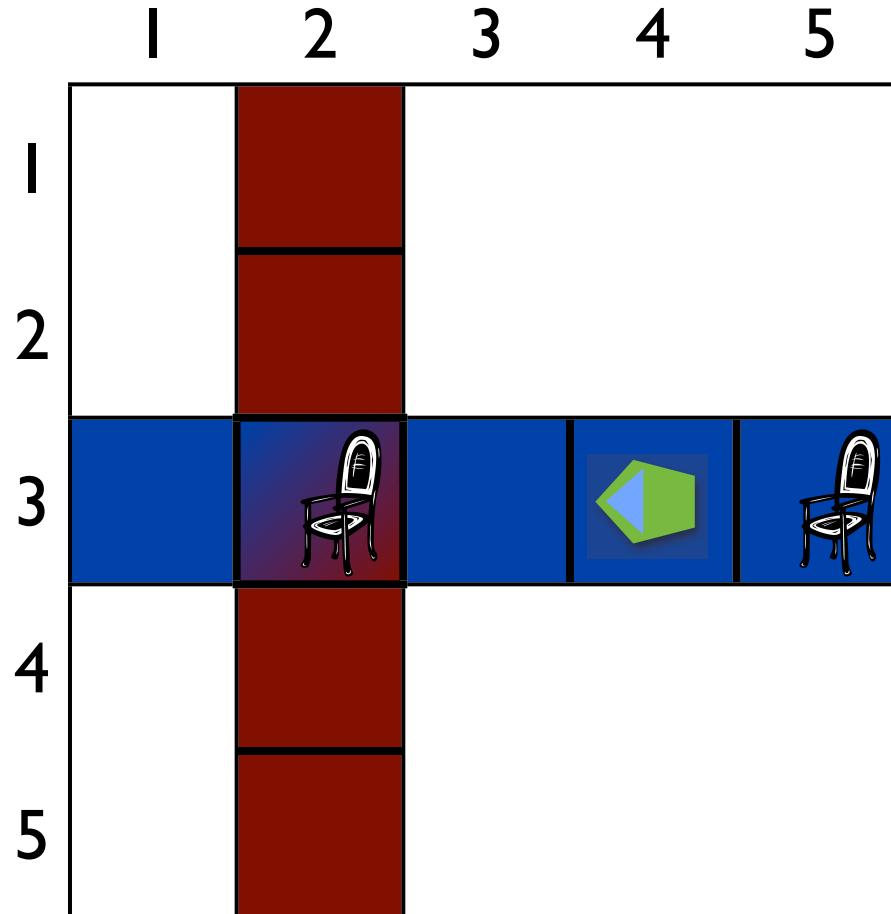
Representing Imperatives



Given a set, what do we **actually** execute?

- Need to select a single action and execute it
- Reasonable solution: select simplest/shortest

Modeling Instructions



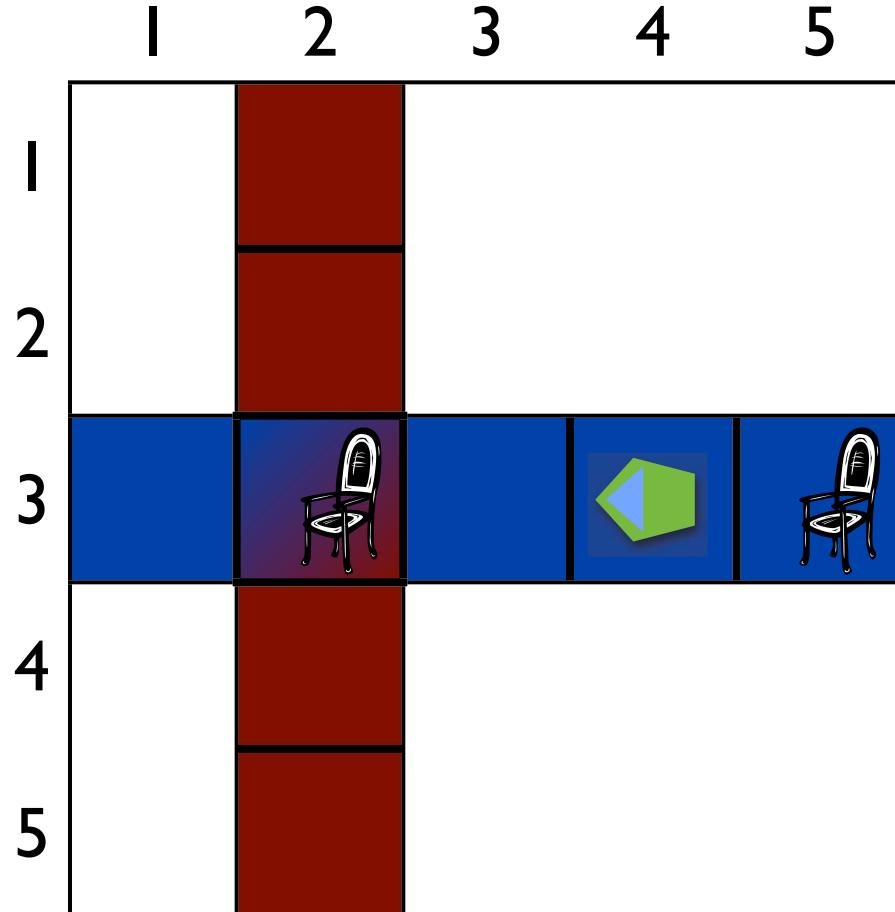
- Imperatives are sets of events
- Events are sequences of identical actions

move

$\lambda a. move(a)$

$\left\{ \begin{array}{c} \text{blue square with black triangle} \\ , \\ \text{blue square with purple square with black triangle} \\ , \\ \text{blue square with purple square with blue square with black triangle} \end{array} \right\}$

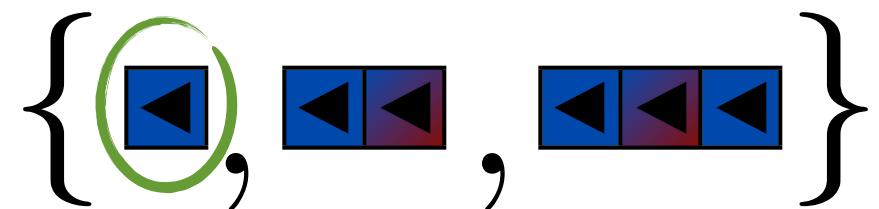
Modeling Instructions



- Imperatives are sets of events
- Events are sequences of identical actions

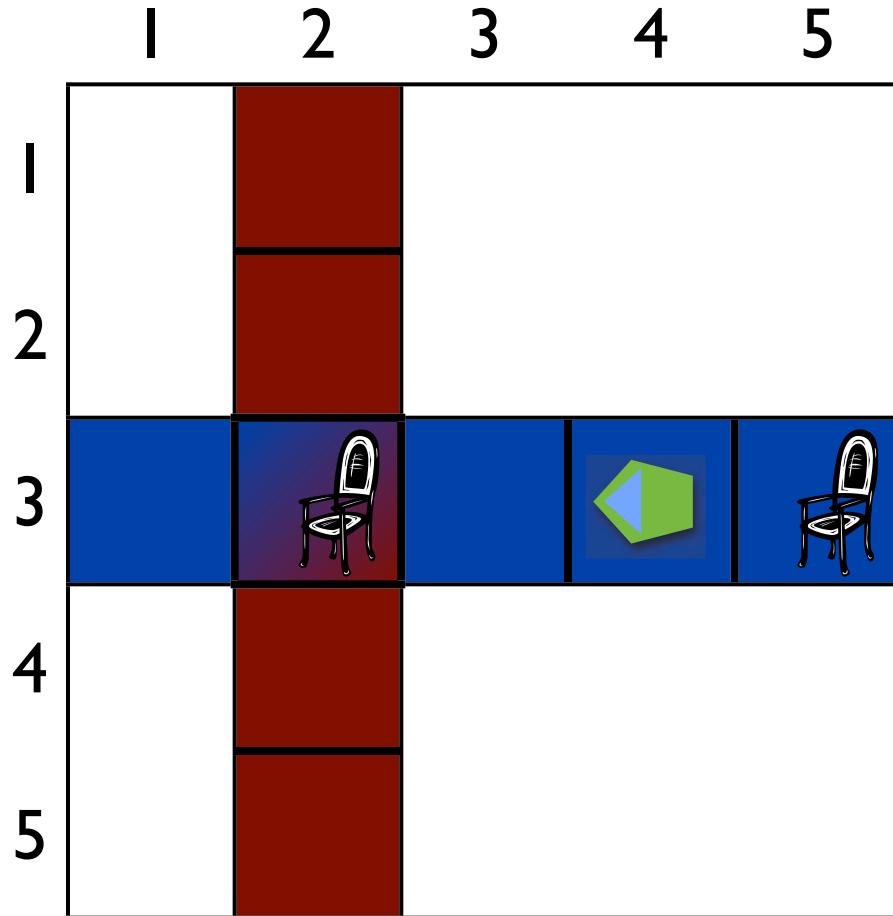
move

$\lambda a.\text{move}(a)$



Disambiguate by preferring shorter sequences

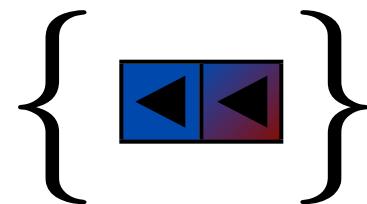
Modeling Instructions



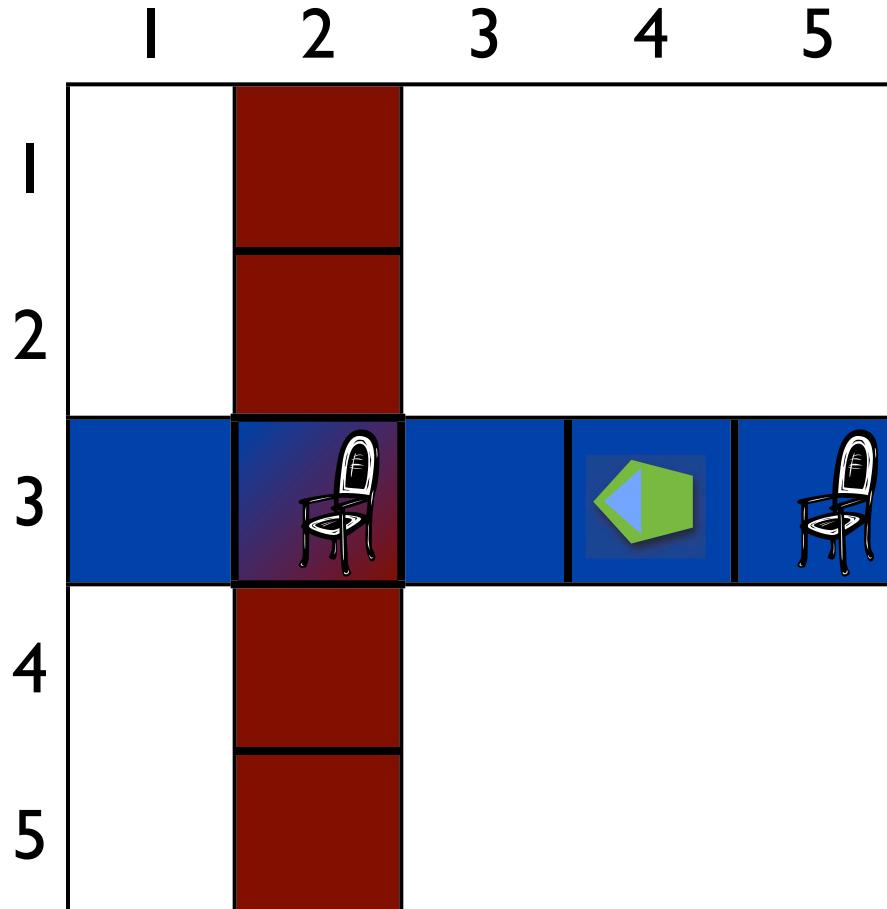
Events can be modified by adverbials

move twice

$\lambda a. move(a) \wedge len(a, 2)$



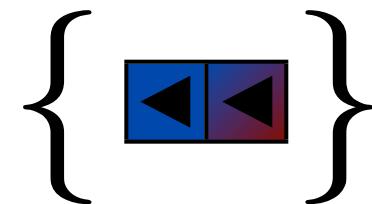
Modeling Instructions



Events can be modified by adverbials

go to the chair

$\lambda a. move(a) \wedge$
 $to(a, \iota x. chair(x))$

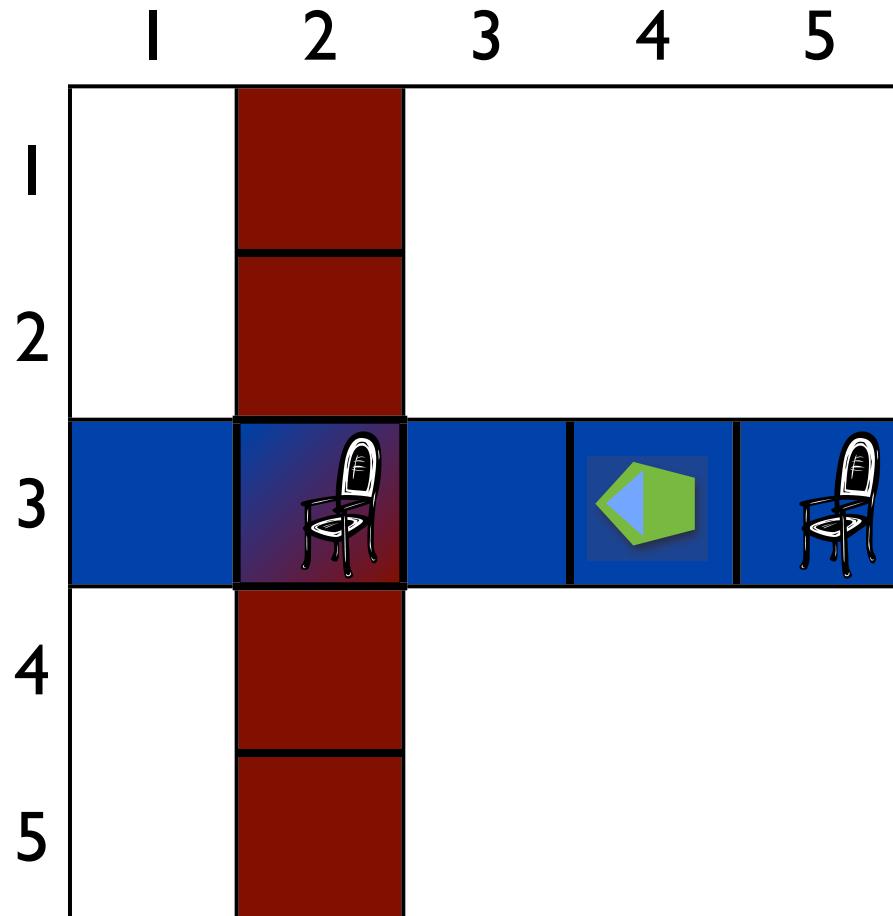


Modeling Instructions

$$\frac{\begin{array}{cccc} \text{go} & \text{to} & \text{the} & \text{chair} \\ \hline S & AP/NP & NP/N & N \\ \lambda a.move(a) & \lambda x.\lambda a.to(a, x) & \lambda f.\lambda x.f(x) & \lambda x.chair(x) \end{array}}{\begin{array}{c} \xrightarrow{NP} \\ \iota x.chair(x) \end{array}} \\ \frac{\begin{array}{c} \xleftarrow{AP} \\ \lambda a.to(a, \iota x.chair(x)) \end{array}}{\frac{\begin{array}{c} S \setminus S \\ \lambda f.\lambda a.f(a) \wedge to(a, \iota x.chair(x)) \end{array}}{\frac{\begin{array}{c} S \\ \lambda a.move(a) \wedge to(a, \iota x.chair(x)) \end{array}}{\quad}}} \quad \leftarrow$$

Treatment of events and their adverbials is similar
to nouns and prepositional phrases

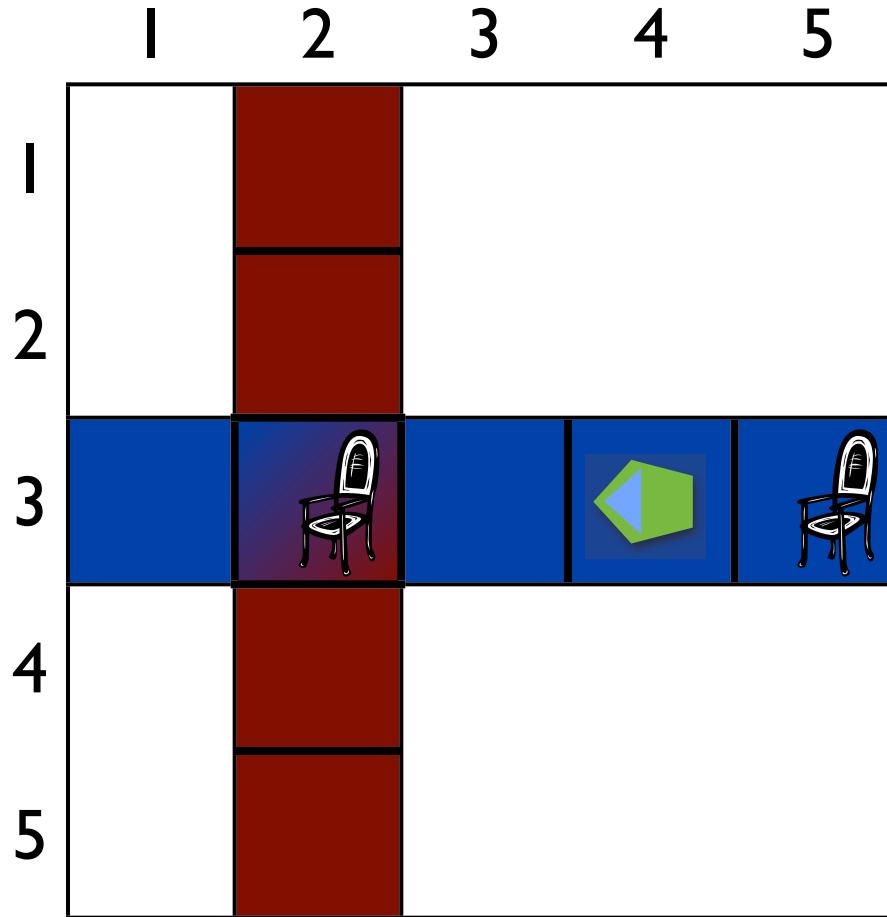
Modeling Instructions



Dynamic Models

Implicit Actions

Dynamic Models

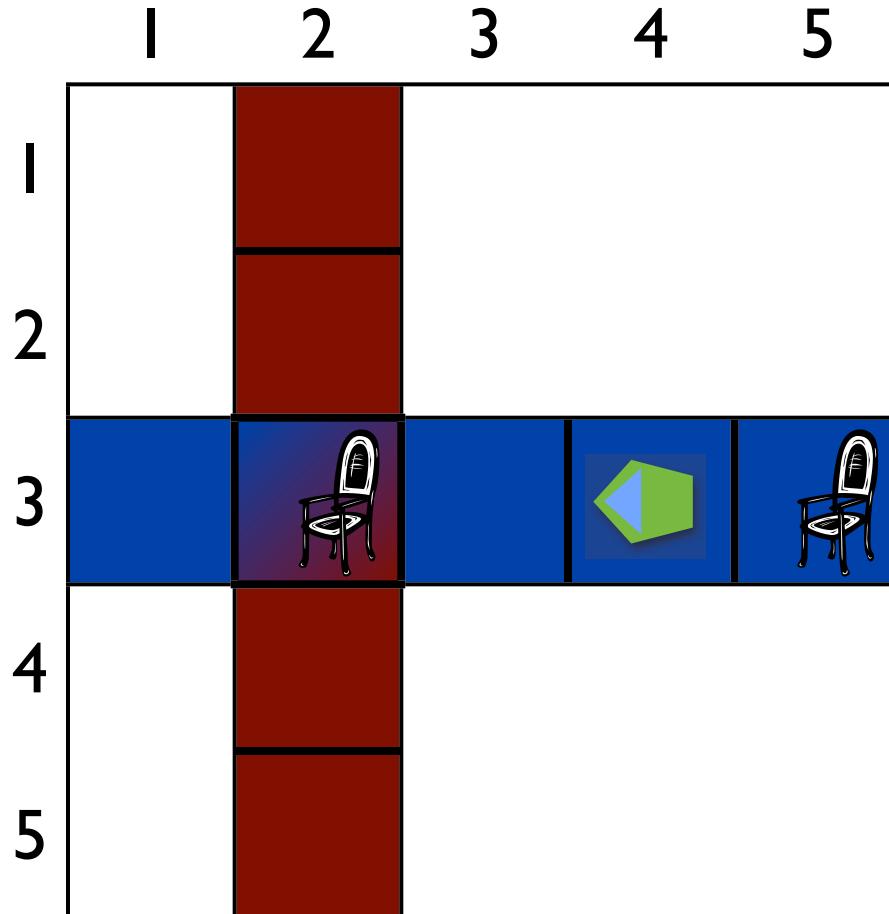


World models change
during execution

move until you reach the chair

$\lambda a. move(a) \wedge$
 $post(a, intersect(ix.chair(x), you))$

Dynamic Models

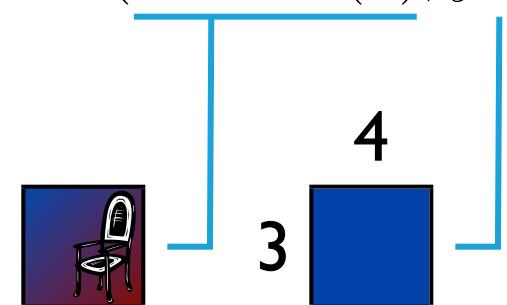


World models change
during execution

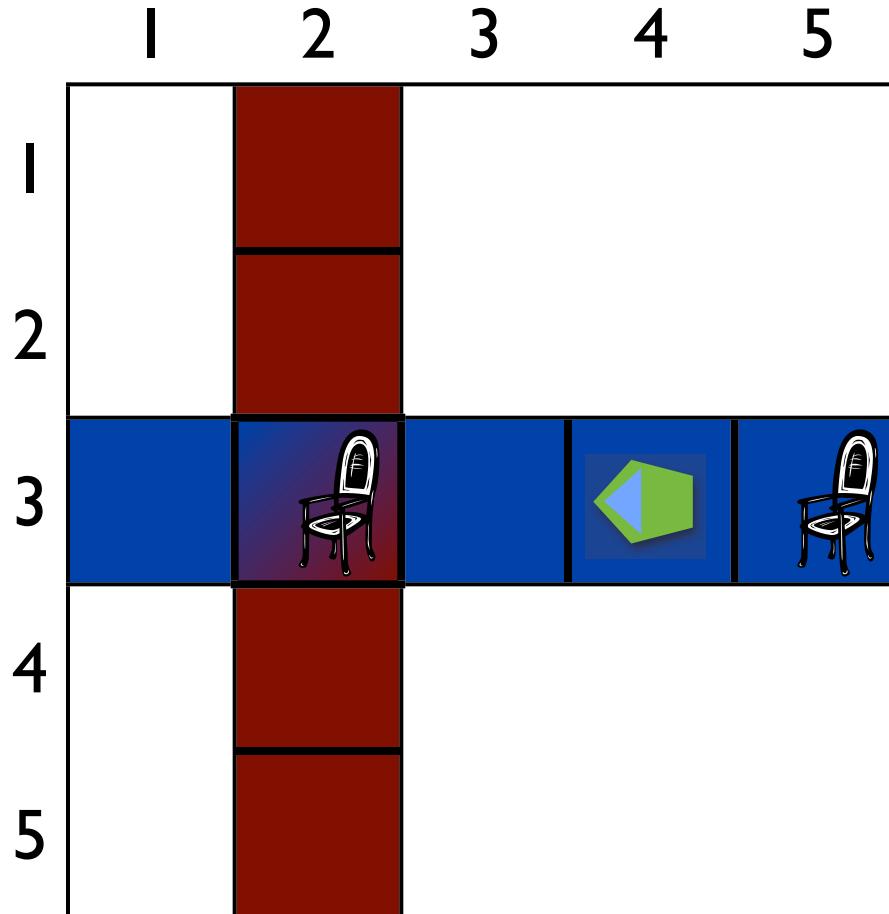
move until you reach the chair

$\lambda a. move(a) \wedge$

$post(a, intersect(ix.chair(x), you))$



Dynamic Models

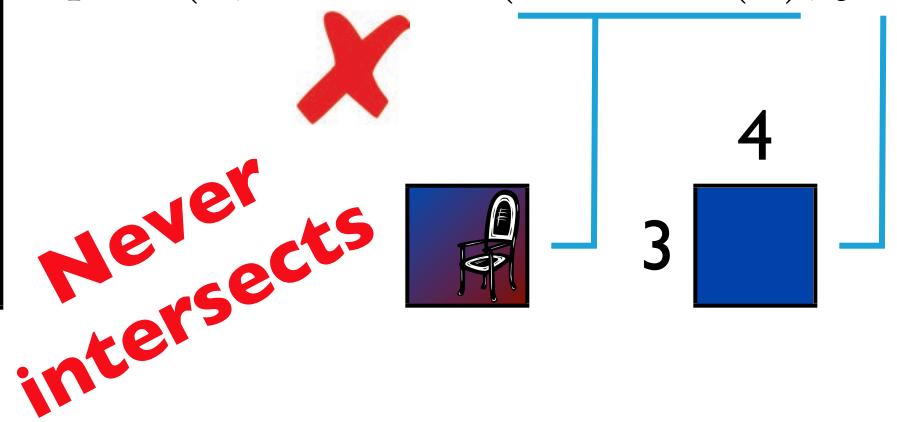


World models change
during execution

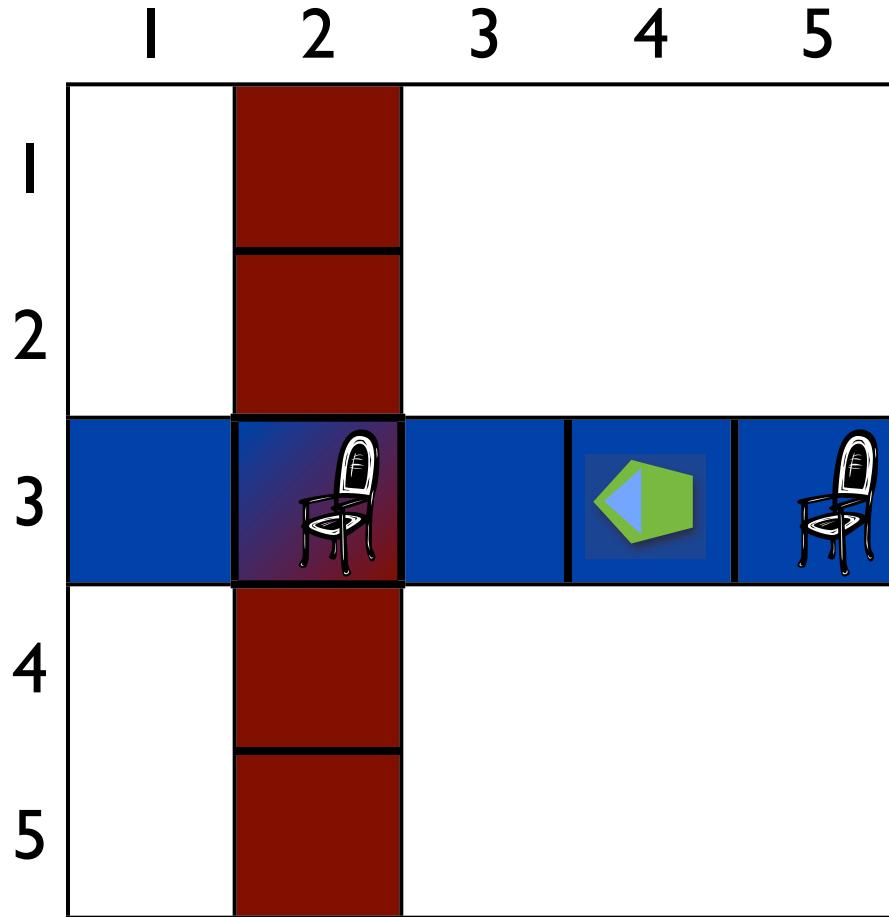
move until you reach the chair

$\lambda a. move(a) \wedge$

$post(a, intersect(ix.chair(x), you))$



Dynamic Models



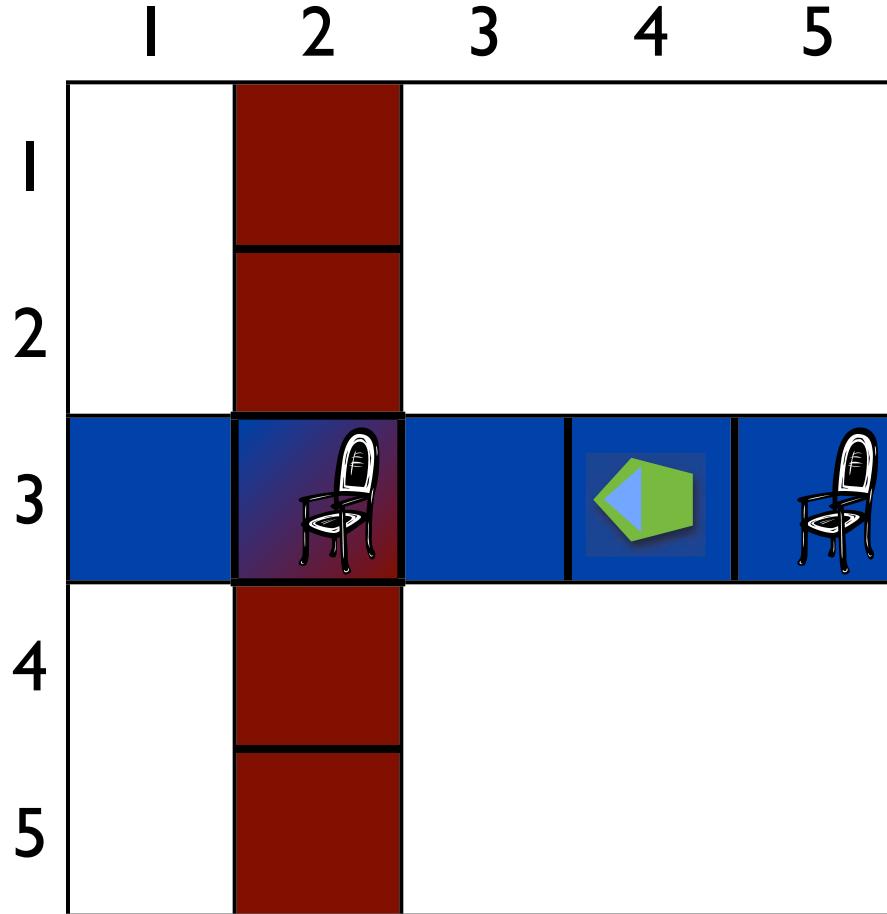
World models change
during execution

move until you reach the chair

$\lambda a. move(a) \wedge$
 $post(a, intersect(\iota x. chair(x), you))$

Update model to reflect state change

Dynamic Models

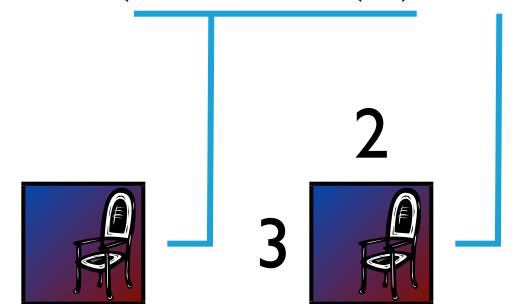


World models change
during execution

move until you reach the chair

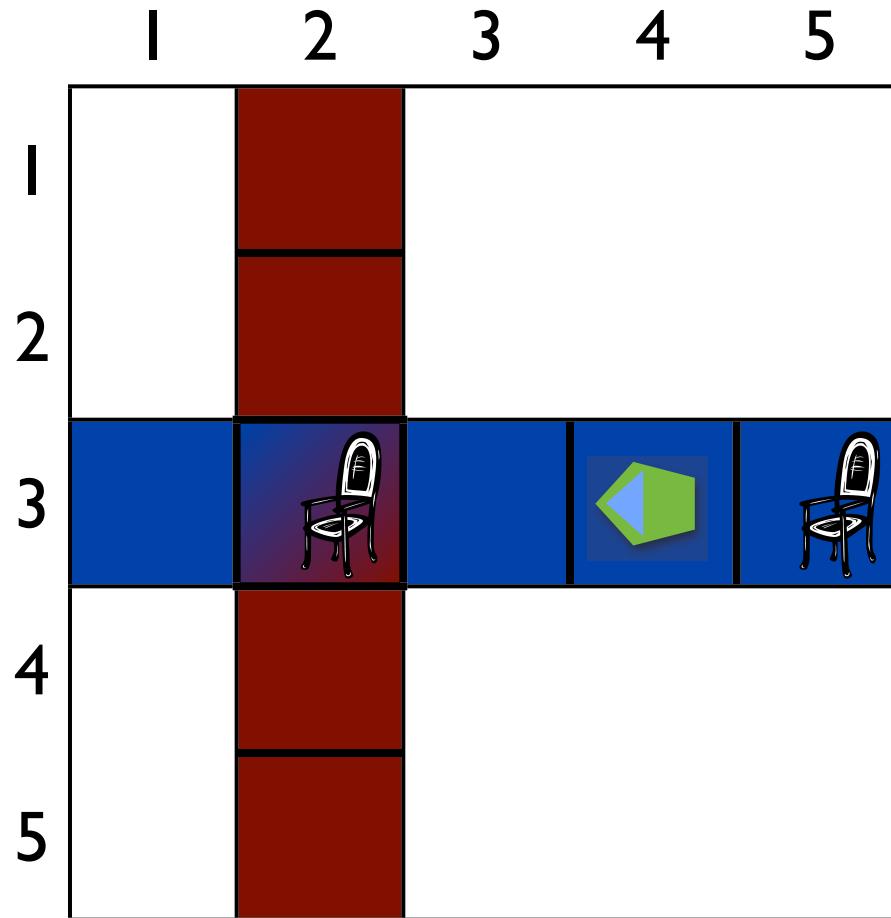
$\lambda a. move(a) \wedge$
post(a , $\text{intersect}(\text{ix.chair}(x), \text{you})$)

Update



Update model to reflect state change

Implicit Actions

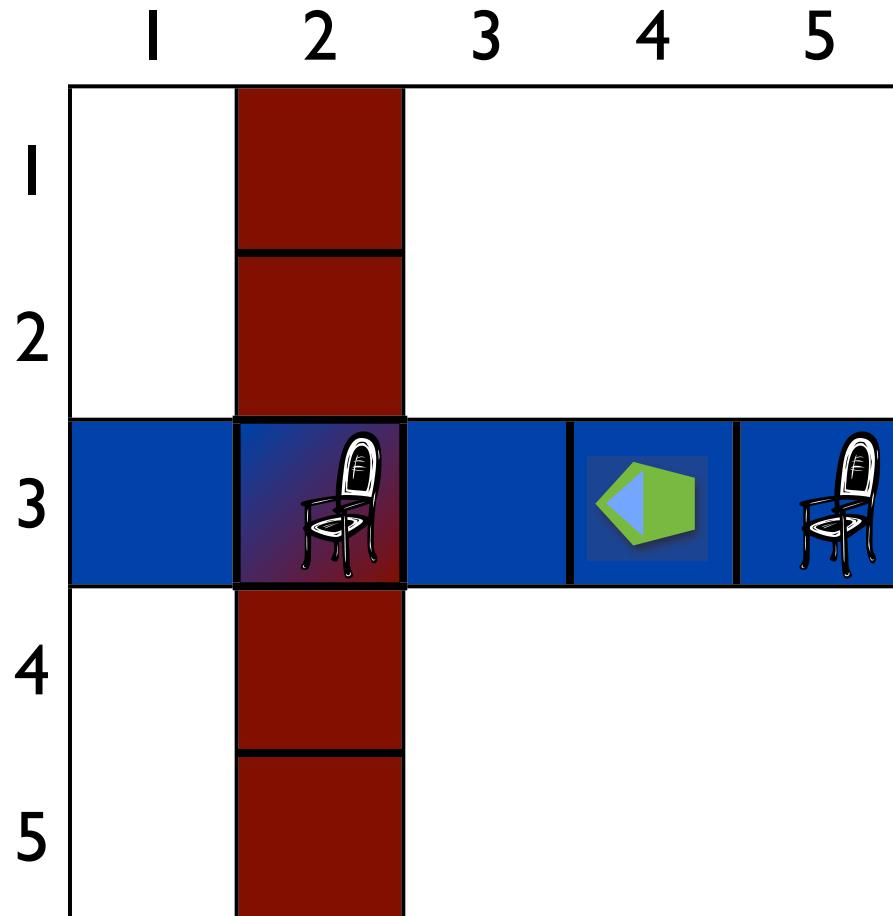


Consider action assignments
with prefixed implicit actions

at the chair; turn left

$$\lambda a. \text{turn}(a) \wedge \text{dir}(a, \text{left}) \wedge \\ \text{pre}(a, \text{intersect}(\text{ix.chair}(x), \text{you}))$$

Implicit Actions

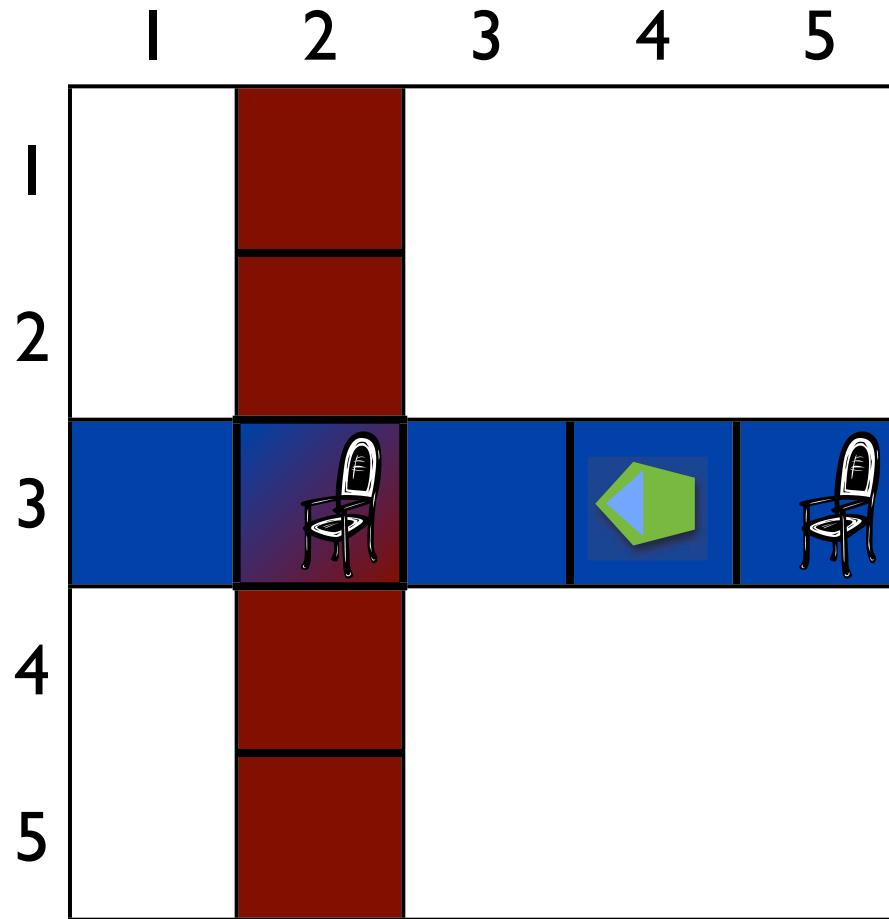


Consider action assignments
with prefixed implicit actions

at the chair; turn left

$$\lambda a. \text{turn}(a) \wedge \text{dir}(a, \text{left}) \wedge \\ \text{pre}(a, \text{intersect}(\text{ix.chair}(x), \text{you}))$$


Implicit Actions



Consider action assignments
with prefixed implicit actions

at the chair; turn left

$\lambda a. turn(a) \wedge dir(a, left) \wedge$
 $pre(a, intersect(ix.chair(x), you))$



Implicit actions

Experiments

Instruction:

at the chair, move forward three steps past the sofa

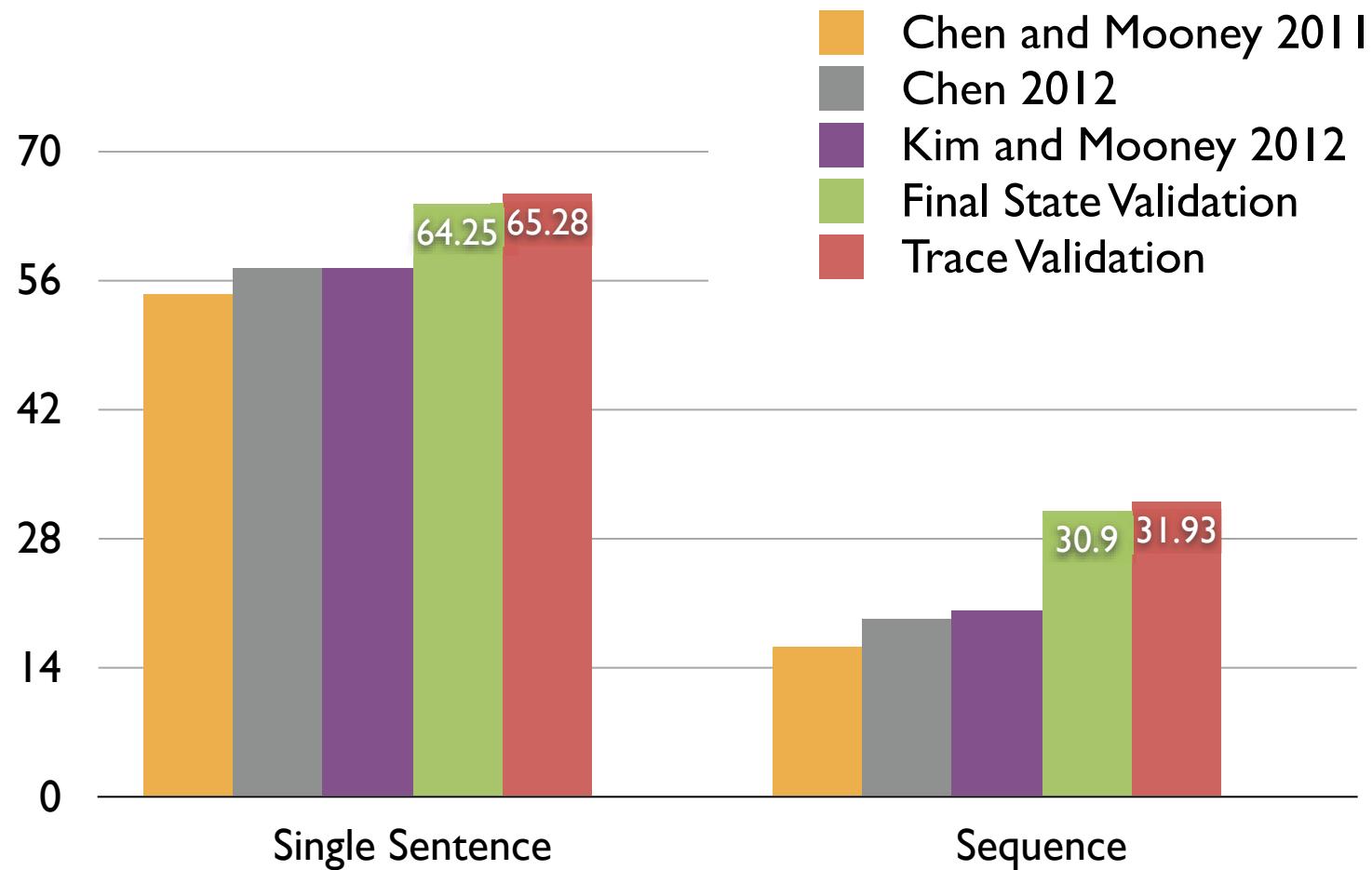
Demonstration:



- Situated learning with joint inference
- Two forms of validation
- Template-based $GENLEX(x, \mathcal{V}; \Lambda, \theta)$

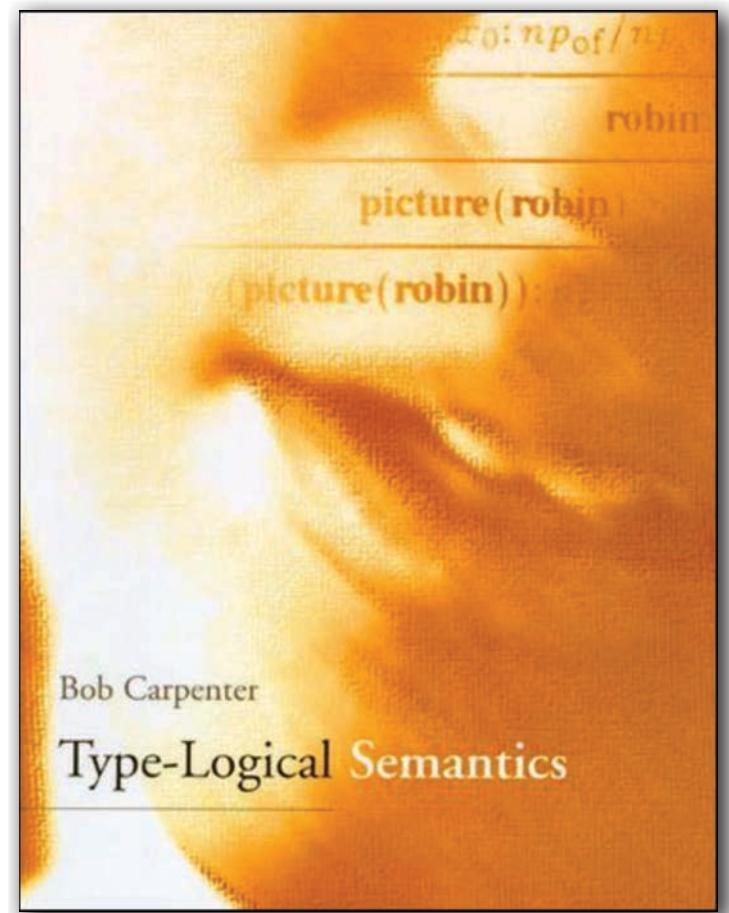
Results

SAIL Corpus - Cross Validation



More Reading about Modeling

Type-Logical Semantics
by Bob Carpenter



[Carpenter 1997]

Today

Parsing

Combinatory Categorial Grammars

Learning

Unified learning algorithm

Modeling

Best practices for semantics design

Looking Forward

Looking Forward: Scale

Goal

Answer any question posed to large, community authored databases

Challenges

- Large domains
- Scalable algorithms
- Unseen words and concepts

See

Cai and Yates 2013a, 2013b



What are the neighborhoods in New York City?

$\lambda x . \text{neighborhoods}(\text{new_york}, x)$

How many countries use the rupee?

$\text{count}(x) . \text{countries_used}(\text{rupee}, x)$

How many Peabody Award winners are there?

$\text{count}(x) . \exists y . \text{award_honor}(y) \wedge \text{award_winner}(y, x) \wedge \text{award}(y, \text{peabody_award})$

Looking Forward: Code

Goal

Program using natural language

Challenges

- Data
- Complex intent
- Complex output

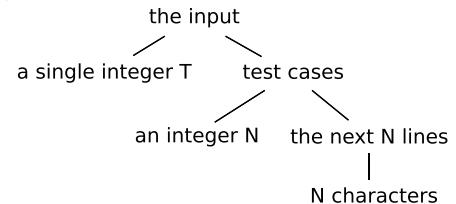
See

Kushman and Barzilay
2013; Lei et al. 2013

(a) *Text Specification:*

The input contains a single integer T that indicates the number of test cases. Then follow the T cases. Each test case begins with a line contains an integer N , representing the size of wall. The next N lines represent the original wall. Each line contains N characters. The j -th character of the i -th line figures out the color ...

(b) *Specification Tree:*



(c) *Two Program Input Examples:*

1
10
YYWYYWWWW
YWWWWYWWWW
YYWYYWWWW
...
WWWWWWWWWW

2
1
Y
5
YWYWW
...
WWYYY

Text Description	Regular Expression
three letter word starting with 'X'	\bX [A-Za-z] {2} \b

Looking Forward: Context

Goal

Understanding how sentence meaning varies with context

Challenges

- Data
- Linguistics: co-ref, ellipsis, etc.

See

Miller et al. 1996;
Zettlemoyer and Collins 2009; Artzi and Zettlemoyer 2013

Example #1:

- (a) show me the flights from boston to philly
 $\lambda x.\text{flight}(x) \wedge \text{from}(x, \text{bos}) \wedge \text{to}(x, \text{phi})$
- (b) show me the ones that leave in the morning
 $\lambda x.\text{flight}(x) \wedge \text{from}(x, \text{bos}) \wedge \text{to}(x, \text{phi}) \wedge \text{during}(x, \text{morning})$
- (c) what kind of plane is used on these flights
 $\lambda y.\exists x.\text{flight}(x) \wedge \text{from}(x, \text{bos}) \wedge \text{to}(x, \text{phi}) \wedge \text{during}(x, \text{morning}) \wedge \text{aircraft}(x) = y$

Example #2:

- (a) show me flights from milwaukee to orlando
 $\lambda x.\text{flight}(x) \wedge \text{from}(x, \text{mil}) \wedge \text{to}(x, \text{orl})$
- (b) cheapest
 $\text{argmin}(\lambda x.\text{flight}(x) \wedge \text{from}(x, \text{mil}) \wedge \text{to}(x, \text{orl}), \lambda y.\text{fare}(y))$
- (c) departing wednesday after 5 o'clock
 $\text{argmin}(\lambda x.\text{flight}(x) \wedge \text{from}(x, \text{mil}) \wedge \text{to}(x, \text{orl}) \wedge \text{day}(x, \text{wed}) \wedge \text{depart}(x) > 1700, \lambda y.\text{fare}(y))$

Looking Forward: Sensors

Goal

Integrate semantic parsing
with rich sensing on real
robots



Challenges

- Data
- Managing uncertainty
- Interactive learning

See

Matuszek et al. 2012; Tellex
et al. 2013; Krishnamurthy
and Kollar 2013

Move the pallet from the truck.
Remove the pallet from the back of the truck.
Offload the metal crate from the truck.



UW SPF

Open source semantic parsing framework

<http://yoavartzi.com/spf>

Semantic
Parser

Flexible High-Order
Logic Representation

Learning
Algorithms

Includes ready-to-run examples

[fin]

Supplementary Material

Function Composition

$$g_{\langle \alpha, \beta \rangle} = \lambda x. G$$

$$f_{\langle \beta, \gamma \rangle} = \lambda y. F$$

$$g(A) = (\lambda x. G)(A) = G[x := A]$$

$$f(g(A)) = (\lambda y. F)(G[x := A]) =$$

$$F[y := G[x := A]]$$

$$\lambda x. f(g(A))[A := x] =$$

$$\lambda x. F[y := G[x := A]][A := x] =$$

$$\lambda x. F[y := G] = (f \cdot g)_{\langle \alpha, \gamma \rangle}$$

References

- Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Artzi, Y. and Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.
- Artzi, Y. and Zettlemoyer, L. (2013b). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.
- Branavan, S., Chen, H., Zettlemoyer, L., and Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.
- Branavan, S., Zettlemoyer, L., and Barzilay, R. (2010). Reading between the lines: learning to map high-level instructions to commands. In *Proceedings of the Conference of the Association for Computational Linguistics*.
- Cai, Q. and Yates, A. (2013a). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Cai, Q. and Yates, A. (2013b). Semantic parsing freebase: Towards open-domain semantic parsing. In *Joint Conference on Lexical and Computational Semantics: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.
- Carpenter, B. (1997). *Type-Logical Semantics*. The MIT Press.
- Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.
- Church, A. (1932). A set of postulates for the foundation of logic. *The Annals of Mathematics*, 33:346–366.
- Church, A. (1940). A formulation of the simple theory of types. *The journal of symbolic logic*, 5:56–68.
- Clark, S. and Curran, J. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.
- Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world’s response. In *Proceedings of the Conference on Computational Natural Language Learning*.

- Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*.
- Davidson, D. (1967). The logical form of action sentences. *Essays on actions and events*, pages 105–148.
- Davidson, D. (1969). The individuation of events. In *Essays in honor of Carl G. Hempel*, pages 216–234. Springer.
- Granroth-Wilding, M. and Steedman, M. (2012). *Statistical parsing for harmonic analysis of jazz chord sequences*. Ann Arbor, MI: MPublishing, University of Michigan Library.
- Joshi, A. K., Shanker, K. V., and Weir, D. (1990). The convergence of mildly context-sensitive grammar formalisms. Technical report.
- Kim, J. and Mooney, R. J. (2012). Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(1):193–206.
- Kushman, N. and Barzilay, R. (2013). Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.
- Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Lei, T., Long, F., Barzilay, R., and Rinard, M. (2013). From natural language specifications to program input parsers. In *Proceedings of the the annual meeting of the Association for Computational Linguistics*.

- Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of the Conference of the Association of Computational Linguistics*.
- Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.
- Maienborn, C., Von Heusinger, K., and Portner, P. (2011). *Semantics: An international handbook of natural language and meaning*. Walter de Gruyter.
- Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012a). A joint model of language and perception for grounded attribute learning. *Proceedings of the International Conference on Machine Learning*.
- Matuszek, C., Herbst, E., Zettlemoyer, L. S., and Fox, D. (2012b). Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.
- Miller, S., Bobrow, R., Ingria, R., and Schwartz, R. (1994). Hidden understanding models of natural language. In *Proceedings of the Conference of the Association of Computational Linguistics*.
- Parsons, T. (1990). *Events in the Semantics of English*. The MIT Press.
- Scontras, G., Graff, P., and Goodman, N. D. (2012). Comparing pluralities. *Cognition*, 123(1):190–197.
- Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.
- Steedman, M. (2000). *The Syntactic Process*. The MIT Press.
- Steedman, M. (2011). *Taking Scope*. The MIT Press.
- Tang, L. R. and Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the European Conference on Machine Learning*.
- Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.
- Tellex, S., Thaker, P., Joseph, J., and Roy, N. (2013). Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, pages 1–17.
- Wong, Y. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

- Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.
- Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.
- Zettlemoyer, L. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.