

# Lecture 6: Learning from Denotations and Entailments

Kyle Richardson

[kyle@ims.uni-stuttgart.de](mailto:kyle@ims.uni-stuttgart.de)

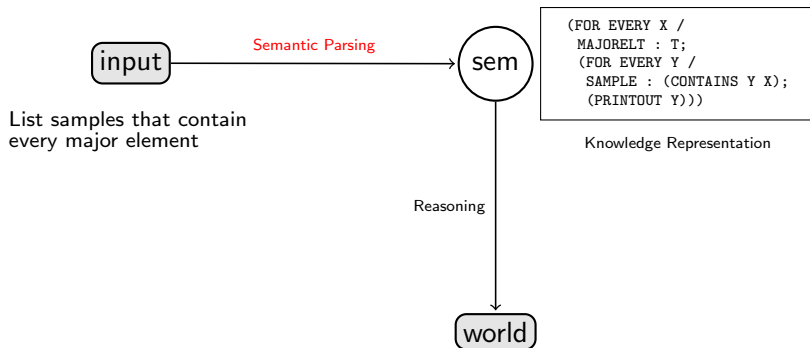
July 7, 2016

# Lecture Plan

- ▶ **Overview:** Review of class topics and outstanding issues.
- ▶ **General topics:** Knowledge Representation, Learning from Entailment

# The Big Picture (reminder)

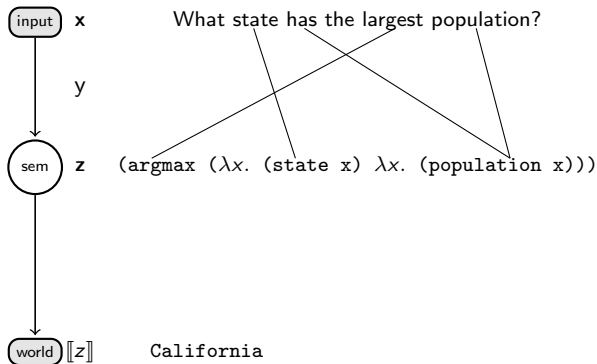
## ► Standard processing pipeline



$$\llbracket sem \rrbracket = \{S10019, S10059, \dots\}$$

# Data-driven Semantic Parsing

- **Goal:** Given data, **learn** a function that can map any given input ( $x$ ) to a meaning representation ( $z$ ).
- What kind of data do we learn from?



**Supervision: Dataset  $D$**

**Logical Forms:**  $D = \{(x_i, z_i)\}_{i=1}^N$   
*Task:* learn (latent)  $y$ , **translation**

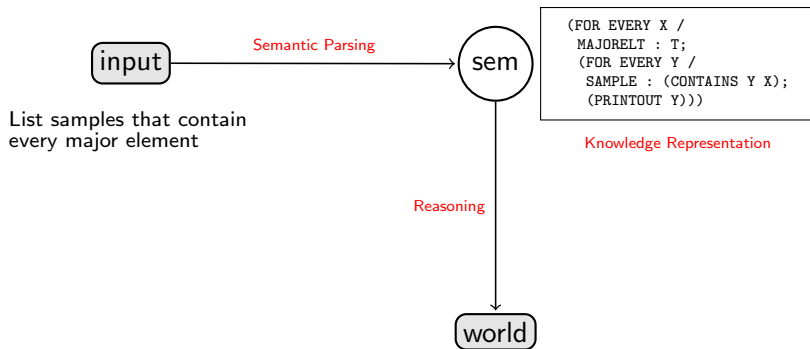
Zettlemoyer and Collins (2009)  
Kwiatkowski et al. (2010)

**Denotations:**  $D = \{(x_j, \llbracket z_j \rrbracket)\}_{i=1}^N$   
*Task:* learn  $z, y$ , **program synthesis**

Liang et al. (2013)  
Berant et al. (2013)

# Question Today

- How do these different subproblems interact?



$$\llbracket sem \rrbracket = \{S10019, S10059, \dots\}$$

# Learning from Logical Forms: CCG Example

- ▶ **Data:** (Oklahoma borders Texas, borders'(oklahoma',texas'))
- ▶ **Latent Variable:** CCG derivations, Probability distribution over derivations.

$$\begin{array}{c}
 \begin{array}{ccc}
 & \text{borders} & \text{Texas} \\
 & \hline
 \text{Oklahoma} & (S \backslash NP) / NP : \lambda y, \lambda x. borders(x, y) & NP : texas' \\
 \hline
 NP : oklahoma' & S \backslash NP : \lambda x. borders(x, texas') & \\
 \hline
 S : borders(oklahoma',texas') & & \checkmark
 \end{array}
 \end{array}
 \begin{array}{c}
 ( > ) \\
 ( < )
 \end{array}$$

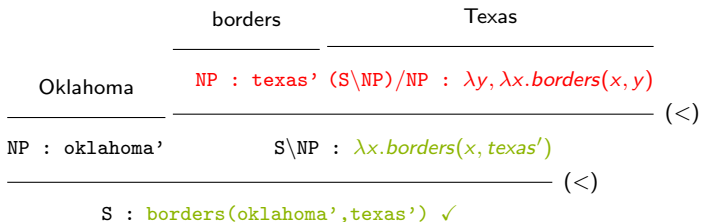
# Learning from Logical Forms: CCG Example

- ▶ **Data:** (Oklahoma borders Texas, borders'(oklahoma',texas'))
- ▶ **Latent Variable:** CCG derivations, Probability distribution over derivations.

$$\begin{array}{c} \text{borders} \qquad \qquad \qquad \text{Texas} \\ \hline \text{Oklahoma} \quad (S \backslash NP) / NP : \lambda y, \lambda x. borders(x, y) \quad NP : \text{texas}' \\ \hline \text{NP : ohio}' \qquad \qquad S \backslash NP : \lambda x. borders(x, \text{texas}') \\ \hline S : borders(\text{ohio}', \text{texas}') \quad \times \end{array} \quad \begin{array}{c} (>) \\ \\ (<) \end{array}$$

# Learning from Logical Forms: CCG Example

- ▶ **Data:** (Oklahoma borders Texas, borders'(oklahoma',texas'))
- ▶ **Latent Variable:** CCG derivations, Probability distribution over derivations.





# Learning from Logical Forms: Compositional Model

|   |  |               |     |
|---|--|---------------|-----|
|   | <u>borders</u>   | <u>Texas</u>  |     |
| Oklahoma                                    | $(S \backslash NP) / NP : \lambda y, \lambda x. borders(x, y)$ | $NP : texas'$ |     |
|   |  |               | (>) |
| $NP : oklahoma'$                            | $S \backslash NP : \lambda x. borders(x, texas')$              |               |     |
|   |  |               | (<) |
| $S : borders(oklahoma', texas') \checkmark$ |  |               |     |

```
Prelude > let borders :: ([Char], [Char]) -> Bool;
Prelude | borders a = (elem a [("oklahoma", "texas"), ...])
Prelude > borders ("nh", "texas")
=> False
```

# Learning from Logical Forms: Compositional Model

|                  | borders  | Texas         |
|------------------|--|---------------|
| Oklahoma         | $(S \backslash NP) / NP : \lambda y, \lambda x. borders(x, y)$ | $NP : texas'$ |
|                  | <hr/>  |               |
| $NP : oklahoma'$ | $S \backslash NP : \lambda x. borders(x, texas')$              |               |
|                  | <hr/>  |               |
|                  | $S : borders(oklahoma', texas') \checkmark$                    |               |

*Prelude* > :type borders

=> borders :: ([Char], [Char]) -> Bool

# Learning from Logical Forms: Compositional Model

|   |  |               |     |
|---|--|---------------|-----|
|   | borders  | Texas         |     |
| Oklahoma                                    | $(S \backslash NP) / NP : \lambda y, \lambda x. borders(x, y)$ | $NP : texas'$ |     |
|   |  |               | (>) |
| $NP : oklahoma'$                            | $S \backslash NP : \lambda x. borders(x, texas')$              |               |     |
|   |  |               | (<) |
| $S : borders(oklahoma', texas') \checkmark$ |  |               |     |

```
Prelude > :type borders
```

```
=> borders :: ([Char], [Char]) -> Bool
```

```
Prelude > let borders_c = curry borders
```

```
Prelude > :type borders_c
```

```
=> borders_c :: [Char] -> [Char] -> Bool
```

**semantic type:**  $e \longrightarrow e \longrightarrow t$

# Lexical rule templates (Triggers)

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

- **Templates:** Extracting CCG entries from example logical forms, does not work without target logical forms.
- Having logical forms keeps the space of rules/programs feasible.

# Lexical rule templates (Triggers)

- ▶ **Templates:** Extracting CCG entries from example logical forms, does not work without target logical forms.
- ▶ Having logical forms keeps the space of rules/programs feasible.

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



# Lexical rule templates (Triggers)

- ▶ **Templates:** Extracting CCG entries from example logical forms, does not work without target logical forms.
- ▶ Having logical forms keeps the space of rules/programs feasible.

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



NP : texas

# Lexical rule templates (Triggers)

- ▶ **Templates:** Extracting CCG entries from example logical forms, does not work without target logical forms.
- ▶ Having logical forms keeps the space of rules/programs feasible.

$$\begin{array}{ccc} & & \lambda x.state(x) \wedge borders(x, texas) \\ & & \downarrow \\ \begin{array}{l} NP \\ (S \backslash NP) / NP \end{array} & : & \begin{array}{l} texas \\ \lambda y \lambda x.borders(x, y) \end{array} \end{array}$$

# Lexical rule templates (Triggers)

- ▶ **Templates:** Extracting CCG entries from example logical forms, does not work without target logical forms.
- ▶ Having logical forms keeps the space of rules/programs feasible.

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



|                          |   |                                     |
|--------------------------|---|-------------------------------------|
| NP                       | : | texas                               |
| $(S \backslash NP) / NP$ | : | $\lambda y \lambda x.borders(x, y)$ |
| $S \backslash N$         | : | $\lambda x.state(x)$                |



# Assumptions for CCG approach

- **Logical Form:** for each input, e.g.,

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

- **Implementation:** Programs that implement domain model.
- **Seed Lexicon:** Initial set of CCG lexical entries.

|        |    |                               |   |  |
|--------|----|-------------------------------|---|--|
| Texas  | := | NP                            | : | texas  |
| border | := | $(S \backslash NP) / NP$      | : | $\lambda y \lambda x.borders(x, y)$                |
| states | := | $S \backslash N$              | : | $\lambda x.state(x)$                               |
| which  | := | $(S / (S \backslash NP)) / N$ | : | $\lambda f, \lambda g, \lambda x.f(x) \wedge g(x)$ |
| Texas  | := | $(S / (S \backslash NP)) / N$ | : | $\lambda f, \lambda g, \lambda x.f(x) \wedge g(x)$ |
| border | := | $S \backslash N$              | : | $\lambda x.state(x)$                               |
| ...    |    |                               |   |  |

# Learning from Logical Forms: General Properties

- ▶ **Goal:** Learn to translate to logical forms using example sentences with target logical representations.
- ▶ **Critical:** Having example logical forms limits the space of mappings and translation rules.
- ▶ The types of models often used are indifferent to the types of representations used.

# Learning from Denotations

- ▶ Alternative approach to learning, only needs example input/output (requires a background database of facts).

# Learning from Denotations

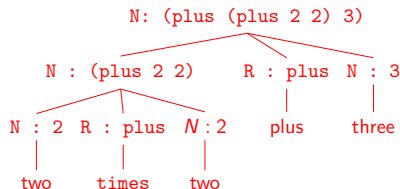
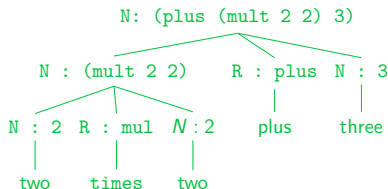
- ▶ Alternative approach to learning, only needs example input/output (requires a background database of facts).
  - ▶ **Logical forms:** (*two times two plus three*, `(plus (mult 2 2) 3)`)
  - ▶ **Denotations:** (*two times two plus three*, `7`)

# Learning from Denotations

- ▶ Alternative approach to learning, only needs example input/output (requires a background database of facts).
  - ▶ **Logical forms:** (*two times two plus three*, `(plus (mult 2 2) 3)`)
  - ▶ **Denotations:** (*two times two plus three*, `7`)
- ▶ **Difference:** In the second case, the logical representation is a latent variables (not only the translation), no program implementation.

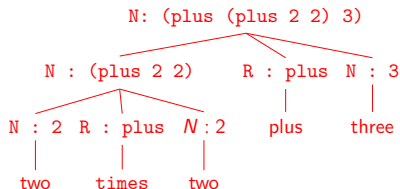
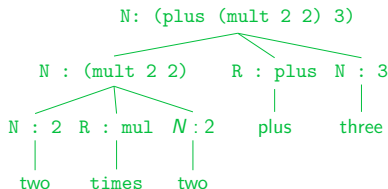
# Learning from Denotations

- ▶ Alternative approach to learning, only needs example input/output (requires a background database of facts).
  - ▶ **Logical forms:** (*two times two plus three*, (plus (mult 2 2) 3))
  - ▶ **Denotations:** (*two times two plus three*, 7)
- ▶ **Difference:** In the second case, the logical representation is a latent variables (not only the translation), no program implementation.



# Learning from Denotations

- ▶ Alternative approach to learning, only needs example input/output (requires a background database of facts).
  - ▶ **Logical forms:** (*two times two plus three*, (plus (mult 2 2) 3))
  - ▶ **Denotations:** (*two times two plus three*, 7)
- ▶ **Why:** Avoids annotation (practical/methodological), can we learn programs from input/output? (scientific)



# Learning from Denotations (Liang et al. (2011))

- **Computational Problem:** Requires exploring an exponential space of possible representations and programs:<sup>1</sup>

**Input:** What is the most populous city in California?



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



**Answer:** Los Angeles

---

<sup>1</sup>Examples throughout adapted from Percy Liang's slides



# Learning from Denotations (Liang et al. (2011))

- **Computational Problem:** Requires exploring an exponential space of possible representations and programs:

**Input:** What is the most populous city in California?



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



**Answer:** Los Angeles

# Learning from Denotations (Liang et al. (2011))

- **Computational Problem:** Requires exploring an exponential space of possible representations and programs:

**Input:** What is the most populous city in California?



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



**Answer:** Los Angeles

# Learning from Denotations (Liang et al. (2011))

- **Computational Problem:** Requires exploring an exponential space of possible representations and programs:

**Input:** What is the most populous city in California?



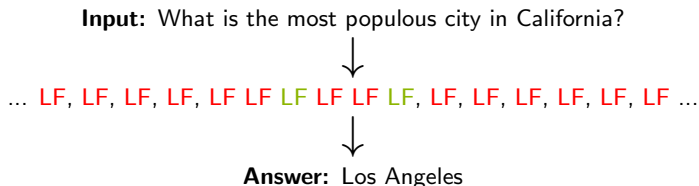
$\text{argmax}(\lambda x. \text{city}(x) \wedge \text{loc}(x, \text{CA}), \lambda x. \text{population}(x))$



**Answer:** Los Angeles

# Learning from Denotations (Liang et al. (2011))

- **Computational Problem:** Requires exploring an exponential space of possible representations and programs:



# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **World:** or domain of discourse is a database consisting of predicates.

# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **World:** or domain of discourse is a database consisting of predicates.

**Input:** A city located in California.



?



**Answer:** San Francisco

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **World:** or domain of discourse is a database consisting of predicates.

**Input:** A city located in California.

$\downarrow$   
 $\lambda x. \text{city}(x)$

$\downarrow$   
**Answer:** San Francisco

| city          | loc           |            |
|---------------|---------------|------------|
| San Francisco | Manhattan     | New York   |
| Chicago       | San Francisco | California |
| New York      | Chicago       | Illinois   |
| ....          | ...           | ...        |

# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **Simple case:** only unary predicates. What is the search space?

**Input:** A city located in California.

↓  
 $\lambda x.\text{city}(x)$

↓  
**Answer:** San Francisco

| city          | loc           |            |
|---------------|---------------|------------|
| San Francisco | Manhattan     | New York   |
| Chicago       | San Francisco | California |
| New York      | Chicago       | Illinois   |
| ....          | ...           | ...        |



# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **Disjunction:** In this case, imposes constraint on equality.

**Input:** A city located in California.



$\lambda x. \text{city}(x) \wedge \text{loc}(x, \text{California})$



**Answer:** San Francisco

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **Next stage:** **Unary+Binary.** What is the search space?.

**Input:** A city located in California.



$\lambda x. \text{city}(x) \wedge \text{loc}(x, \text{California})$



**Answer:** San Francisco

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

# Why exponential?

- ▶ **Geoquery:** Answering questions about American geography.
- ▶ **Unrestrained:** What is the search space?.

**Input:** A city located in California.



$\lambda x. \text{city}(x) \wedge \text{loc}(x, \text{California}) \wedge P_1(x, Y) \wedge P_2(x, Y) \wedge \dots \wedge \dots$



**Answer:** San Francisco

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

# Learning from Denotations and Knowledge Representation

- ▶ The search space for unrestricted lambda calculus logical forms is too large to search.

# Learning from Denotations and Knowledge Representation

- ▶ The search space for unrestricted lambda calculus logical forms is too large to search.
  - ▶ Consider set of predicates  $P = \{\text{loc}, \text{city}, \text{ borders}, \dots\}$  of size  $n$ , the set of disjunctions is  $2^n$  (equal to the powerset of  $P$ )

$$2^{50} = 1,125,899,906,842,624$$

# Learning from Denotations and Knowledge Representation

- ▶ The search space for unrestricted lambda calculus logical forms is too large to search.
  - ▶ Consider set of predicates  $P = \{\text{loc}, \text{city}, \text{ borders}, \dots\}$  of size  $n$ , the set of disjunctions is  $2^n$  (equal to the powerset of  $P$ )

$$2^{50} = 1,125,899,906,842,624$$

- ▶ We cannot rely on example logical forms to constrain the space.

# Learning from Denotations and Knowledge Representation

- ▶ The search space for unrestricted lambda calculus logical forms is too large to search.
  - ▶ Consider set of predicates  $P = \{\text{loc}, \text{city}, \text{ borders}, \dots\}$  of size  $n$ , the set of disjunctions is  $2^n$  (equal to the powerset of  $P$ )

$$2^{50} = 1,125,899,906,842,624$$

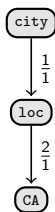
- ▶ We cannot rely on example logical forms to constrain the space.
- ▶ **Solution** (Liang et al. (2011)): Develop a constrained version of lambda calculus, simplifies representations, tree structured

# DCS Language (Liang et al. (2011))

- ▶ Tree structured, nodes are predicates and edges are relations.
- ▶ **Basic version:** Uses the join relations, which specify constraints.

**Input:** A city in California

**Representation**



**Constraints**

**World**

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

CA

|            |
|------------|
| California |
|------------|

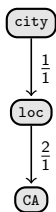


# DCS Language (Liang et al. (2011))

- ▶ Tree structured, nodes are predicates and edges are relations.
- ▶ **Basic version:** Uses the join relations, which specify constraints.

**Input:** A city in California

**Representation**



**Constraints**

$c \in \text{city}$   
 $l \in \text{loc}$   
 $s \in \text{CA}$

**World**

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

CA

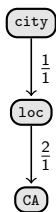
|            |
|------------|
| California |
|------------|

# DCS Language (Liang et al. (2011))

- ▶ Tree structured, nodes are predicates and edges are relations.
- ▶ **Basic version:** Uses the join relations, which specify constraints.

**Input:** A city in California

**Representation**



**Constraints**

$c \in \text{city}$

$l \in \text{loc}$

$s \in \text{CA}$

$c_1 = l_1$

$l_2 = s_1$

**World**

city

|               |
|---------------|
| San Francisco |
| Chicago       |
| New York      |
| ....          |

loc

|               |            |
|---------------|------------|
| Manhattan     | New York   |
| San Francisco | California |
| Chicago       | Illinois   |
| ...           | ...        |

CA

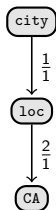
|            |
|------------|
| California |
|------------|

# DCS Language (Liang et al. (2011))

- ▶ Tree structured, nodes are predicates and edges are relations.
- ▶ **Basic version:** Uses the join relations, which specify constraints.

**Input:** A city in California

**Representation**



**Constraints**

$c \in \text{city}$

$l \in \text{loc}$

$s \in \text{CA}$

$c_1 = l_1$

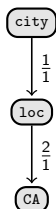
$l_2 = s_1$

# DCS Language (Liang et al. (2011))

- ▶ Tree structured, nodes are predicates and edges are relations.
- ▶ **Basic version:** Uses the join relations, which specify constraints.

**Input:** A city in California

| Representation | Constraints |
|----------------|-------------|
|----------------|-------------|



$c \in \text{city}$

$l \in \text{loc}$

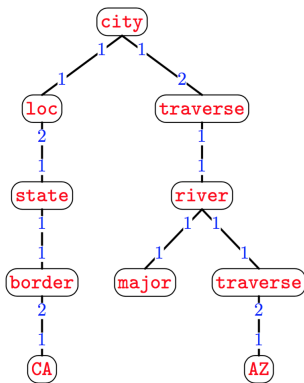
$s \in \text{CA}$

$c_1 = l_1$

$l_2 = s_1$

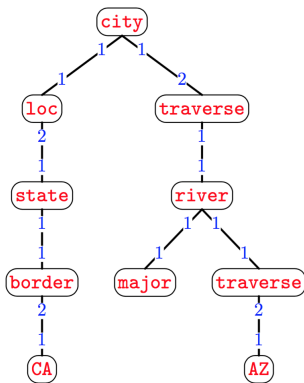
**Expansion:**  $\lambda c. \exists l. \exists s. \text{city}(c) \wedge \text{loc}(l, \text{CA}) \wedge \text{CA}(s) \wedge c_1 = l_1 \wedge l_2 = s_1$

## DCS Language: Another Join Example



- ▶ Defines a constraint satisfaction problem (CSP)
- ▶ Computing constraints can be done in linear time using dynamic programming.

## DCS Language: Another Join Example

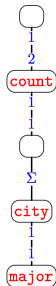


- ▶ Defines a constraint satisfaction problem (CSP)
- ▶ Computing constraints can be done in linear time using dynamic programming.
- ▶ **Tree structure:** Keeps computation and search tractable, why?

# DCS Language: Other Relations

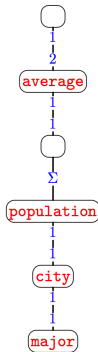
- ▶ 5 other relations: aggregate, execute, extract, quantify, compare.
- ▶ **Aggregate relation:** captures higher-order phenomena that go beyond basic CSPs.

*number of  
major cities*



(a) Counting

*average population of  
major cities*



(b) Averaging

# Comparison with Lambda Calculus (again)

## Lambda Calculus

### *Formulae*

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

### *Predicates*

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

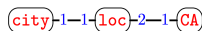
$\lambda x.\lambda y.\text{borders}(x, y)$

$\lambda p.\lambda x.p(x) \wedge \text{major}(x)$

### *Functions*

$\lambda g.\text{argmax}(g, \lambda x.\text{size}(x))$

## DCS



state

border

major

argmax



# Instantiating Predicates and Generating Trees

The most populous city in CA

- **String Match:** between words and predicate names.

# Instantiating Predicates and Generating Trees

$\text{argmax}$  CA  
The most populous city in CA

- ▶ **String Match:** between words and predicate names.
- ▶ **Function Words:** small lexicon of function words.

# Instantiating Predicates and Generating Trees

|     |        |            |            |       |
|-----|--------|------------|------------|-------|
|     |        | city       | city       |       |
|     |        | state      | state      |       |
|     | argmax | population | population | CA    |
| The | most   | populous   | city       | in CA |

- ▶ **String Match:** between words and predicate names.
- ▶ **Function Words:** small lexicon of function words.
- ▶ **Pos Tags:** Find nouns and adjectives.

# Instantiating Predicates and Generating Trees

|     |        |            |            |       |    |
|-----|--------|------------|------------|-------|----|
|     |        | city       |            | city  |    |
|     |        | state      |            | state |    |
|     | argmax | population | population |       | CA |
| The | most   | populous   | city       | in    | CA |

- ▶ **String Match:** between words and predicate names.
- ▶ **Function Words:** small lexicon of function words.
- ▶ **Pos Tags:** Find nouns and adjectives.
- ▶ **k-best parsing:** enumerate trees using k-best parser, update on good trees using variant of EM (by now a typical approach)

# Learning and Knowledge Representation

- ▶ **Big Idea:** Learning puts certain constraints on knowledge representation.
- ▶ **Theoretical Question:** What representations are needed to make the semantic learning problem tractable?

# Learning and Knowledge Representation

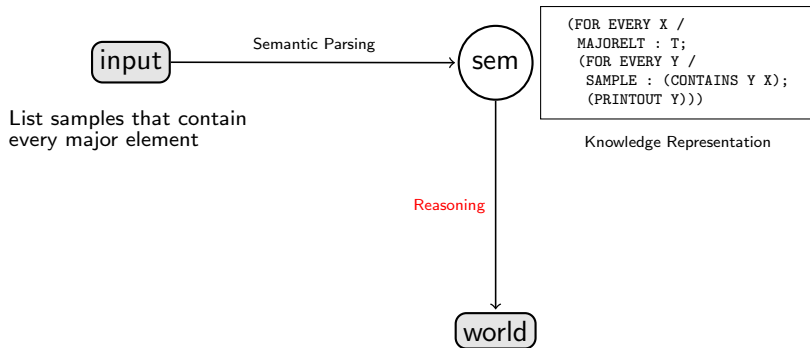
- ▶ **Big Idea:** Learning puts certain constraints on knowledge representation.
- ▶ **Theoretical Question:** What representations are needed to make the semantic learning problem tractable?
  - ▶ **Learning from Logical Form:** Generally agnostic to such questions, having target representations enforces constraints.
  - ▶ **Learning from Denotation:** Is more sensitive to type of knowledge representation, hard computational problem.

# Learning and Knowledge Representation

- ▶ **Big Idea:** Learning puts certain constraints on knowledge representation.
- ▶ **Theoretical Question:** What representations are needed to make the semantic learning problem tractable?
  - ▶ **Learning from Logical Form:** Generally agnostic to such questions, having target representations enforces constraints.
  - ▶ **Learning from Denotation:** Is more sensitive to type of knowledge representation, hard computational problem.
- ▶ Liang et al. (2011): Choose a simplified, more domain specific, version of lambda calculus, reduce to constraint satisfaction problem.

# What about Reasoning?

- How do these different subproblems interact?

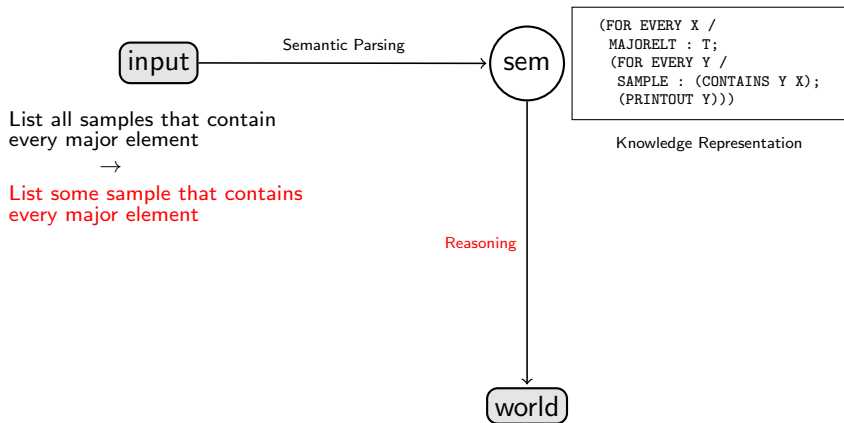


$$\llbracket sem \rrbracket = \{S10019, S10059, \dots\}$$



# What about Reasoning?

- How do these different subproblems interact?



$$\llbracket sem \rrbracket = \{S10019, S10059, \dots\} \supseteq \{\textcolor{red}{S10019}\}$$

# Recognizing Textual Entailment (RTE)

- ▶ **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** Norway's most famous painting, 'The Scream' by Edward Munch, ....

**Hypothesis:** Edward Much painted 'The Scream'

---

True

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** Google files for its long awaited IPO

**Hypothesis:** Google goes public

---

True

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** The president of Norway made an official visit to the United States

**Hypothesis:** Angela Merkel yesterday visited the United States.

---

False/Uncertain

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** The president of Norway made an official visit to the United States

**Hypothesis:** Angela Merkel yesterday visited the United States.

---

False/Uncertain

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** The president of Norway made an official visit to the United States

**Hypothesis:** Angela Merkel yesterday visited the United States.

---

False/Uncertain

- "The basic aim of semantics is to characterize the notions of a true sentence .. and of entailment" Montague (1970)

# Recognizing Textual Entailment (RTE)

- **Task:** Given a *text* and *hypothesis*, determine the following from Dagan et al. (2005):

Would a human reading *t* infer that *h* is most probably true?

**Text:** The president of Norway made an official visit to the United States

**Hypothesis:** Angela Merkel yesterday visited the United States.

---

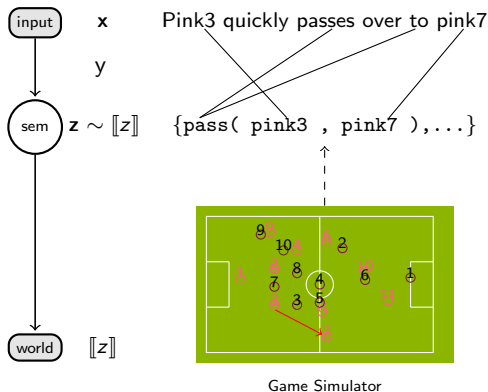
False/Uncertain

- "The basic aim of semantics is to characterize the notions of a true sentence .. and of entailment" Montague (1970)
- A type of Turing test, minimal requirement for intelligence.



# Learning to Sportscast

- ▶ Learning from “grounded” supervision.
- ▶ Minimal annotation effort.



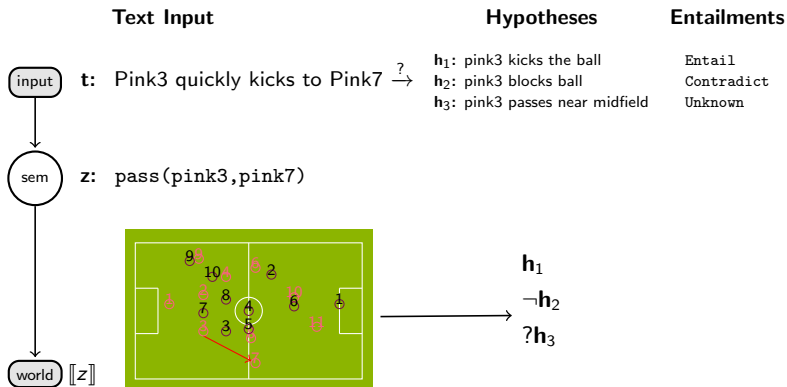
## Supervision: Dataset D

**Event Streams:**  $D = \{(x_i, \{z_1, \dots, z_k\})\}_{i=1}^N$   
**Task:** learn (latent)  $y$ , **translation**

Chen and Mooney (2008)

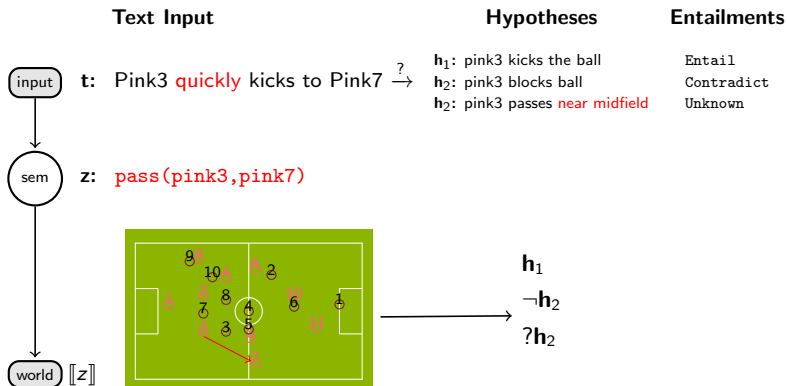
# Requirements for Semantic Representations

- **Minimal requirement:** Semantic parser should be able to recognize certain types of inferences.



# Learning from Entailment (Richardson and Kuhn (2016))

- **Goal:** Use textual entailment judgements as weak supervision to help train a semantic parser.
- Learn more precise representations and domain knowledge, account for inferential patterns.



# Motivation: Crude Representations

- ▶ Target representations are **not expressive, underspecified**
- ▶ Not based on background logical theory (no knowledge)

|   |   | <b>Entailment</b>                      |                               |
|---|---|--|-------------------------------|
| <b>Text <math>t</math></b>  | <b>Hypothesis <math>h</math></b>  | $t \rightarrow h$<br>$h \rightarrow t$ | <b>Naive</b> (do reps match?) |
| 1. Pink 3 <b>quickly</b> kicks<br>to pink 1<br><code>pass(pink3,pink1)</code> | Pink 3 kicks over to<br>pink 1 <b>near midfield</b><br><code>pass(pink3,pink1)</code> | Unknown<br>Unknown                     | Entail                        |
| 2. Purple player 10<br>kicks <b>the ball</b><br><code>kick(purple10)</code>   | Purple 10 <b>again</b><br>shoots for the goal<br><code>kick(purple10)</code>          | Unknown<br>Entail                      | Entail                        |

- ▶ **Desiderata:** **explicit treatment of modifiers**

# Motivation: Missing Knowledge

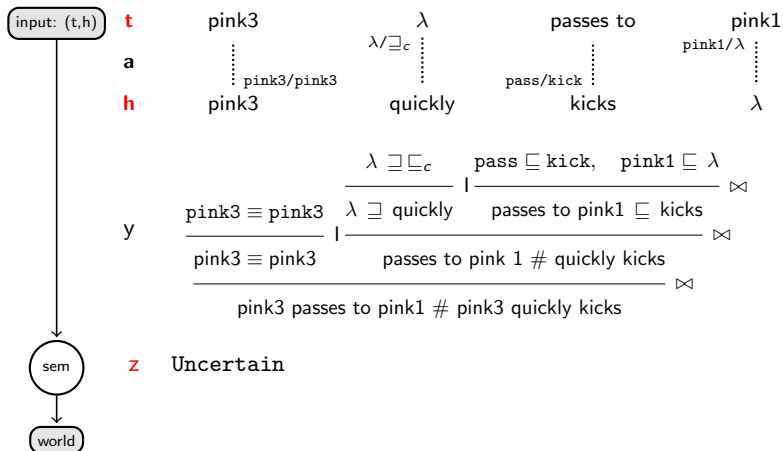
- ▶ Target representations are not expressive, underspecified.
- ▶ Not based on background logical theory (no knowledge)

|    | Text $t$   | Hypothesis $h$   | Entailment                             |                        |
|----|--|--|--|------------------------|
|    |  |  | $t \rightarrow h$<br>$h \rightarrow t$ | Naive (do reps match?) |
| 3. | Pink 10 <b>kicks</b> the ball<br><code>kick(pink10)</code>             | Pink 10 <b>passes</b> over to pink1<br><code>pass(pink10,pink1)</code>         | Unknown<br>Entail                      | Contr.                 |
| 4. | <b>Purple 7</b> makes a long <b>kick</b><br><code>kick(purple7)</code> | <b>Purple team</b> scores another <b>goal</b><br><code>playmode(goal_1)</code> | Unknown<br>Unknown                     | Contr.                 |

- ▶ **Desiderata:** explicit treatment of modifiers, sense distinctions, **abstract relations between symbols**

# Learning from Entailment

- Entailments are used to reason about target symbols and find holes in the analyses.

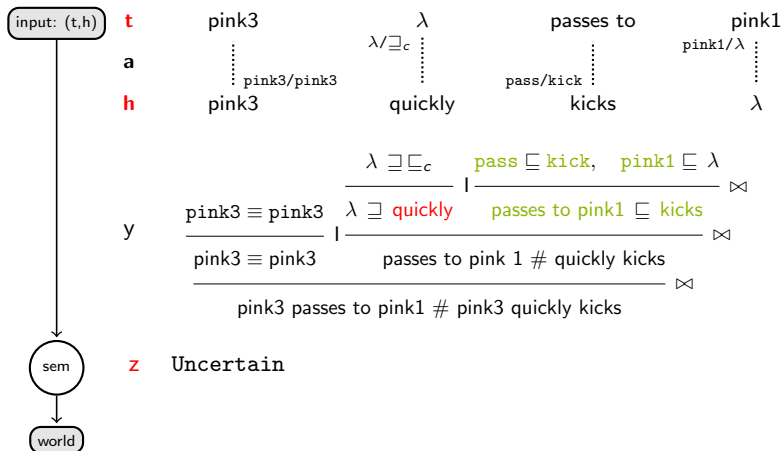


**Data:**  $D = \{((t, h)_i, z_i)\}_{i=1}^N$ , **Task:** learn (latent) proof  $y$



# Learning from Entailment

- Entailments are used to reason about target symbols and find holes in the analyses.

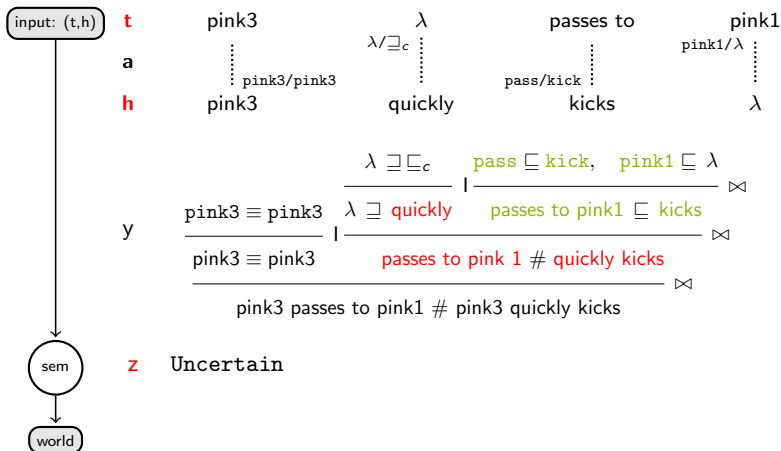


**Data:**  $D = \{((t, h)_i, z_i)\}_{i=1}^N$ , **Task:** learn (latent) proof  $y$



# Learning from Entailment

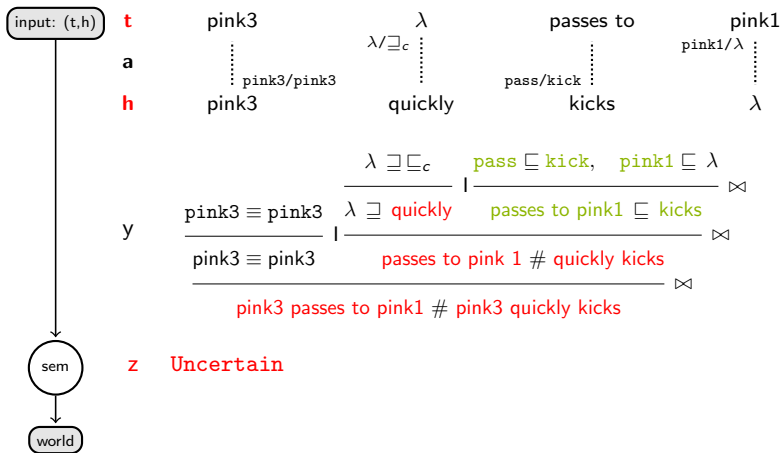
- Entailments are used to reason about target symbols and find holes in the analyses.



**Data:**  $D = \{((t, h)_i, z_i)\}_{i=1}^N$ , **Task:** learn (latent) proof y

# Learning from Entailment

- Entailments are used to reason about target symbols and find holes in the analyses.



**Data:**  $D = \{((t, h)_i, z_i)\}_{i=1}^N$ , **Task:** learn (latent) proof  $y$

# Learning from Entailment: Proofs

$$\begin{array}{c}
 \frac{\lambda \sqsupseteq \sqsubseteq_c \quad \text{pass} \sqsubseteq \text{kick}, \text{pink1} \sqsubseteq \lambda}{\lambda \sqsupseteq \text{quickly} \quad \text{passes to pink1} \sqsubseteq \text{kicks}} \quad \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \quad \lambda \sqsupseteq \text{quickly} \quad \text{passes to pink1} \sqsubseteq \text{kicks}}{\text{pink3} \equiv \text{pink3} \quad \text{passes to pink 1} \# \text{quickly kicks}} \quad \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \quad \text{passes to pink 1} \# \text{quickly kicks}}{\text{pink3 passes to pink1} \# \text{pink3 quickly kicks}} \quad \bowtie
 \end{array}$$

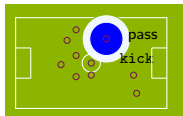
- **Logical inference:** requires logical inference, in this case using the natural logic calculus (MacCartney and Manning (2009)).

# Learning from Entailment: Proofs

$$\begin{array}{c}
 \frac{\lambda \sqsupseteq \sqsubseteq_c}{\text{pink3} \equiv \text{pink3}} \mid \frac{\text{pass} \sqsubseteq \text{kick}, \text{pink1} \sqsubseteq \lambda}{\text{passes to pink1} \sqsubseteq \text{kicks}} \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink1} \sqsubseteq \text{kicks}}{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink1} \# \text{quickly kicks}} \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink1} \# \text{quickly kicks}}{\text{pink3 passes to pink1} \# \text{pink3 quickly kicks}} \bowtie
 \end{array}$$

- **Logical inference:** requires logical inference, in this case using the natural logic calculus (MacCartney and Manning (2009)).
- $/$ : axioms, set-theoretic relations between symbols.

pass  $\sqsubseteq$  kick



# Learning from Entailment: Proofs

$$\begin{array}{c}
 \frac{\lambda \sqsubseteq \sqsubseteq_c}{\text{pink3} \equiv \text{pink3}} \mid \frac{\text{pass} \sqsubseteq \text{kick}, \text{pink1} \sqsubseteq \lambda}{\text{passes to pink1} \sqsubseteq \text{kicks}} \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink1} \sqsubseteq \text{kicks}}{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink 1} \# \text{quickly kicks}} \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \mid \text{passes to pink 1} \# \text{quickly kicks}}{\text{pink3 passes to pink1} \# \text{pink3 quickly kicks}} \bowtie
 \end{array}$$

- **Logical inference:** requires logical inference, in this case using the natural logic calculi (MacCartney and Manning (2009); Icard III (2012)).
- $/$ : axioms, set-theoretic relations between symbols.
- $\bowtie$ : natural logic join inference rule

$$\sqsubseteq \bowtie \sqsubseteq = \sqsubseteq$$

# Learning from Entailment: Proofs

$$\begin{array}{c}
 \frac{\text{pink3} \equiv \text{pink3}}{\text{pink3} \equiv \text{pink3}} \quad \bigg| \quad \frac{\lambda \sqsupseteq_c \quad \text{pass} \sqsubseteq \text{kick}, \text{pink1} \sqsubseteq \lambda}{\text{passes to pink1} \sqsubseteq \text{kicks}} \bowtie \\
 \frac{\text{pink3} \equiv \text{pink3} \quad \lambda \sqsupseteq \text{quickly} \quad \text{passes to pink1} \sqsubseteq \text{kicks}}{\text{pink3 passes to pink1} \# \text{pink3 quickly kicks}} \bowtie
 \end{array}$$

- ▶ **Logical inference:** requires logical inference, in this case using the natural logic calculi (MacCartney and Manning (2009); Icard III (2012)).
  - ▶  $/$ : axioms, set-theoretic relations between symbols.
  - ▶  $\bowtie$ : natural logic inference rules, algebraic
- ▶ **Latent variable:** axioms or relations, inference rules are constant.

# Outline of Approach

- ▶ **Step 1:** Learn a **base** semantic parser on normal data (i.e. sentences  $\rightarrow$  logic) using a PCFG approach
- ▶ **Step 2:** Retrain on inference pairs using extended **inference grammar** (i.e. sentences  $\rightarrow$  logic, pairs  $\rightarrow$  proofs).
- ▶ **What's needed:** inference dataset, logical calculus and learning algorithm.

# Outline of Approach

- ▶ **Step 1:** Learn a **base** semantic parser on normal data (i.e. sentences  $\rightarrow$  logic) using a PCFG approach
- ▶ **Step 2:** Retrain on inference pairs using extended **inference grammar** (i.e. sentences  $\rightarrow$  logic, pairs  $\rightarrow$  proofs).
- ▶ **What's needed:** inference dataset, logical calculus and learning algorithm.
- ▶ For this talk, let's assume that we have already learned a semantic grammar.



# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, inference rules (joins and functions).  
Transform  $t$  into  $h$ .

((**t**: purple7 kicks the ball, **h**: purple team scores a goal), Uncertain)

transform.                      kick  $\xrightarrow{\text{sub.}}$  score                      purple7  $\xrightarrow{\text{sub.}}$  purple team  
relation.

7

inference

# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: **semantic relations**, inference rules (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 kicks the ball, **h**: purple team scores a goal), Uncertain)

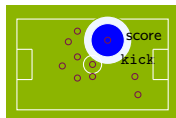
transform.

relation.

| symbol      | definition    |
|-------------|---------------|
| $\sqsubset$ | $x \subset y$ |
| $\sqsupset$ | $x \supset y$ |
| $\equiv$    | $x = y$       |
|             | neg.          |
| #           | other         |

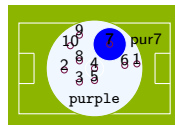
kick  $\xrightarrow{sub.}$  score

$\sqsupset$



purple7  $\xrightarrow{sub.}$  purple team

$\sqsubset$



inference

# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 kicks the ball, **h**: purple team scores a goal), Uncertain)

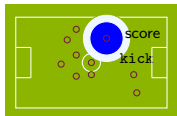
transform.

relation.

| symbol      | definition    |
|-------------|---------------|
| $\sqsubset$ | $x \subset y$ |
| $\sqsupset$ | $x \supset y$ |
| $\equiv$    | $x = y$       |
| $ $         | neg.          |
| $\#$        | other         |

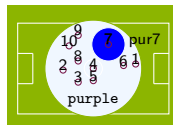
kick  $\xrightarrow{sub.}$  score

$\sqsupset$



purple7  $\xrightarrow{sub.}$  purple team

$\sqsubset$



inference

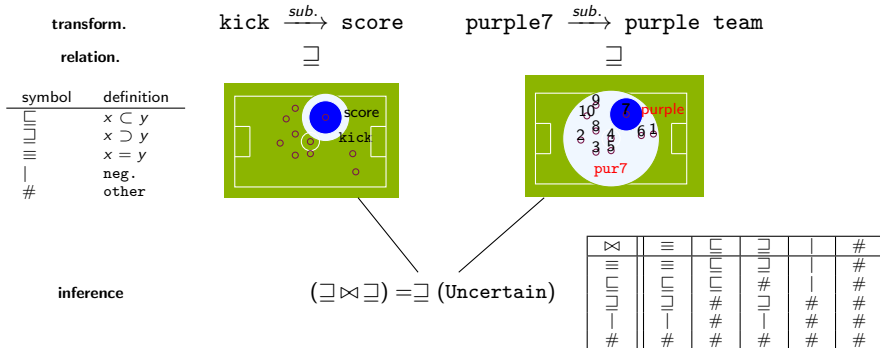
$(\sqsupset \bowtie \sqsubset) = \#(\text{Uncertain})$

|             |             |             |             |     |      |
|-------------|-------------|-------------|-------------|-----|------|
| $\bowtie$   | $\equiv$    | $\sqsubset$ | $\sqsupset$ | $ $ | $\#$ |
| $\equiv$    | $\equiv$    | $\sqsubset$ | $\sqsupset$ | $ $ | $\#$ |
| $\sqsubset$ | $\sqsubset$ | $\sqsubset$ | $\sqsupset$ | $ $ | $\#$ |
| $\sqsupset$ | $\sqsupset$ | $\sqsupset$ | $\sqsupset$ | $ $ | $\#$ |
| $ $         | $ $         | $\sqsubset$ | $\sqsupset$ | $ $ | $\#$ |
| $\#$        | $\#$        | $\sqsubset$ | $\sqsupset$ | $ $ | $\#$ |

# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 kicks the ball, **h**: purple team scores a goal), Uncertain)



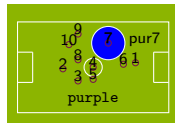
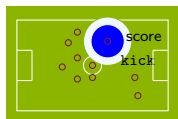
## Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 scores a goal, **h**: purple7 kicks the ball), Entail)

transform.

relation.

$$\begin{array}{ccc} \text{score} & \xrightarrow{\text{sub.}} & \text{kick} \\ \sqsubset & & \\ \text{purple7} & \xrightarrow{\text{sub.}} & \text{purple7} \\ \equiv & & \end{array}$$


**inference**

$$(\sqsubseteq \bowtie \equiv) =_{\sqsubseteq} (\text{Entail})$$

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| ⌂ | ≡ | ⌈ | ⌋ | — | # |
| ≡ | ≡ | ⌈ | ⌋ | — | # |
| ⌈ | ⌈ | ⌈ | # | — | # |
| ⌋ | ⌋ | # | # | # | # |
| — | — | # | ⌋ | # | # |
| # | # | # | # | # | # |

# Pairs to Proofs

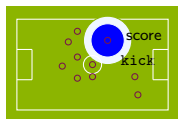
- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 scores a goal, **h**: purple7 kicks the ball **again**), Uncertain)

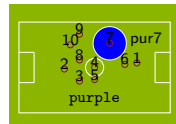
transform.  
relation.

score  $\xrightarrow{sub.}$  kick     $\lambda \xrightarrow{ins.} \sqsubseteq_c$     purple7  $\xrightarrow{sub.}$  purple7

$\sqsubseteq$                        $\sqsupseteq$  modifier                       $\equiv$



$\sqsubseteq \bowtie \sqsupseteq = \#$



inference

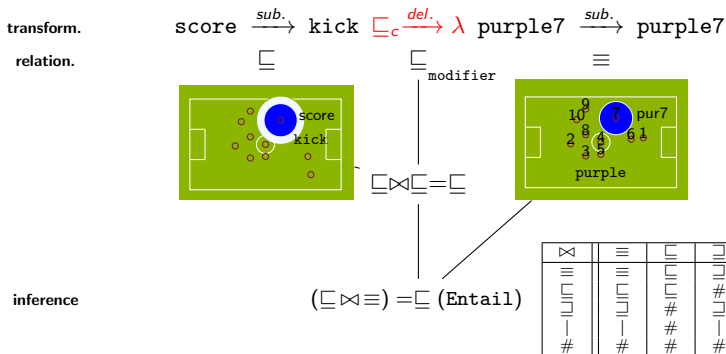
$(\# \bowtie \equiv) = \#(\text{Uncertain})$

| $\bowtie$     | $\equiv$      | $\sqsubseteq$ | $\sqsupseteq$ | $ $  | $\#$ |
|---------------|---------------|---------------|---------------|------|------|
| $\equiv$      | $\equiv$      | $\sqsubseteq$ | $\sqsupseteq$ | $ $  | $\#$ |
| $\sqsubseteq$ | $\sqsubseteq$ | $\sqsubseteq$ | $\#$          | $\#$ | $\#$ |
| $ $           | $ $           | $\#$          | $ $           | $\#$ | $\#$ |
| $\#$          | $\#$          | $\#$          | $\#$          | $\#$ | $\#$ |

# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

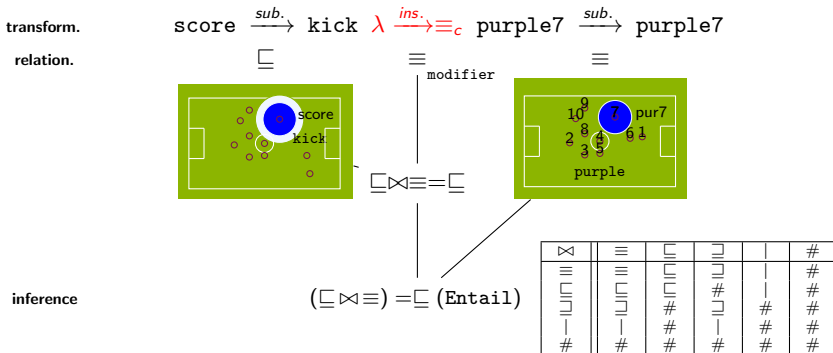
(( $t$ : purple7 scores a goal **again**,  $h$ : purple7 kicks the ball), Entail)



# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 scores a goal, **h**: purple7 kicks **the ball**), Entail)



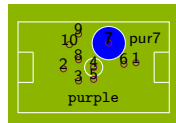
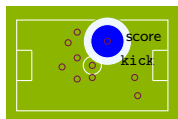


# Pairs to Proofs

- ▶ Going from pairs of text to proofs.
- ▶ two components: semantic relations, **inference rules** (joins and functions). Transform  $t$  into  $h$ .

((**t**: purple7 kicks, **h**: purple7 shoots for the goal), Uncertain)

transform.       $\text{kick} \xrightarrow{\text{sub.}} \text{kick\_1}$        $\text{purple7} \xrightarrow{\text{sub.}} \text{purple7}$   
 relation.       $\sqsupseteq$        $\equiv$



inference

$(\sqsupseteq \bowtie \equiv) = \sqsupseteq (\text{Uncertain})$

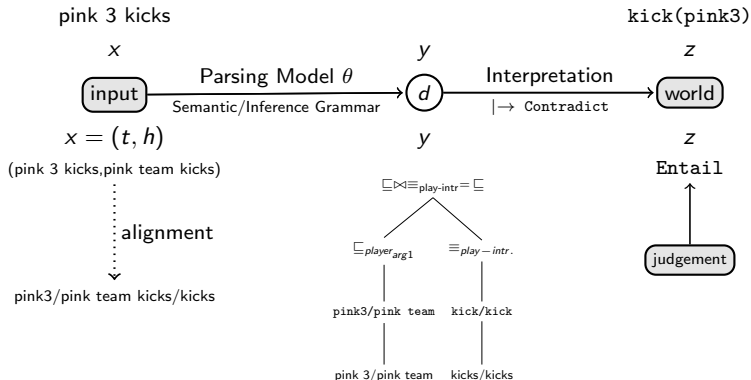
|               |               |               |               |      |      |
|---------------|---------------|---------------|---------------|------|------|
| $\bowtie$     | $\equiv$      | $\sqsupseteq$ | $\sqsupseteq$ | $ $  | $\#$ |
| $\equiv$      | $\equiv$      | $\sqsupseteq$ | $\sqsupseteq$ | $ $  | $\#$ |
| $\sqsupseteq$ | $\sqsupseteq$ | $\sqsupseteq$ | $\sqsupseteq$ | $ $  | $\#$ |
| $ $           | $ $           | $ $           | $ $           | $ $  | $\#$ |
| $\#$          | $\#$          | $\#$          | $\#$          | $\#$ | $\#$ |

# Learning from Entailment: General Idea

- ▶ Generating proofs is done jointly with learning an ordinary semantic parser, both help each other.
- ▶ Learning is done using a version of the EM algorithm.

# Learning from Entailment: General Idea

- ▶ Generating proofs is done jointly with learning an ordinary semantic parser, both help each other.
- ▶ Learning is done using a version of the EM algorithm.

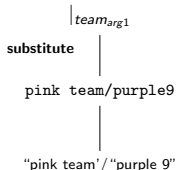


# Learned knowledge

## ► Learned lexical relations from example proof trees.

$(t, h)$ :

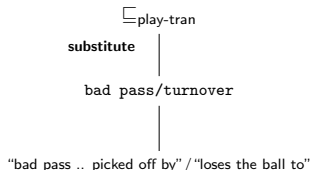
(pink team is offside, purple 9 passes)



analysis:  
relation:

pink team | purple 9

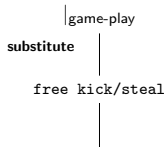
(bad pass..., loses the ball to)



bad pass ⊆ turnover

$(t, h)$ :

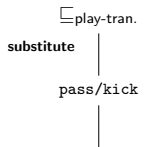
(free kick for, steals the ball from)



analysis:  
relation:

free kick | steal

(purple 6 kicks to, purple 6 kicks)



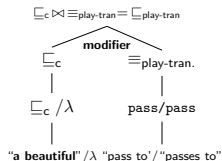
pass ⊆ kick

# Learned knowledge

## ► Learned modifiers from example proof trees.

$(t, h)$ :

(a beautiful pass to, passes to)

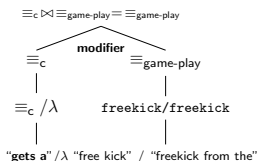


analysis:

generalization:

$\text{beautiful}(X) \sqsubseteq X$

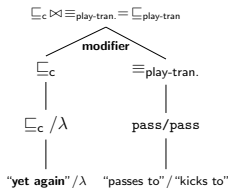
(gets a free kick, freekick from the)



$\text{get}(X) \equiv X$

$(t, h)$ :

(yet again passes to, kicks to)

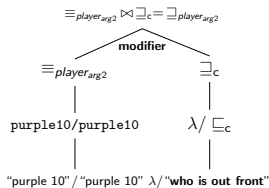


analysis:

generalization:

$\text{yet-again}(X) \sqsubseteq X$

(purple 10, purple 10 who is out front)



$X \sqsupseteq \text{out\_front}(X)$

# Conclusions

- ▶ Tried to fill in the gaps in this overall pipeline model
- ▶ While people have studied the different sub-problems independently of one another, it's important to have a holistic view of the problem.
- ▶ We looked at issues related to knowledge representation and inference.

Thank You!

# References I

- Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *in Proceedings of EMNLP-2013*, pages 1533–1544.
- Chen, D. L. and Mooney, R. J. (2008). Learning to sportscast: A test of grounded language acquisition. In *Proceedings of ICML-2008*, pages 128–135.
- Dagan, I., Glickman, O., and Magnini, B. (2005). The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*.
- Icard III, T. F. (2012). Inclusion and exclusion in natural language. *Studia Logica*, 100(4):705–725.
- Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP-2010*, pages 1223–1233.
- Liang, P., Jordan, M. I., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of ACL-11*, pages 590–599.
- Liang, P., Jordan, M. I., and Klein, D. (2013). Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.
- MacCartney, B. and Manning, C. D. (2009). An extended model of natural logic. In *Proceedings of the eighth International Conference on Computational Semantics*, pages 140–156.
- Montague, R. (1970). Universal grammar. *Theoria*, 36(3):373–398.



# References II

- Richardson, K. D. and Kuhn, J. (2016). Learning to make inferences in a semantic parsing task. *Transactions of the Association for Computational Linguistics*, 4:155–168.
- Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, pages 441–450.
- Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI-1996*, pages 1050–1055.
- Zettlemoyer, L. S. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of ACL-2009*, pages 976–984.