# AI

# Artificial Intelligence

## *8.3.5*
## *Knowledge Representation (Ch. 12)*

# Semantics of FOL

- FOL sentences can be assigned a value of *true* or *false.*

  *ISA(Milo,Cat) = true*

- *Milo is younger than Martin*

  *<(AgeOf(Milo),AgeOf(Martin)) = true*

  *=(AgeOf(Milo),AgeOf(Martin)) = false*

# Examples with Quantifiers

- All cats eat fish

  $\forall x{:}ISA(x,Cat) \Rightarrow EatFish(x)$

# Representing Events

- Martin ate
- Martin ate in the morning
- Martin ate fish
- Martin ate fish in the morning

# One Possible Representation

- FOL representations
  - Eating1(Martin)
  - Eating2(Martin,Morning)
  - Eating3(Martin,Fish)
  - Eating4(Martin,Fish,Morning)
- Meaning postulates
  - Eating4(x,y,z) -> Eating3(x,y)
  - Eating4(x,y,z) -> Eating2(x,z)
  - Eating4(x,y,z) -> Eating1(x)

Example from Jurafsky and Martin

# Second Possible Representation

- Eating4(x,y,z)
  - With some arguments unspecified
- Problems
  - Too many commitments
  - Hard to combine Eating4(Martin,Fish,z) with Eating4(Martin,y,Morning)

Example from Jurafsky and Martin

# Third Possible Representation

- Reification
  - $\exists$ e: ISA(e,Eating) $\wedge$ Eater(e,Martin) $\wedge$ Eaten(e,Fish)

Example from Jurafsky and Martin

# Outline

- Ontologies
- Categories and objects
- Events
- Times
- Beliefs

# Knowledge Representation

- Representing events, time, physical objects, and beliefs
- Upper ontology

# Upper Ontology

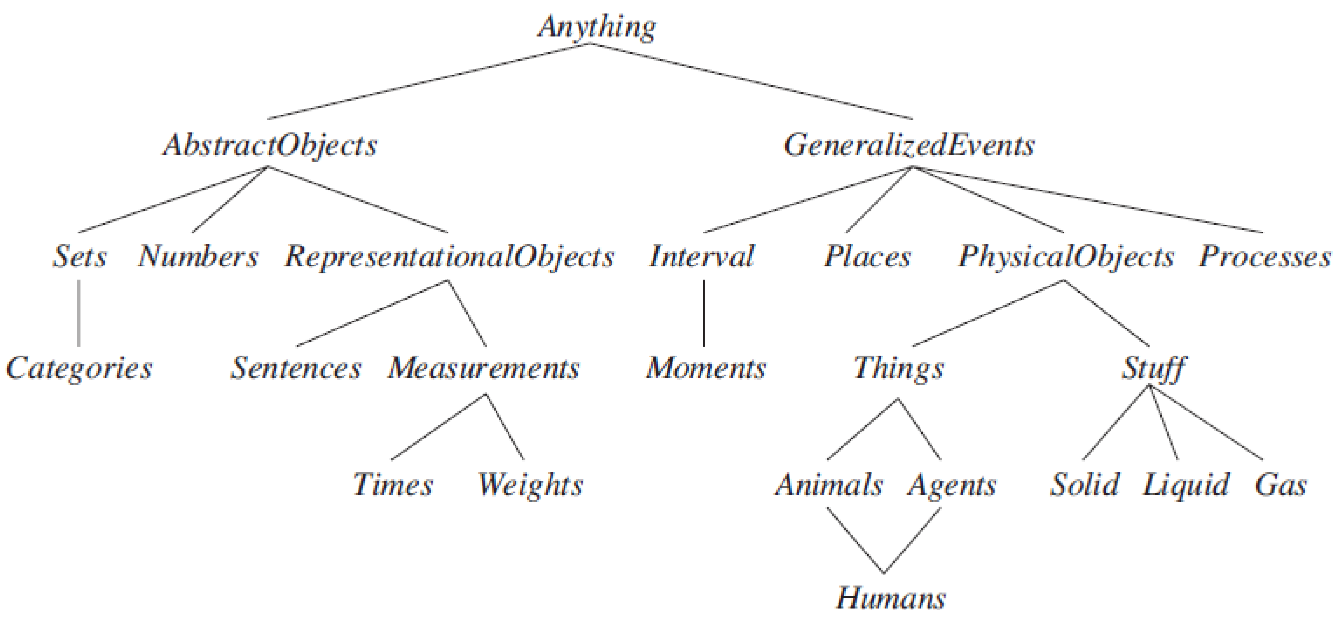



**Figure 12.1** The upper ontology of the world, showing the topics to be covered later in the chapter. Each link indicates that the lower concept is a specialization of the upper one. Specializations are not necessarily disjoint; a human is both an animal and an agent, for example. We will see in Section 12.3.3 why physical objects come under generalized events.

# Representing categories

- An object is a member of a category.
  $BB_9 \in Basketballs$
- A category is a subclass of another category.
  $Basketballs \subset Balls$
- All members of a category have some properties.
  $(x \in Basketballs) \Rightarrow Spherical(x)$
- Members of a category can be recognized by some properties.
  $Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5'' \wedge x \in Balls \Rightarrow x \in Basketballs$
- A category as a whole has some properties.
  $Dogs \in DomesticatedSpecies$

# Partitions

$$Disjoint(\{Animals, Vegetables\})$$
$$ExhaustiveDecomposition(\{Americans, Canadians, Mexicans\}, \\ NorthAmericans)$$
$$Partition(\{Males, Females\}, Animals) .$$

$$Disjoint(s) \Leftrightarrow (\forall c_1, c_2 \;\; c_1 \in s \land c_2 \in s \land c_1 \neq c_2 \Rightarrow Intersection(c_1, c_2) = \{\ \})$$
$$ExhaustiveDecomposition(s, c) \Leftrightarrow (\forall i \;\; i \in c \Leftrightarrow \exists c_2 \;\; c_2 \in s \land i \in c_2)$$
$$Partition(s, c) \Leftrightarrow Disjoint(s) \land ExhaustiveDecomposition(s, c) .$$

# PartOf relations

$$PartOf(Bucharest, Romania)$$
$$PartOf(Romania, EasternEurope)$$
$$PartOf(EasternEurope, Europe)$$
$$PartOf(Europe, Earth)\ .$$

$$PartOf(x,y) \land PartOf(y,z) \;\Rightarrow\; PartOf(x,z)\ .$$
$$PartOf(x,x)\ .$$

# Composite Objects

$$
\begin{aligned}
Biped(a) \quad \Rightarrow \quad & \exists\, l_1, l_2, b \;\; Leg(l_1) \land Leg(l_2) \land Body(b) \;\land \\
& PartOf(l_1, a) \land PartOf(l_2, a) \land PartOf(b, a) \;\land \\
& Attached(l_1, b) \land Attached(l_2, b) \;\land \\
& l_1 \neq l_2 \land [\forall\, l_3 \;\; Leg(l_3) \land PartOf(l_3, a) \;\Rightarrow\; (l_3 = l_1 \lor l_3 = l_2)]\,.
\end{aligned}
$$

# Measurements

$$Diameter(Basketball_{12}) = Inches(9.5)\ .$$
$$ListPrice(Basketball_{12}) = \$(19)\ .$$
$$d \in Days \;\Rightarrow\; Duration(d) = Hours(24)\ .$$

# Events

$$E_1 \in Flyings \land Flyer(E_1, Shankar) \land Origin(E_1, SF) \land Destination(E_1, DC)\,.$$

| | |
|---|---|
| $T(f,t)$ | Fluent $f$ is true at time $t$ |
| $Happens(e,i)$ | Event $e$ happens over the time interval $i$ |
| $Initiates(e,f,t)$ | Event $e$ causes fluent $f$ to start to hold at time $t$ |
| $Terminates(e,f,t)$ | Event $e$ causes fluent $f$ to cease to hold at time $t$ |
| $Clipped(f,i)$ | Fluent $f$ ceases to be true at some point during time interval $i$ |
| $Restored(f,i)$ | Fluent $f$ becomes true sometime during time interval $i$ |

# Time Intervals

$$Partition(\{Moments, ExtendedIntervals\}, Intervals)$$
$$i \in Moments \;\Leftrightarrow\; Duration(i) = Seconds(0)\ .$$

$$Interval(i) \;\Rightarrow\; Duration(i) = (Time(End(i)) - Time(Begin(i)))\ .$$
$$Time(Begin(AD1900)) = Seconds(0)\ .$$
$$Time(Begin(AD2001)) = Seconds(3187324800)\ .$$
$$Time(End(AD2001)) = Seconds(3218860800)\ .$$
$$Duration(AD2001) = Seconds(31536000)\ .$$

$$Time(Begin(AD2001)) = Date(0, 0, 0, 1, Jan, 2001)$$
$$Date(0, 20, 21, 24, 1, 1995) = Seconds(3000000000)\ .$$

$$Meets(ReignOf(GeorgeVI), ReignOf(ElizabethII))\ .$$
$$Overlap(Fifties, ReignOf(Elvis))\ .$$
$$Begin(Fifties) = Begin(AD1950)\ .$$
$$End(Fifties) = End(AD1959)\ .$$

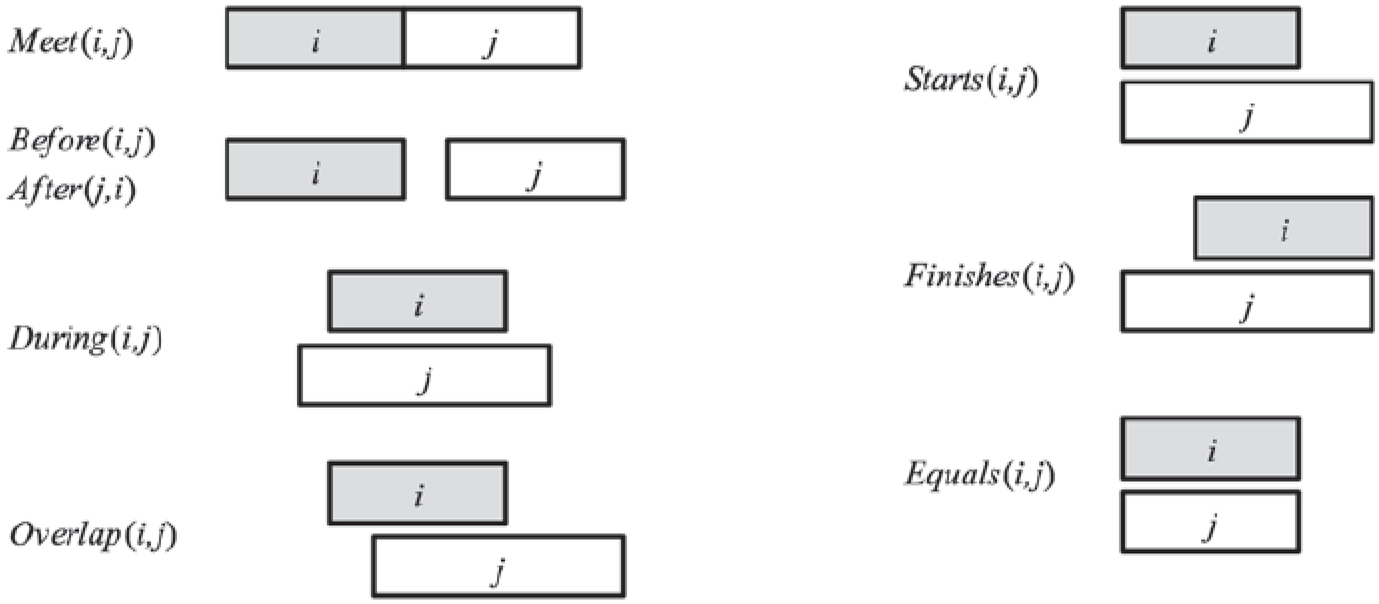



**Figure 12.2** Predicates on time intervals.

$$
\begin{array}{ll}
\begin{array}{l}
Books \subset Products \\
MusicRecordings \subset Products \\
\quad MusicCDs \subset MusicRecordings \\
Electronics \subset Products \\
\quad DigitalCameras \subset Electronics \\
\quad StereoEquipment \subset Electronics \\
\quad Computers \subset Electronics \\
\quad\quad DesktopComputers \subset Computers \\
\quad\quad LaptopComputers \subset Computers \\
\quad\quad \ldots
\end{array}
&
\begin{array}{l}
Name(\text{“books”}, Books) \\
Name(\text{“music”}, MusicRecordings) \\
\quad Name(\text{“CDs”}, MusicCDs) \\
Name(\text{“electronics”}, Electronics) \\
\quad Name(\text{“digital cameras”}, DigitalCameras) \\
\quad Name(\text{“stereos”}, StereoEquipment) \\
\quad Name(\text{“computers”}, Computers) \\
\quad\quad Name(\text{“desktops”}, DesktopComputers) \\
\quad\quad Name(\text{“laptops”}, LaptopComputers) \\
\quad\quad Name(\text{“notebooks”}, LaptopComputers) \\
\quad\quad \ldots
\end{array}
\\
\text{(a)} & \text{(b)}
\end{array}
$$

**Figure 12.9** (a) Taxonomy of product categories. (b) Names for those categories.

The logical definition of *RelevantCategoryName* is as follows:

$$
\begin{aligned}
&RelevantCategoryName(query, text) \Leftrightarrow \\
&\quad \exists\, c_1, c_2\;\; Name(query, c_1) \land Name(text, c_2) \land (c_1 \subseteq c_2 \lor c_2 \subseteq c_1)\,. \qquad (12.1)
\end{aligned}
$$

# Description Logic

$$
\begin{aligned}
&\exists\, c, \mathit{offer}\quad c \in \mathit{LaptopComputers} \wedge \mathit{offer} \in \mathit{ProductOffers} \wedge \\
&\quad \mathit{Manufacturer}(c, \mathit{IBM}) \wedge \mathit{Model}(c, \mathit{ThinkBook970}) \wedge \\
&\quad \mathit{ScreenSize}(c, \mathit{Inches}(14)) \wedge \mathit{ScreenType}(c, \mathit{ColorLCD}) \wedge \\
&\quad \mathit{MemorySize}(c, \mathit{Gigabytes}(2)) \wedge \mathit{CPUSpeed}(c, \mathit{GHz}(1.2)) \wedge \\
&\quad \mathit{OfferedProduct}(\mathit{offer}, c) \wedge \mathit{Store}(\mathit{offer}, \mathit{GenStore}) \wedge \\
&\quad \mathit{URL}(\mathit{offer}, \text{“}\texttt{example.com/computers/34356.html}\text{”}) \wedge \\
&\quad \mathit{Price}(\mathit{offer}, \$(399)) \wedge \mathit{Date}(\mathit{offer}, \mathit{Today})\,.
\end{aligned}
$$

# Artificial Intelligence

***Semantic Parsing (from 3.6.5)***

# Semantic Parsing

- Converting natural language to a logical form
  - e.g., executable code for a specific application
- Example:
  - Airline reservations
  - Geographical query systems

# Stages of Semantic Parsing

- Input
  - Sentence
- Syntactic Analysis
  - Syntactic structure
- Semantic Analysis
  - Semantic representation

# Compositional Semantics

- Add semantic attachments to CFG rules
- Compositional semantics
  - Parse the sentence syntactically
  - Associate some semantics to each word
  - Combine the semantics of words and non-terminals recursively
  - Until the root of the sentence

# Example

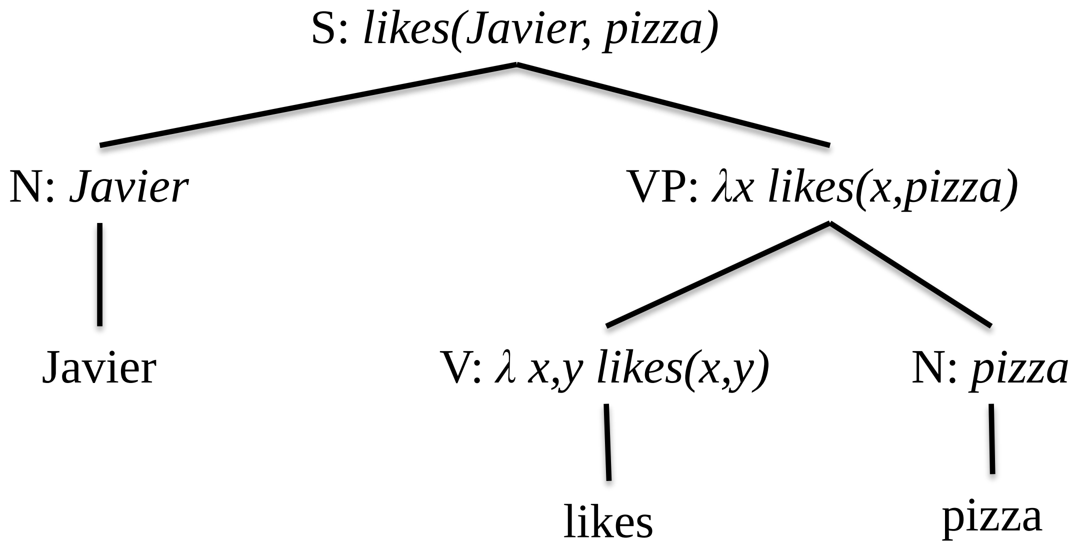
- Input
  - Javier likes pizza
- Output
  - *like(Javier, pizza)*

# Example

```
S   -> NP VP    {VP.Sem(NP.Sem)}      t
VP  -> V NP     {V.Sem(NP.Sem)}       <e,t>
NP  -> N        {N.Sem}               e
V   -> likes    {λ x,y likes(x,y)}    <e,<e,t>>
N   -> Javier   {Javier}              e
N   -> pizza    {pizza}               e
```

# Semantic Parsing

- Associate a semantic expression with each node



# Grammar with Semantic Attachments

| Grammar Rule | Semantic Attachment |
|---|---|
| $S \rightarrow NP\ VP$ | $\{NP.sem(VP.sem)\}$ |
| $NP \rightarrow Det\ Nominal$ | $\{Det.sem(Nominal.sem)\}$ |
| $NP \rightarrow ProperNoun$ | $\{ProperNoun.sem\}$ |
| $Nominal \rightarrow Noun$ | $\{Noun.sem\}$ |
| $VP \rightarrow Verb$ | $\{Verb.sem\}$ |
| $VP \rightarrow Verb\ NP$ | $\{Verb.sem(NP.sem)\}$ |
| $Det \rightarrow every$ | $\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$ |
| $Det \rightarrow a$ | $\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$ |
| $Noun \rightarrow restaurant$ | $\{\lambda r.Restaurant(r)\}$ |
| $ProperNoun \rightarrow Matthew$ | $\{\lambda m.m(Matthew)\}$ |
| $ProperNoun \rightarrow Franco$ | $\{\lambda f.f(Franco)\}$ |
| $ProperNoun \rightarrow Frasca$ | $\{\lambda f.f(Frasca)\}$ |
| $Verb \rightarrow closed$ | $\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$ |
| $Verb \rightarrow opened$ | $\{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z)$ $\wedge Opened(e,x))\}$ |

Example from Jurafsky and Martin

# Using CCG (Steedman 1996)

- CCG representations for semantics
  - *ADJ: λx.tall(x)*
  - (S\NP)/ADJ : *λf.λx.f(x)*
  - *NP: YaoMing*

$$
\dfrac{\dfrac{\text{YaoMing}}{\begin{array}{c}NP\\ YaoMing\end{array}} \quad \dfrac{\dfrac{\text{is}}{\begin{array}{c}(S\backslash NP)/ADJ\\ \lambda f.\lambda x.f(x)\end{array}} \quad \dfrac{\text{tall}}{\begin{array}{c}ADJ\\ \lambda x.tall(x)\end{array}}}{\begin{array}{c}S\backslash NP\\ \lambda x.tall(x)\end{array}}>}{\begin{array}{c}S\\ Tall\ (YaoMing)\end{array}}<
$$

# CCG Parsing

- Example:
  - https://bitbucket.org/yoavartzi/spf
- Tutorial by Artzi, FitzGerald, Zettlemoyer
  - http://yoavartzi.com/pub/afz-tutorial.acl.2013.pdf

# GeoQuery (Zelle and Mooney 1996)

What is the capital of the state with the largest population?
```
answer(C, (capital(S,C), largest(P, (state(S),
population(S,P))))).
```

What are the major cities in Kansas?
```
answer(C, (major(C), city(C), loc(C,S),
equal(S,stateid(kansas)))).
```

| Type | Form | Example |
|---|---|---|
| country | countryid(Name) | countryid(usa) |
| city | cityid(Name, State) | cityid(austin,tx) |
| state | stateid(Name) | stateid(texas) |
| river | riverid(Name) | riverid(colorado) |
| place | placeid(Name) | placeid(pacific) |

| Form | Predicate |
|---|---|
| capital(C) | C is a capital (city). |
| city(C) | C is a city. |
| major(X) | X is major. |
| place(P) | P is a place. |
| river(R) | R is a river. |
| state(S) | S is a state. |
| capital(C) | C is a capital (city). |
| area(S,A) | The area of S is A. |
| capital(S,C) | The capital of S is C. |
| equal(V,C) | variable V is ground term C. |
| density(S,D) | The (population) density of S is P |
| elevation(P,E) | The elevation of P is E. |
| high_point(S,P) | The highest point of S is P. |
| higher(P1,P2) | P1's elevation is greater than P2's. |
| loc(X,Y) | X is located in Y. |
| low_point(S,P) | The lowest point of S is P. |
| len(R,L) | The length of R is L. |
| next_to(S1,S2) | S1 is next to S2. |
| size(X,Y) | The size of X is Y. |
| traverse(R,S) | R traverses S. |

# Zettlemoyer and Collins (2005)

**a)** What states border Texas

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

**b)** What is the largest state

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

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

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

| | | |
|---|---|---|
| Utah | := | $NP$ |
| Idaho | := | $NP$ |
| borders | := | $(S\backslash NP)/NP$ |

| | | |
|---|---|---|
| Utah | := | $NP : utah$ |
| Idaho | := | $NP : idaho$ |
| borders | := | $(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$ |

a)

$$\begin{array}{ccc}
\text{Utah} & \text{borders} & \text{Idaho} \\
\hline
NP & (S\backslash NP)/NP & NP \\
utah & \lambda x.\lambda y.borders(y,x) & idaho \\
\hline
 & \multicolumn{2}{c}{(S\backslash NP) \quad >} \\
 & \multicolumn{2}{c}{\lambda y.borders(y, idaho)} \\
\hline
\multicolumn{3}{c}{S \quad <} \\
\multicolumn{3}{c}{borders(utah, idaho)}
\end{array}$$

b)

$$\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas \\
\hline
\multicolumn{2}{c}{S/(S\backslash NP) \quad >} & \multicolumn{2}{c}{(S\backslash NP) \quad >} \\
\multicolumn{2}{c}{\lambda g.\lambda x.state(x) \wedge g(x)} & \multicolumn{2}{c}{\lambda y.borders(y, texas)} \\
\hline
\multicolumn{4}{c}{S \quad >} \\
\multicolumn{4}{c}{\lambda x.state(x) \wedge borders(x, texas)}
\end{array}$$

# Zettlemoyer and Collins (2005)

| | | |
|---|---|---|
| states | := | $N : \lambda x.state(x)$ |
| major | := | $N/N : \lambda f.\lambda x.major(x) \wedge f(x)$ |
| population | := | $N : \lambda x.population(x)$ |
| cities | := | $N : \lambda x.city(x)$ |
| rivers | := | $N : \lambda x.river(x)$ |
| run through | := | $(S\backslash NP)/NP : \lambda x.\lambda y.traverse(y, x)$ |
| the largest | := | $NP/N : \lambda f.\arg\max(f, \lambda x.size(x))$ |
| river | := | $N : \lambda x.river(x)$ |
| the highest | := | $NP/N : \lambda f.\arg\max(f, \lambda x.elev(x))$ |
| the longest | := | $NP/N : \lambda f.\arg\max(f, \lambda x.len(x))$ |

Figure 6: Ten learned lexical items that had highest associated parameter values from a randomly chosen development run in the Geo880 domain.

# Dong and Lapata (2016)

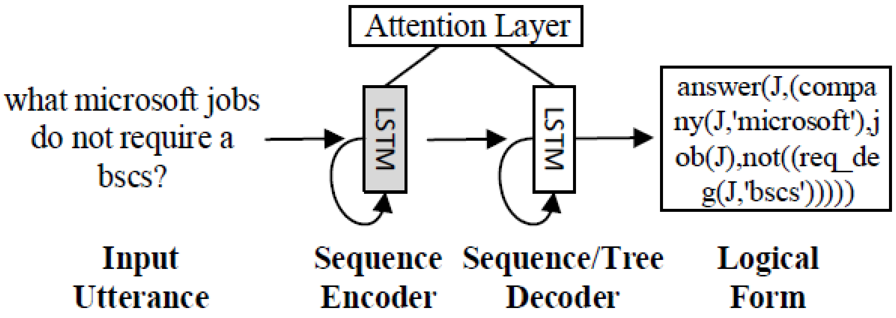



Figure 1: Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments.

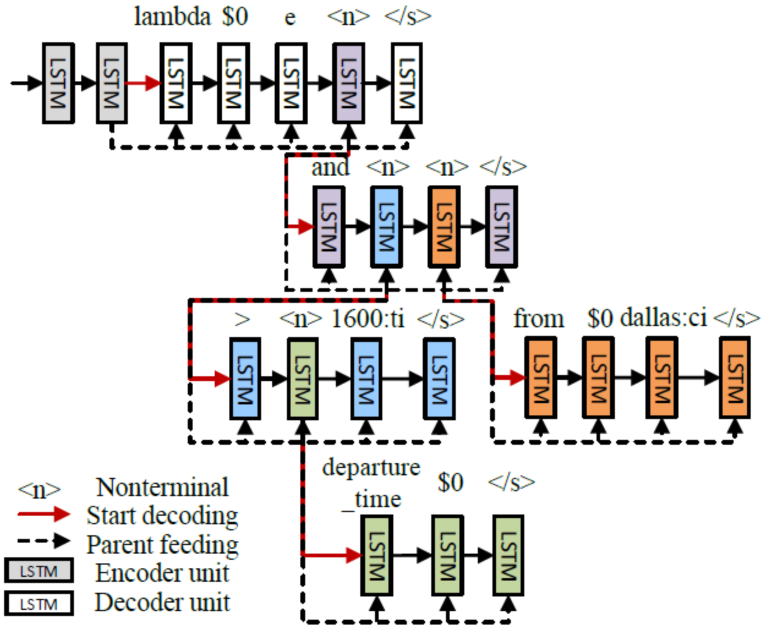



Figure 3: Sequence-to-tree (SEQ2TREE) model with a hierarchical tree decoder.

# Dong and Lapata (2016)

**JOBS** This benchmark dataset contains 640 queries to a database of job listings. Specifically, questions are paired with Prolog-style queries. We used the same training-test split as Zettlemoyer and Collins (2005) which contains 500 training and 140 test instances. Values for the variables company, degree, language, platform, location, job area, and number are identified.

**GEO** This is a standard semantic parsing benchmark which contains 880 queries to a database of U.S. geography. GEO has 880 instances split into a training set of 680 training examples and 200 test examples (Zettlemoyer and Collins, 2005). We used the same meaning representation based on lambda-calculus as Kwiatkowski et al. (2011). Values for the variables city, state, country, river, and number are identified.

**ATIS** This dataset has 5,410 queries to a flight booking system. The standard split has 4,480 training instances, 480 development instances, and 450 test instances. Sentences are paired with lambda-calculus expressions. Values for the variables date, time, city, aircraft code, airport, airline, and number are identified.

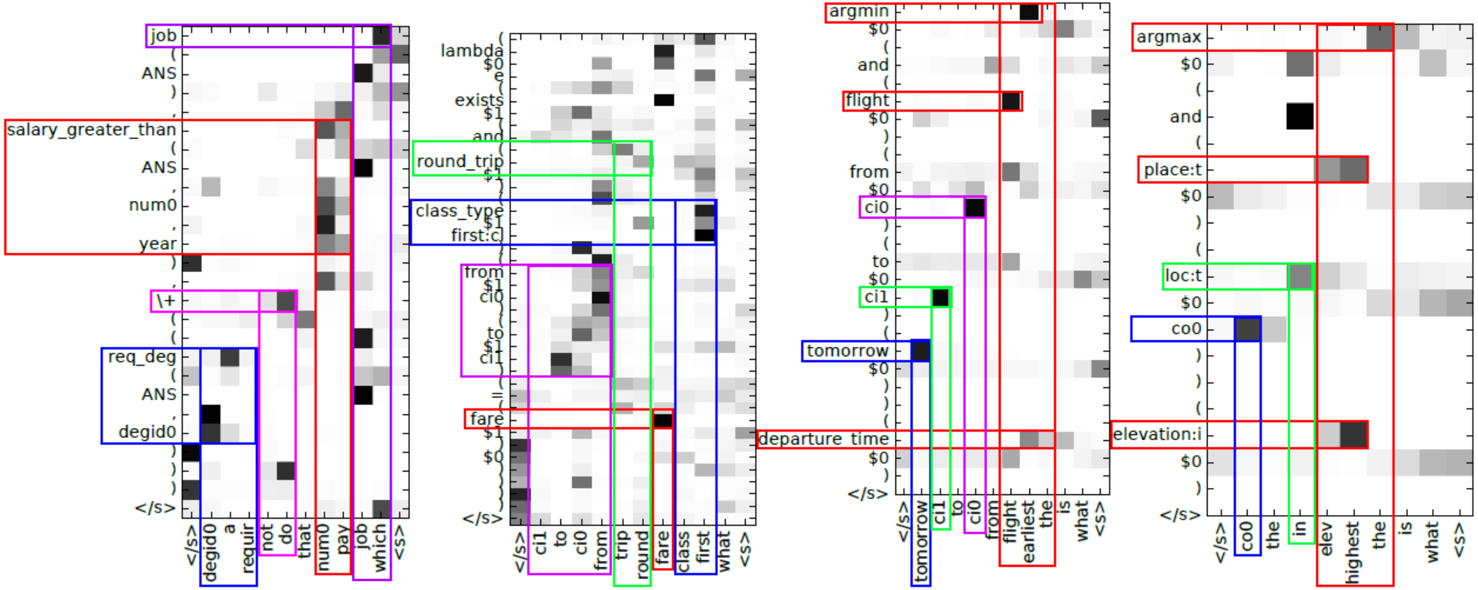
| Dataset | Length | Example |
|---|---|---|
| JOBS | 9.80 | *what microsoft jobs do not require a bscs?* |
| | 22.90 | answer(company(J,'microsoft'),job(J),not((req_deg(J,'bscs')))) |
| GEO | 7.60 | *what is the population of the state with the largest area?* |
| | 19.10 | (population:i (argmax $0 (state:t $0) (area:i $0))) |
| ATIS | 11.10 | *dallas to san francisco leaving after 4 in the afternoon please* |
| | 28.10 | (lambda $0 e (and (>(departure_time $0) 1600:ti) (from $0 dallas:ci) (to $0 san_francisco:ci))) |
| IFTTT | 6.95 | *Turn on heater when temperature drops below 58 degree* |
| | 21.80 | TRIGGER: Weather - Current_temperature_drops_below - ((Temperature (58)) (Degrees_in (f))) |
| | | ACTION: WeMo_Insight_Switch - Turn_on - ((Which_switch? (""))) |

Table 1: Examples of natural language descriptions and their meaning representations from four datasets. The average length of input and output sequences is shown in the second column.

| Method | Accuracy |
|---|---|
| COCKTAIL (Tang and Mooney, 2001) | 79.4 |
| PRECISE (Popescu et al., 2003) | 88.0 |
| ZC05 (Zettlemoyer and Collins, 2005) | 79.3 |
| DCS+L (Liang et al., 2013) | 90.7 |
| TISP (Zhao and Huang, 2015) | 85.0 |
| SEQ2SEQ | 87.1 |
| — attention | 77.9 |
| — argument | 70.7 |
| SEQ2TREE | 90.0 |
| — attention | 83.6 |

Table 2: Evaluation results on JOBS.

# Dong and Lapata (2016)



(a) which jobs pay num0 that do not require a degid0

(b) what's first class fare round trip from ci0 to ci1

(c) what is the earliest flight from ci0 to ci1 tomorrow

(d) what is the highest elevation in the co0

Figure 6: Alignments (same color rectangles) produced by the attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed. Model output is shown for SEQ2SEQ (a, b) and SEQ2TREE (c, d).

# Jia and Liang 2016

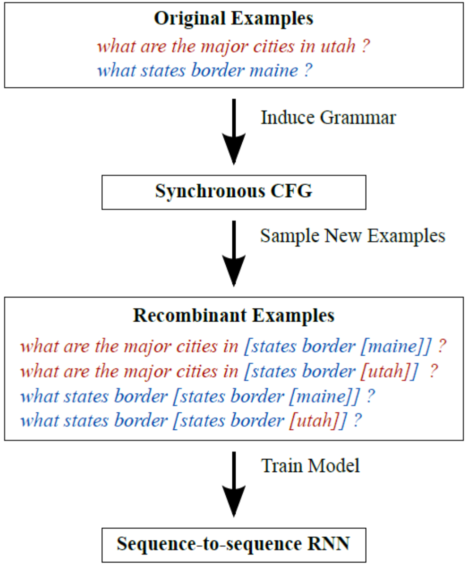



Figure 1: An overview of our system. Given a dataset, we induce a high-precision synchronous context-free grammar. We then sample from this grammar to generate new "recombinant" examples, which we use to train a sequence-to-sequence RNN.

**GEO**

$x$: *"what is the population of iowa ?"*

$y$:
```
_answer ( NV , (
 _population ( NV , V1 ) , _const (
  V0 , _stateid ( iowa ) ) ) )
```

**ATIS**

$x$: *"can you list all flights from chicago to milwaukee"*

$y$:
```
( _lambda $0 e ( _and
 ( _flight $0 )
 ( _from $0 chicago : _ci )
 ( _to $0 milwaukee : _ci ) ) )
```

**Overnight**

$x$: *"when is the weekly standup"*

$y$:
```
( call listValue ( call
  getProperty meeting.weekly_standup
  ( string start_time ) ) )
```

Figure 2: One example from each of our domains. We tokenize logical forms as shown, thereby casting semantic parsing as a sequence-to-sequence task.

**Examples**

(“*what states border texas ?*”,
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas)))))`
(“*what is the highest mountain in ohio ?*”,
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio))))))`

**Rules created by AbsEntities**

Root → ⟨ “*what states border* StateId *?*”,
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(`StateId `)))))`⟩
StateId → ⟨ “*texas*”, `texas` ⟩
Root → ⟨ “*what is the highest mountain in* StateId *?*”,
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),`
`const(V0, stateid(`StateId `))))))`⟩
StateId → ⟨“*ohio*”, `ohio`⟩

**Rules created by AbsWholePhrases**

Root → ⟨ “*what states border* State *?*”, `answer(NV, (state(V0), next_to(V0, NV),` State `))`⟩
State → ⟨ “*states border texas*”, `state(V0), next_to(V0, NV), const(V0, stateid(texas))`⟩
Root → ⟨ “*what is the highest mountain in* State *?*”,
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),` State `))))`⟩

**Rules created by Concat-2**

Root → ⟨$\text{Sent}_1$ </s> $\text{Sent}_2$, $\text{Sent}_1$ </s> $\text{Sent}_2$⟩
Sent → ⟨ “*what states border texas ?*”,
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas)))))` ⟩
Sent → ⟨ “*what is the highest mountain in ohio ?*”,
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio))))))` ⟩

Figure 3: Various grammar induction strategies illustrated on Geo. Each strategy converts the rules of an input grammar into rules of an output grammar. This figure shows the base case where the input grammar has rules Root $\to \langle x, y \rangle$ for each $(x, y)$ pair in the training dataset.

AI