AUGMENTED GRAMMARS AND SEMANTIC INTERPRETATION

AUGMENTED GRAMMARS AND SEMANTIC INTERPRETATION

- Lexicalized PCFGs
- Formal definition of augmented grammar rules
- Case agreement and subject–verb agreement
- Semantic interpretation
- Complications

Lexicalized PCFGs

- The probabilities for a rule depend on the relationship between words in the parse tree, not just on the adjacency of words in a sentence.
- The **head** of a phrase—the most important word.
- The relationship between the verb “eat” and the nouns “banana” versus “bandanna,”
- In this “eat” is the head of the VP
- “eat a banana” and “banana” is the head of the NP “a banana.”
- We use the notation VP(v) to denote a phrase with category VP whose head word is v.
- The category VP is **augmented** with the head variable v

Lexicalized PCFGs - Augmented Grammar

- an **augmented grammar** that describes the verb–object relation:
- the probability P1(v, n) depends on the head words v and n.

$$
\begin{array}{lll}
VP(v) \rightarrow Verb(v)\ NP(n) & & [P_1(v,n)] \\
VP(v) \rightarrow Verb(v) & & [P_2(v)] \\
NP(n) \rightarrow Article(a)\ Adjs(j)\ Noun(n) & & [P_3(n,a)] \\
Noun(\mathbf{banana}) \rightarrow \mathbf{banana} & & [p_n] \\
\ldots & & \ldots
\end{array}
$$

- Probability to be relatively high (P1) when v is “eat” and n is “banana,” and low (P2) when n is “bandanna.”
- These objectless probabilities are still very useful;
- capture the distinction between a transitive verb like “eat”—which will have a high value for P1
- Learn these probabilities from a **treebank.**

Formal definition of augmented grammar rules

- Augmented rules are complicated
- The sentence will have the form of a definite clause, so the result is called a **definite clause grammar**, or DCG.
- example
- A rule from the lexicalized grammar for NP with notation:
- NP(n) → Article(a) Adjs(j) Noun(n) {Compatible (j, n)} .

- The notation {constraint} to denote a logical constraint on some of the variables; then, the rule only holds when the constraint is **true**

- The predicate Compatible(j,n) is to test whether adjective j and noun n are compatible;
- Example, Compatible (**black**, **dog**).

Convert Grammar Rule Into A Definite Clause

- We can convert the grammar rule into a definite clause by
- (1) reversing the order of right- and left-hand sides,
- (2) making a conjunction of all the constituents and constraints,
- (3) adding a variable s_i to the list of arguments for each constituent to represent the sequence of words spanned by the constituent,
- (4) adding a term for the **concatenation of words**, Append(s1, . . .), to the list of arguments for the root of the tree.
- **Article(a, s1) ∧ Adjs(j, s2) ∧ Noun(n, s3) ∧ Compatible (j, n)**
- **⇒ NP(n, Append(s1, s2, s3)) .**

- **Article(a, s1) ∧ Adjs(j, s2) ∧ Noun(n, s3) ∧ Compatible (j, n)**
- **⇒ NP(n, Append(s1, s2, s3)) .**

- if the predicate Article is true of a head word a and a string s1, and
- Adjs is similarly true of a head word j and a string s2, and
- Noun is true of a head word n and a string s3, and
- if j and n are compatible,
- then the predicate NP is true of the head word n and
- the result of **appending** strings s1, s2, and s3.

- The translation from grammar rule to definite clause means parsing as logical inference.
- many different ways.
- **bottom-up parsing using forward chaining or**
- **top-down parsing using backward chaining.**

Case agreement and subject–verb agreement

- The pronoun “I” is in the subjective case, and “me” is in the objective case.
- The grammar would have to know that “me” is not a valid NP when it is the subject of a sentence.
- Example - “Me smell a stench.”
- Hence, split NP into two categories, NP_S and NP_O, to stand for noun phrases in the subjective and objective case, respectively.
- split the category Pronoun into the two categories $Pronoun_S$ (which includes “I”) and $Pronoun_O$ (which includes “me”).

grammar for **case agreement**;

- Part of a grammar for the resulting language E1
- E1 handles subjective and objective cases in noun phrases
- All the NP rules must be duplicated, once for NP_S and once for NP_O

$$
\begin{array}{lrcl}
\mathcal{E}_1: & S & \rightarrow & NP_S\ VP \mid \ldots \\
& NP_S & \rightarrow & Pronoun_S \mid Name \mid Noun \mid \ldots \\
& NP_O & \rightarrow & Pronoun_O \mid Name \mid Noun \mid \ldots \\
& VP & \rightarrow & VP\ NP_O \mid \ldots \\
& PP & \rightarrow & Prep\ NP_O \\
& Pronoun_S & \rightarrow & \textbf{I} \mid \textbf{you} \mid \textbf{he} \mid \textbf{she} \mid \textbf{it} \mid \ldots \\
& Pronoun_O & \rightarrow & \textbf{me} \mid \textbf{you} \mid \textbf{him} \mid \textbf{her} \mid \textbf{it} \mid \ldots
\end{array}
$$

- Unfortunately, the Language E1 still over-generates.
- English requires **subject–verb agreement** for person and number of the subject and main verb of a sentence.
- For example, if “I” is the subject, then “I smell” is grammatical, but “I smells” is not.
- If “it” is the subject, we get the reverse.
- In English, the agreement distinctions are minimal:
- most verbs have one form for third-person singular subjects (he, she, or it), and a second form for all other combinations of person and number.
- There is one exception: the verb “to be” has three forms, “I am / you are / he is.”
- So one distinction (case) splits NP two ways,
- another distinction (person and number) splits NP three ways.
- Augmentations are a better approach: they can represent an exponential number of forms as a single rule.

An Augmented Grammar For The Language E2,

- part of an augmented grammar for Language E2,
- with three augmentations:
- case agreement,
- subject–verb agreement, and
- head word.
- *Sbj, Obj, 1S, 1P* and *3P* are constants, and
- lowercase names are variables.

$$
\begin{array}{lrcl}
\mathcal{E}_2: & S(head) & \rightarrow & NP(Sbj, pn, h)\ VP(pn, head) \mid \ \ldots \\
& NP(c, pn, head) & \rightarrow & Pronoun(c, pn, head) \mid\ Noun(c, pn, head) \mid\ \ldots \\
& VP(pn, head) & \rightarrow & VP(pn, head)\ NP(Obj, p, h)\ \mid\ \ldots \\
& PP(head) & \rightarrow & Prep(head)\ NP(Obj, pn, h) \\
& Pronoun(Sbj, 1S, \mathbf{I}) & \rightarrow & \mathbf{I} \\
& Pronoun(Sbj, 1P, \mathbf{we}) & \rightarrow & \mathbf{we} \\
& Pronoun(Obj, 1S, \mathbf{me}) & \rightarrow & \mathbf{me} \\
& Pronoun(Obj, 3P, \mathbf{them}) & \rightarrow & \mathbf{them} \\
& & & \\
& & & \ldots
\end{array}
$$

An Augmented Grammar For The Language E2…

- In Language E2 has one NP category, but NP(c, pn, head) has three augmentations:
  - c is a parameter for case,
  - pn is a parameter for person and number, and
  - head is a parameter for the head word of the phrase.
- The other categories also are augmented with heads and other arguments.
- Let's consider one rule in detail:
- S(head) → NP(Sbj , pn, h) VP(pn, head) .

An Augmented Grammar For The Language E2…

- S(head) → NP(Sbj , pn, h) VP(pn, head) .
- This rule is easiest to understand right-to-left:
- when an *NP* and a *VP* are conjoined they form an *S*,
- but only if the *NP* has the subjective (*Sbj*) case and
- the person and number (*pn*) of the *NP* and *VP* are identical.
- *And S(head) = VP(head)*
- Note the head of the *NP*, denoted by the dummy variable *h*, is not part of the augmentation of the *S*.

An Augmented Grammar For The Language E2…

- The lexical rules for E2 fill in the values of the parameters and are also best read right-to-left.
- For example, the rule
- Pronoun(Sbj , 1S, **I**) → **I**
- Here, "I" as a *Pronoun* in the subjective case, first-person singular, with head "I."
- Augmentation can also work with automated learning mechanisms, learning algorithm can automatically split the NP category into NP_S and NP_O.

Semantic Interpretation

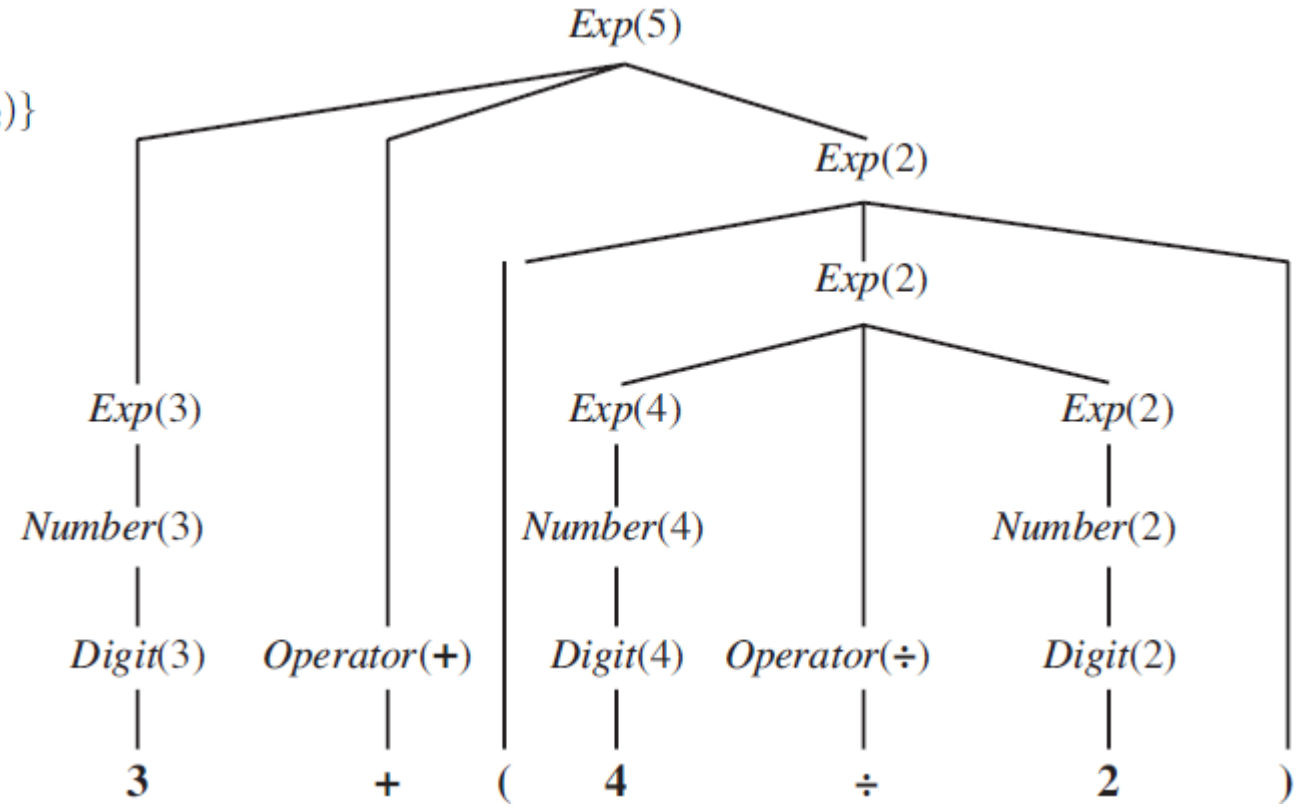
- Adding semantics to a grammar.
- Example - A grammar for arithmetic expressions, augmented with semantics.
- Each variable xi represents the semantics of a constituent.
- Each rule is augmented with a variable indicating the semantic interpretation of the phrase.

$$
\begin{aligned}
Exp(x) &\rightarrow Exp(x_1)\ Operator(op)\ Exp(x_2)\ \{x = Apply(op, x_1, x_2)\} \\
Exp(x) &\rightarrow (\ Exp(x)\ ) \\
Exp(x) &\rightarrow Number(x) \\
Number(x) &\rightarrow Digit(x) \\
Number(x) &\rightarrow Number(x_1)\ Digit(x_2)\ \{x = 10 \times x_1 + x_2\} \\
Digit(x) &\rightarrow x\ \{0 \leq x \leq 9\} \\
Operator(x) &\rightarrow x\ \{x \in \{+, -, \div, \times\}\}
\end{aligned}
$$

- The semantics of a digit such as “3” is the digit itself.
- The semantics of an expression such as “3 + 4” is the operator “+” applied to the semantics of the phrase “3” and the phrase “4.”
- The rules obey the principle of **compositional semantics**—the semantics of a phrase is a function of the semantics of the subphrases

Parse tree with semantic interpretations for the string “3 + (4 ÷ 2)”.

$$
\begin{aligned}
&Exp(x) \rightarrow Exp(x_1)\ Operator(op)\ Exp(x_2)\ \{x = Apply(op, x_1, x_2)\}\\
&Exp(x) \rightarrow (\ Exp(x)\ )\\
&Exp(x) \rightarrow Number(x)\\
&Number(x) \rightarrow Digit(x)\\
&Number(x) \rightarrow Number(x_1)\ Digit(x_2)\ \{x = 10 \times x_1 + x_2\}\\
&Digit(x) \rightarrow x\ \{0 \leq x \leq 9\}\\
&Operator(x) \rightarrow x\ \{x \in \{+, -, \div, \times\}\}
\end{aligned}
$$



First Order Logic (FOL)

- First-order logic is also known as Predicate logic or First-order predicate logic.
- First-order logic, like natural language has well defined syntax and semantics.
- It assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Properties of FOL

- It has ability to represent facts about some or all of the objects and relations in the universe
- Represent law and rules extracted from real world
- Useful language for Maths, Philosophy and AI
- Represent facts in realistic manner rather than just true / false
- Makes ontological commitment

Syntax of FOL: Basic Elements

- Constants — KingJohn, 2, NUS,...
- Predicates — Brother, >,...
- Functions — Sqrt, LeftLegOf, ...
- Variables — x, y, a, b,...
- Connectives — $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- Equality — $=$
- Quantifiers — $\forall$, $\exists$
- Sentences — Atom / complex sentences
- Atom — True / False / AP (atomic proposition)
- Complex sentence — (sentence) / connective sentence / $\neg$ sentence

Atomic sentences

- Atomic sentence = *predicate* ($term_1$,...,$term_n$) or $term_1$ = $term_2$
- Term = *function* ($term_1$,...,$term_n$) or *constant* or *variable*

- Example.
- *Brother(KingJohn,RichardTheLionheart)*
- *> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

- Complex sentences are made from atomic sentences using five logical connectives
- S1 and S2 are two atomic statements then
- $\neg S1$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$,

E.g. *Sibling(KingJohn,Richard)* $\Leftrightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

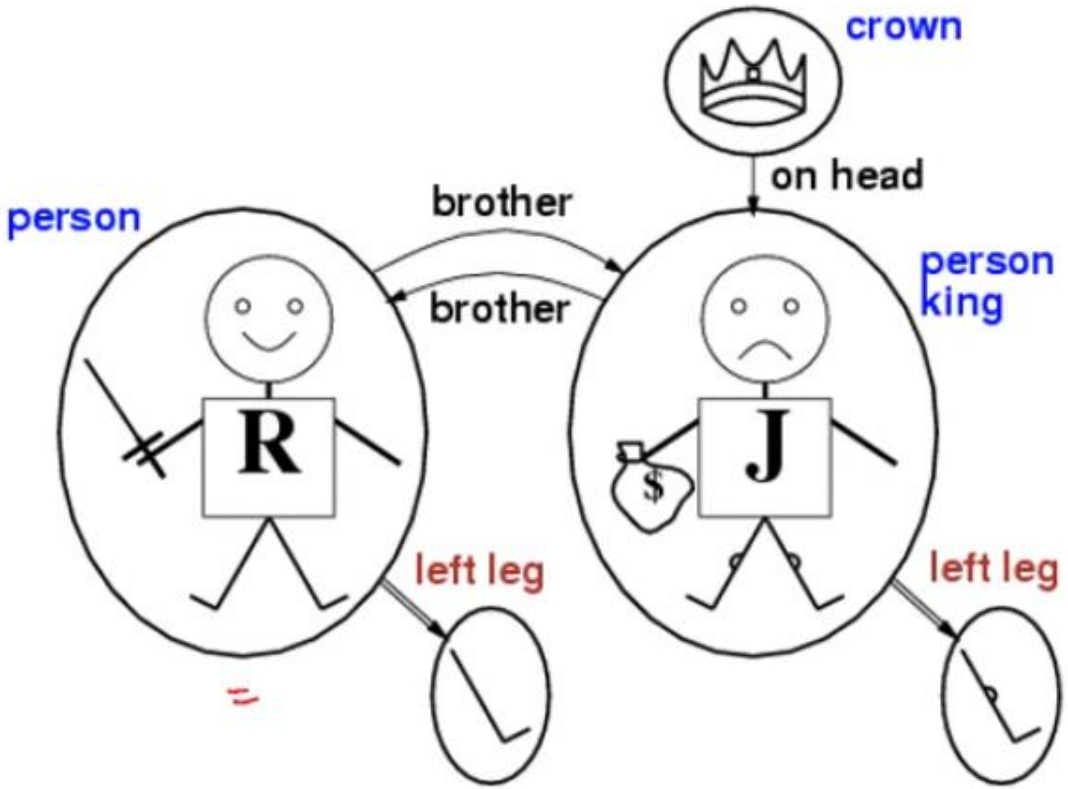
Truth in FOL

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for
  Constant Symbols → Objects
  Predicate Symbols → Relations
  Function Symbols → Functional relations
- An atomic sentence $predicate(term_1,...,term_n)$ is true iff the objects referred by $term_1,...,term_n$ are in the relation referred by *predicate*

Models for FOL: Example

- Richard the Lionheart was a King of England from 1189 to 1199;
- his younger brother was the evil King John, who ruled from 1199 to 1215;
- The left legs of Richard and John were different;
- and John had a crown (Because he was king).

Models for FOL: Example



- Objects :
  - Person king John
  - Person Richard
  - Crown
  - Left leg of John
  - Left leg of Richard
- Relation :
  - “onhead” <the crown, king John>
  - “brother” <John, Richard >
  - “person” <John>
  - “person” <Richard>
  - “king” <John>
- Function:
  - [no other person wears crown except the king]
  - <John the king> - on-head(crown)
  - <John the king> - shoe(left-leg)

Semantics of English

- Example sentence “John Loves Mary”
- The NP “John” should have as its semantic interpretation the logical term John
- The logical sentence Loves(John, Mary).
- The complicated part is the VP “loves Mary.”
- The semantic interpretation of this phrase is neither a logical term nor a complete logical sentence.
- “loves Mary” is a **predicate**, when combined with a term that represents a person yields (John) a complete logical sentence.

Semantics of English…

- “loves Mary” as the predicate
- λx Loves(x, Mary) .
- NP with semantics obj followed by a VP with semantics pred yields a sentence whose semantics is the result of applying pred to obj :
- S(pred (obj)) → NP(obj) VP(pred) .
- the semantic interpretation of “John loves Mary” is
- (λx Loves(x, Mary))(John) ,
- which is equivalent to Loves(John, Mary).

Semantics of English…

- VPs are represented as predicates
- The verb “loves” is represented as λy λx Loves(x, y), the predicate that,
- when given the argument Mary, returns the predicate
- λx Loves(x, Mary).

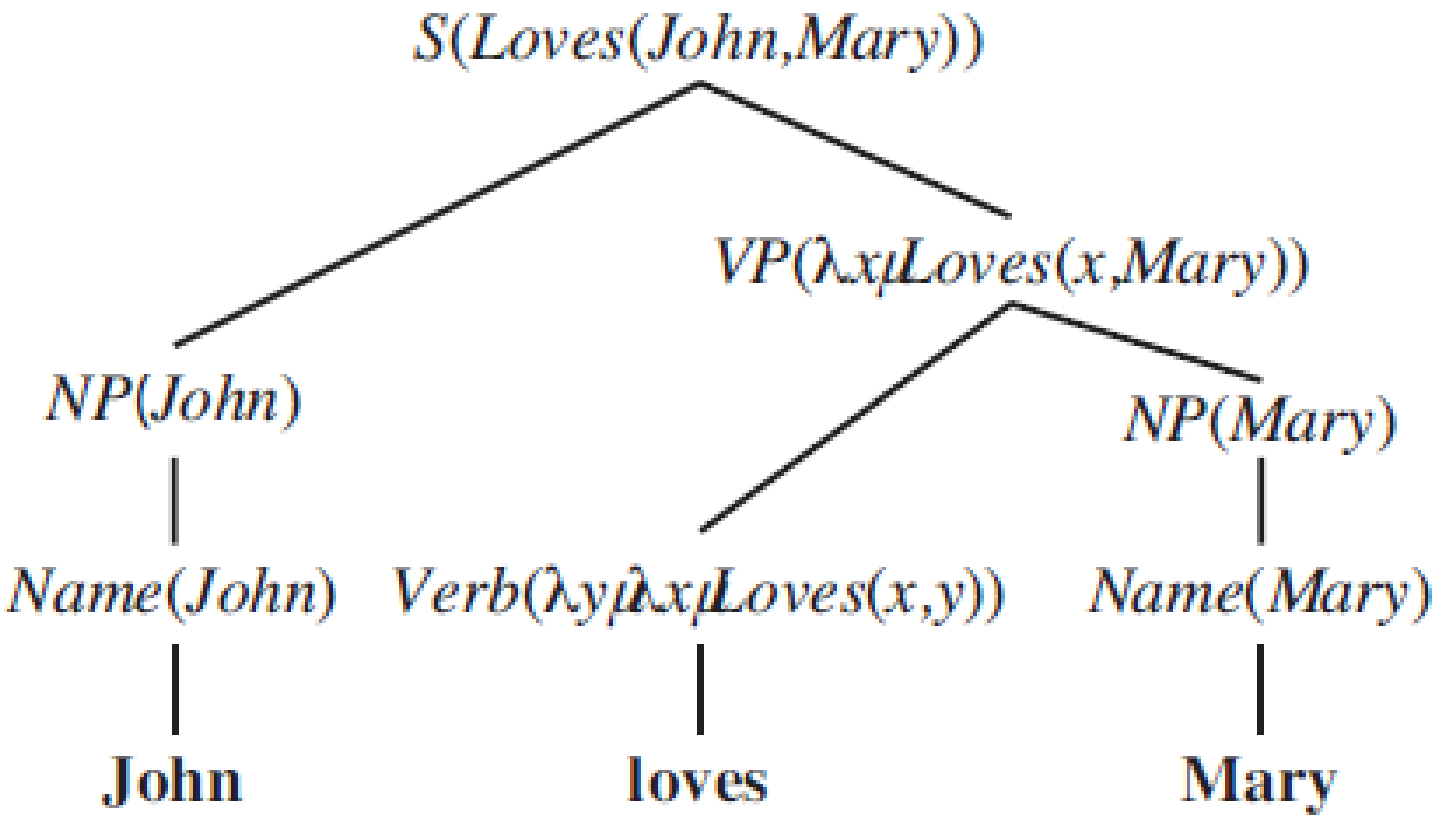
Semantics of English…

- A grammar that can derive a parse tree and semantic interpretation for “John loves Mary” (and three other sentences).
- Each category is augmented with a single argument representing the semantics.

$$S(pred(obj)) \rightarrow NP(obj)\ VP(pred)$$
$$VP(pred(obj)) \rightarrow Verb(pred)\ NP(obj)$$
$$NP(obj) \rightarrow Name(obj)$$

$$Name(John) \rightarrow \textbf{John}$$
$$Name(Mary) \rightarrow \textbf{Mary}$$
$$Verb(\lambda y\ \lambda x\ Loves(x, y)) \rightarrow \textbf{loves}$$

A parse tree with semantic interpretations for the string “John loves Mary”.



Inductive Logic Programming (ILP)

- An Inductive Logic Programming (ILP) program that learns a grammar and a specialized parser for that grammar from examples.
- The target domain is natural language database queries.
- The training examples consist of pairs of word strings and corresponding **semantic forms**.
- for example;
- What is the capital of the state with the largest population?
- Answer(c,Capital(s, c) ∧ Largest(p, State(s) ∧ Population(s, p)))

Complications

- The grammar of real English is endlessly complex.
- **Time and tense**: - present, past, future , today, tomorrow…
- **Quantification**: - all, few, some, many, lot, huge …
- **Pragmatics (realistic, practical truth)**: (explains the situation)
- **Long-distance dependencies**:
- **Ambiguity**:
  - **Lexical ambiguity**,
  - **Syntactic ambiguity**
  - **semantic ambiguity**,
  - **Metaphor**
- **Disambiguation**
  - The **world model**
  - The **mental model**:
  - The **language model**:
  - The **acoustic model**:

Time and tense:

- “John loves Mary” and “John loved Mary” are different.
- English uses verb tenses (past, present, and future) to indicate the relative time of an event.
- The event calculus notation is used to represent the time of events.

Time and tense:

- John loves Mary: E1 ∈ Loves(John, Mary) ∧ During(Now, Extent(E1))
- John loved Mary: E2 ∈ Loves(John, Mary) ∧ After(Now, Extent(E2)) .
- This suggests that our two lexical rules for the words “loves” and “loved” should be these:
- Verb(λy λx e ∈ Loves(x, y) ∧ During(Now, e)) → **loves**
- Verb(λy λx e ∈ Loves(x, y) ∧ After(Now, e)) → **loved** .

- “John slept a lot last night,”
- where Sleeping is a process category, but it is odd to say
- “John found a unicorn a lot last night,”
- where Finding is a discrete event category.
- A grammar would reflect that fact by having a low probability for adding the adverbial phrase “a lot” to discrete events.

Quantification:

- Consider the sentence “Every agent feels a breeze.”
- The sentence has only one syntactic parse under E0, but it is actually semantically ambiguous;
- the preferred meaning is
- “For every agent there exists a breeze that the agent feels,”
- but an acceptable alternative meaning is
- “There exists a breeze that every agent feels.”

- “For every agent there exists a breeze that the agent feels,”
- “There exists a breeze that every agent feels.”
- The two interpretations can be represented as

$$\forall a \;\; a \in Agents \;\Rightarrow$$
$$\exists b \;\; b \in Breezes \wedge \exists e \;\; e \in Feel(a, b) \wedge During(Now, e)\;;$$
$$\exists b \;\; b \in Breezes \; \forall a \;\; a \in Agents \;\Rightarrow$$
$$\exists e \;\; e \in Feel(a, b) \wedge During(Now, e)\;.$$

Pragmatics: (realistic)

- **Indexicals**
- phrases that refer directly to the current situation.
- For example, in the sentence
- “I am in Boston today,” both “I” and “today” are indexicals.
- The word “I” would be represented by the fluent Speaker, and it would be up to the hearer to resolve the meaning of the fluent—that is not considered part of the grammar but rather an issue of pragmatics; of using the context of the current situation to interpret fluent.

Pragmatics: (realistic)…

- The speaker's action is considered a **speech act**,
- It is up to the hearer to decipher what type of action it is—
- a question, a statement, a promise, a warning, a command, and so on.
- A command such as "go to 2 2" implicitly refers to the hearer.

- So far, our grammar for S covers only declarative sentences.
- We can easily extend it to cover commands.
- A command can be formed from a VP, where the subject is implicitly the hearer.
- We need to distinguish commands from statements, so we alter the rules for S to include the type of speech act:
- S(Statement (Speaker , pred (obj))) → NP(obj) VP(pred)
- S(Command(Speaker , pred (Hearer))) → VP(pred) .

Long-distance dependencies:

- Questions introduce a new grammatical complexity.
- In "Who did the agent tell you to give the gold to?" the final word "to" should be parsed as
- [PP to ␣], where the " ␣ " denotes a gap or **trace** where an NP is missing;
- the missing NP is licensed by the first word of the sentence, "who."
- A complex system of augmentations is used to make sure that the missing NPs match up with the licensing words in just the right way, and prohibit gaps in the wrong places.
- For example, you can't have a gap in one branch of an NP conjunction:
- "What did he play [NP Dungeons and ␣]?" is ungrammatical.
- But you can have the same gap in both branches of a VP conjunction:
- "What did you [VP [VP smell ␣] and [VP shoot an arrow at ␣]]?"

Ambiguity:

- Here are some examples taken from newspaper headlines:
- Squad helps dog bite victim.
- Police begin campaign to run down jaywalkers.
- Helicopter powered by human flies.
- Once-sagging cloth diaper industry saved by full dumps.
- Portable toilet bombed; police have nothing to go on.
- Teacher strikes idle kids.
- Include your children when baking cookies.
- Hospitals are sued by 7 foot doctors.
- Milk drinkers are turning to powder.
- Safety experts say school bus passengers should be belted.

- ***almost every utterance (word) is highly ambiguous, even though the alternative interpretations (understanding) might not be apparent to a native speaker.***

Lexical ambiguity,

- A system with a large grammar and lexicon might find thousands of interpretations for a perfectly ordinary sentence
- **Lexical ambiguity**, in which a word has more than one meaning.
- “back” can be
  - an adverb (go back),
  - an adjective (back door),
  - a noun (the back of the room) or
  - a verb (back up your files).
- “Jack” can be
- a name, a noun
  - a playing card,
  - a six-pointed metal game piece,
  - a nautical flag,
  - a fish, a socket, or a device for raising heavy objects), or
- a verb
  - (to jack up a car, to hunt with a light, or to hit a baseball hard).

Syntactic ambiguity

- **Syntactic ambiguity** refers to a phrase that has multiple parses:
- “I smelled a wumpus in 2,2” has two parses:
- one where the prepositional phrase “in 2,2” modifies the noun and one where it modifies the verb.

Semantic Ambiguity

- The syntactic ambiguity leads to a **semantic ambiguity**,
- because one parse means that the wumpus is in 2,2 and the other means that a stench is in 2,2.
- In this case, getting the wrong interpretation could be a deadly mistake for the agent.

A metaphor

- A **metaphor** is another figure of speech, in which a phrase with one literal meaning is used to suggest a different meaning by way of an analogy.
- Thus, metaphor can be seen as a kind of metonymy where the relation is one of similarity.

Disambiguation

- **Disambiguation** is the process of recovering the most probable intended meaning of an word.
- In one sense we already have a framework for solving this problem:
- each rule has a probability associated with it, so the probability of an interpretation is the product of the probabilities of the rules that led to the interpretation.
- Unfortunately, the probabilities reflect how common the phrases are in the corpus from which the grammar was learned, and thus reflect general knowledge, not specific knowledge of the current situation.

- To do disambiguation properly, we need to combine four models:
- 1. The **world model**: the likelihood that a proposition occurs in the world.
- Given what we know about the world,
- it is more likely that a speaker who says
- “I’m dead” means “I am in big trouble” rather than “My life ended, and yet I can still talk.”

- 2. The **mental model**: the likelihood that the speaker forms the intention of communicating a certain fact to the hearer.
- This approach combines models of what the speaker believes, what the speaker believes the hearer believes, and so on.
- For example, when a politician says, “I am not a crook,”
- the world model might assign a probability of only 50% to the proposition that the politician is not a criminal, and 99.999% to the proposition that he is not a hooked shepherd’s staff.
- Nevertheless, we select the former interpretation because it is a more likely thing to say.

- 3. The **language model**: the likelihood that a certain string of words will be chosen, given that the speaker has the intention of communicating a certain fact.
- 4. The **acoustic model**: for spoken communication, the likelihood that a particular sequence of sounds will be generated,
- given that the speaker has chosen a given string of words.

Thank You