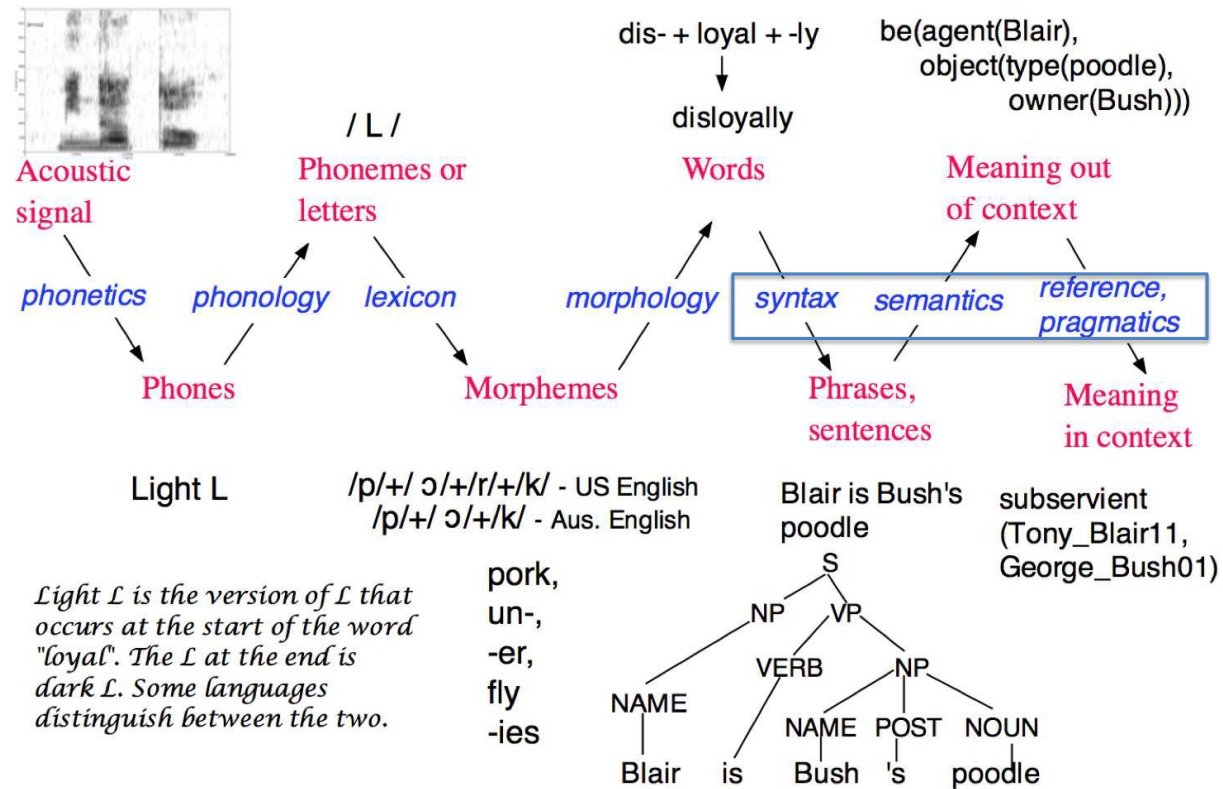


COMP3411: Artificial Intelligence
Natural Language Processing

Linguistics Landscape



Natural Language Processing

- Syntax
 - Linguistic Knowledge
 - Grammars and Parsing
 - Probabilistic Parsing
- Semantics
 - Semantic Interpretation and Logical Form
- Pragmatics
 - Discourse Processing
 - Speech Act Theory
 - (Spoken) Dialogue Systems

NLP Applications

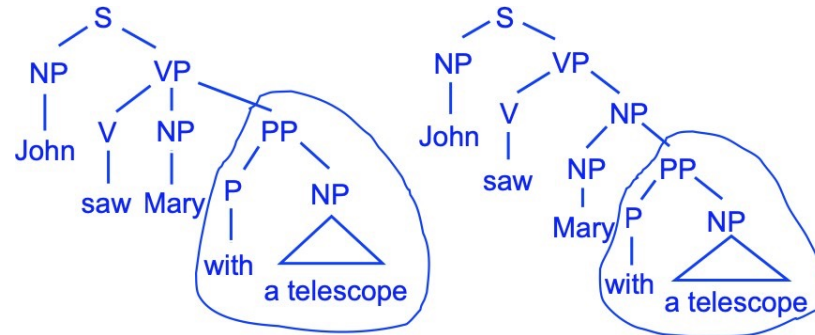
- Chatbots
 - Customer service, e.g. CBA, Amtrak, Lyft, Spotify, Whole Foods
- Personal Assistants
 - Siri, Alexa, Google Assistant
- Information Extraction
 - Financial reports, news articles
- Machine (Assisted) Translation
 - Weather reports, EU contracts, Canada Hansard
- Social Robotics
 - Home care robots

Central Problem – Ambiguity

- Natural languages exhibit **ambiguity**
 - “The fisherman went to the bank” (lexical)
 - “The boy saw a girl with a telescope” (structural) “Every student took an exam” (semantic)
 - “The table won’t fit through the doorway because it is too [wide/narrow]” (pragmatic)
- Ambiguity makes it difficult to interpret meaning of phrases/sentences
 - But also makes inference harder to define and compute
- Resolve ambiguity by mapping to unambiguous representation

Structural Ambiguity

“John saw Mary with a telescope”



- Different interpretation → different representation

“John sold a car to Mary” and “Mary was sold a car by John”

- Same interpretation → same representation

Syntax

Sentence vs Utterance

- Sentence is a group of words that convey some meaning
- An utterance is a group of words between pauses in speech.

Lexical Items (Basic Words)

- Open class (can add new words)
 - Nouns: denote objects (e.g. cat, John, justice)
 - Verbs: denote actions, events (e.g. buy, break, believe)
 - Adjectives: denote properties of objects (e.g. red, large)
 - Adverbs: denote properties of events (e.g. quickly)
- Closed class (function words, mostly fixed in the language)
 - Prepositions: at, in, of, on, . . .
 - Articles: the, a, an
 - Conjunctions: and, or, if, then, than, . . .

Sentence Forms

- Declarative (indicative)
 - Bart is listening.
- Yes/No question (interrogative)
 - Is Bart listening?
- Wh-question (interrogative)
 - When is Bart listening?
- Imperative (command)
 - Listen, Bart!
- Subjunctive (conditional or imaginary situations)
 - If Bart were listening, he might hear something useful.

Noun Phrases

- Examples
 - John, The dog, The big ugly dog, The man in the red car,
The oldest man in the world with a beard, The oldest man who lives in
China, . . .
- Noun phrases: occur as “subject” with a range of “predicates”
 - (noun phrase) ate the bone
 - (noun phrase) saw the bird in the sky
 - (noun phrase) believes that $2 + 2 = 4$
- Sentences need not “make sense”

Verb Phrases

- Examples
 - **ate** the bone
 - **saw** the bird in the sky
 - **believes** that $2 + 2 = 4$
- Verb phrases: occur as “predicate” with a range of “subjects”
 - John (verb phrase)
 - The dog (verb phrase)
 - Any noun phrase (verb phrase)
- Verb phrase depends on noun phrase

Prepositional Phrases

- Prepositions:
 - at, on, with, ...
- Prepositional phrase:
 - Preposition followed by a noun phrase
 - Modifies a noun phrase or a verb phrase
- Examples:
 - She caught the bus *on time*
 - That puppy *at the park* is so happy.
 - Jane cheered for her team *with excitement*.

Context Free Grammars

- Terminal symbols (lexical items)
- Nonterminal symbols (grammatical categories)
- Start symbol (a nonterminal) e.g. (sentence)
- Rewrite rules
 - nonterminal \rightarrow sequence of nonterminals, terminals
 - e.g. (sentence) \rightarrow (noun phrase) (verb phrase)
- Open question: is English context free?

BNF Notation (Backus-Naur Form)

- A BNF grammar specification consists of *production rules*.

$\langle s \rangle ::= a b$

$\langle s \rangle ::= a \langle s \rangle b$

- First rule says that whenever s appears in a string, it can be rewritten with the sequence ab .
- Second rule says that s can be rewritten with a followed by s followed by b .

BNF Notation

- s is a non-terminal symbol.
- a and b are terminal symbols.
- A grammar rule can generate a string, e.g.

$s \rightarrow a s b$

$a s b \rightarrow a a s b b$

$a a s b b \rightarrow a a a b b b$

Definite Clause Grammars

- Prolog DCG notation parses sequences of symbols in a list.

```
s --> [a], [b].
```

```
s --> [a], s, [b].
```

```
?- s([a, a, b, b], X).
```

```
X = []
```

```
?- s([a, c, b], X).
```

```
false.
```

Typical (Small) Grammar

$S \rightarrow NP VP$

$NP \rightarrow [Det] Adj^* N [AP \mid PP \mid Rel\ Clause]^*$

$VP \rightarrow V [NP] [NP] PP^*$

$AP \rightarrow Adj PP$

$PP \rightarrow P NP$

$Det \rightarrow a \mid an \mid the \mid \dots$

$N \rightarrow John \mid Mary \mid park \mid telescope \mid \dots$

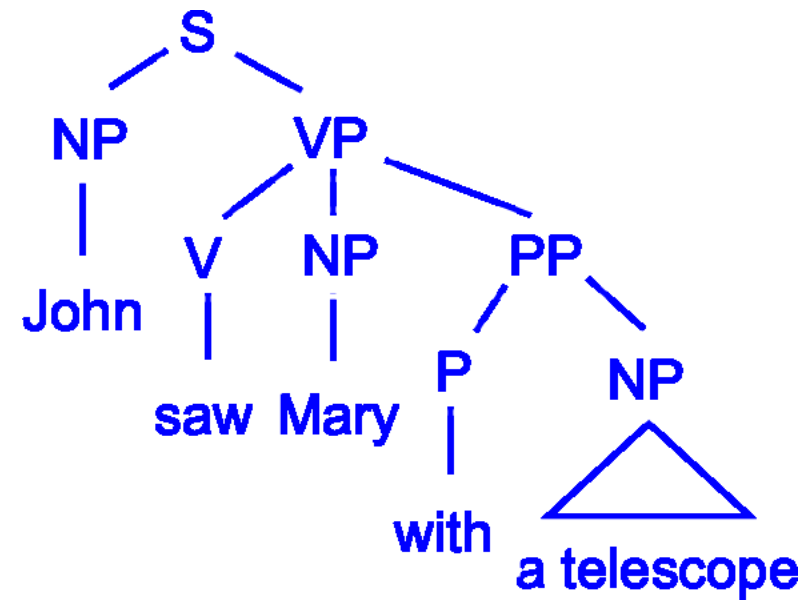
$V \rightarrow saw \mid likes \mid believes \mid \dots$

$Adj \rightarrow hot \mid hotter \mid \dots$

$P \rightarrow in \mid \dots$

Extra notation: $*$ is “0 or more”; $[\dots]$ is “optional”

Syntactic Structure



Syntactically ambiguous = more than one parse tree

(Leftmost) Derivation of Example

S
⇒ NP VP
⇒ N VP
⇒ John VP
⇒ John V NP PP
⇒ John saw NP PP
⇒ John saw N PP
⇒ John saw Mary PP
⇒ John saw Mary P NP
⇒ John saw Mary with NP
⇒ John saw Mary with Det N
⇒ John saw Mary with a N
⇒ John saw Mary with a telescope

S → NP VP
NP → [Det] Adj* N [AP PP Rel Clause]*
VP → V [NP] [NP] PP*
AP → Adj PP
PP → P NP
Det → a an the ...
N → John Mary park telescope ...
V → saw likes believes ...
Adj → hot hotter ...
P → in ...

⇒ means “rewrites as”

Rightmost Derivation

S
⇒ NP VP
⇒ NP V NP PP
⇒ NP V NP P NP
⇒ NP V NP P Det N
⇒ NP V NP P Det telescope
⇒ NP V NP P a telescope
⇒ ...
⇒ ...
⇒ ...

S → NP VP
NP → [Det] Adj* N [AP PP Rel Clause]*
VP → V [NP] [NP] PP*
AP → Adj PP
PP → P NP
Det → a an the ...
N → John Mary park telescope ...
V → saw likes believes ...
Adj → hot hotter ...
P → in ...

Parsing

- Aim is to compute a derivation of a sentence
 - produces parse tree
- Methods
 - Top down
 - Start with S , apply rewrite rules until sentence reached
 - Bottom up
 - Start with sentence, apply rewrite rules “in reverse” until S is reached
 - Chart parsing
 - Chart records parsed fragments and hypotheses
 - Can mix top down and bottom up strategies

Top-Down Parsing

- Use a stack to record working hypothesis
- Start with S as only symbol on stack
- At each step
 - Rewrite top of stack T using grammar rule $T \rightarrow \text{RHS}$
- i.e. replace T by RHS (in reverse order), OR
- Match word on top of stack to next word in sentence
- Apply backtracking on failure
- Accept sentence when stack is empty and all words in sentence matched; reject sentence when no rules to try
- Produces leftmost derivation

Example

STACK	INPUT
S	John saw Mary with a telescope
NP VP	John saw Mary with a telescope
N VP	John saw Mary with a telescope
John VP	John saw Mary with a telescope
VP	saw Mary with a telescope
V NP PP	saw Mary with a telescope
Saw NP PP	saw Mary with a telescope
NP PP	Mary with a telescope
...	...
...	...

Bottom Up Parsing

- Use a stack to record parsed (left-right) fragment
- Start with stack empty
- At each step
 - Rewrite sequence at top of stack using rule $T \rightarrow \text{RHS}$ i.e. replace RHS (in reverse) by T, OR
 - Move word from input to stack
- Apply backtracking on failure
- Accept sentence when input empty and stack contains S; reject sentence when no more rules to try
- Produces rightmost derivation (in reverse)

Example

STACK	INPUT
	John saw Mary with a telescope
John	saw Mary with a telescope
N	saw Mary with a telescope
NP	saw Mary with a telescope
NP saw	Mary with a telescope
NP V	Mary with a telescope
NP V Mary	with a telescope
NP V N	with a telescope
.
.

Chart Parsing

- Top-down and bottom-up parsers may have to repeat parsing because they backtrack
- E.g. “The old man the boats”:
 - “man” can be a noun or a verb
 - Initially “old man” might be put together as a noun phrase,
 - but then we see that “man” is the verb
- A chart parser maintains a table or graph of all possible parsed fragments to avoid backtracking

“Garden Path” Sentence

Grammar:

1. $S \rightarrow NP VP$
2. $NP \rightarrow ART N$
3. $NP \rightarrow ART ADJ N$
4. $VP \rightarrow V NP$

Lexicon:

the: ART
old: ADJ, N
man: N, V
boat: N

Sentence: 1 The 2 old 3 man 4 the 5 boat 6

Chart Parser Example

Grammar:

1. $S \rightarrow NP VP$
2. $NP \rightarrow ART N$
3. $NP \rightarrow ART ADJ N$
4. $VP \rightarrow V NP$

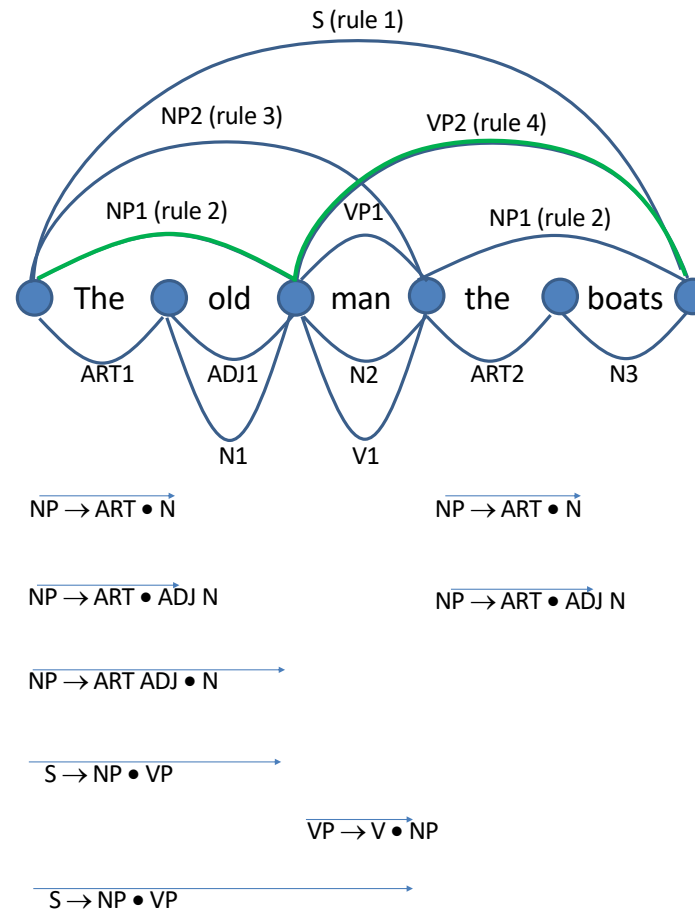
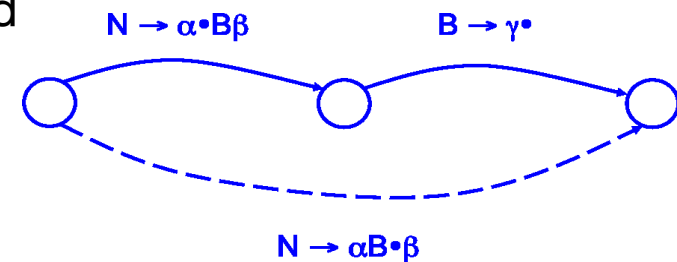
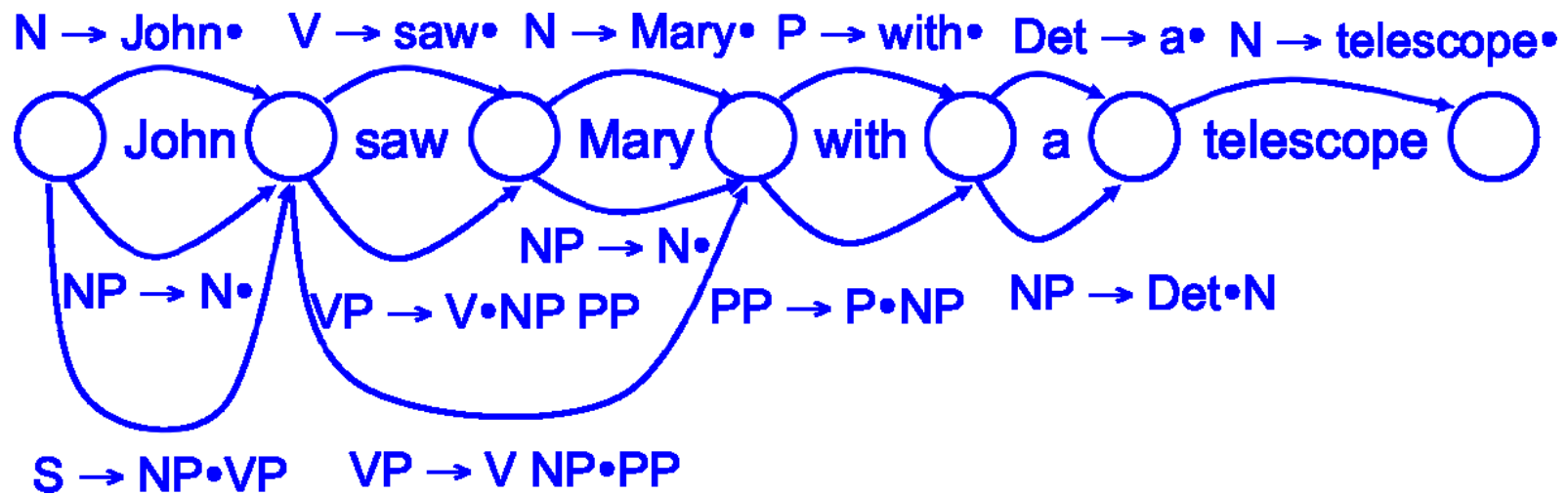


Chart Parsing

- Use a chart to record parsed fragments and hypotheses
- Hypotheses $N \rightarrow \alpha \cdot \beta$ where $N \rightarrow \alpha\beta$ is a grammar rule means “trying to parse N as $\alpha\beta$ and have so far parsed α ”
- One node in chart for each word gap, start and end
- One arc in chart for each hypothesis
- At each step, apply fundamental rule
 - If chart has $N \rightarrow \alpha \cdot B\beta$ from $n1$ to $n2$ and $B \rightarrow \gamma \cdot$ from $n2$ to $n3$
 - add $N \rightarrow \alpha B \cdot \beta$ from $n1$ to $n3$
- Accept sentence when $S \rightarrow \alpha \cdot$ is added from start to end
- Can produce any sort of derivation



Example Chart



Comparing Parsing Methods

- Top Down Parsing
 - Simple, Memory efficient
 - Much repeated work, may loop infinitely
- Bottom Up Parsing
 - Less repeated work, harder to control
- Chart Parsing
 - Memory inefficient (especially with features)
 - No repeated work, difficult to control

Summary

- Syntactic Knowledge
 - Grammatical categories defined by distribution
 - Much determined by properties of lexical items
- Context Free Grammars
 - Useful and powerful formalism
 - Relatively efficient parsers
 - Limited when dealing with complex phenomena
- Parsing
 - Top down method is easy to understand, but not efficient
 - Bottom up method is more efficient

Semantics

Agreement

- Number agreement
 - Which country borders France?
 - Which countries border France?
 - *Which country border France?
 - *Which countries borders France?
- Case
 - I saw him
 - Him saw I
- To capture these facts, need lexical knowledge

Noun Features

Pronoun	Person	Number	Gender	Case
I	first	sing		nom
you	second	sing/plural		nom/acc
we	first	plural		nom
us	first	plural		acc
he	third	sing	masculine	nom
she	third	sing	feminine	nom
it	third	sing	neuter	nom/acc
him	third	sing	masculine	acc
her	third	sing	feminine	acc
they	third	plural		nom
them	third	plural		acc

Nominative: refers to subject
Accusative: refers to object

Verb Forms

Verb	Form	Example
cry	base	
cries	simple present	He cries
cried	simple past	He cried
crying	present participle	He is crying
cried	past participle	He has cried

Tense	Verb sequence	Example
future	will + infinitive	He will cry
present perfect	has + past participle	He has cried
future perfect	will + have + past participle	He will have cried
past perfect	had + past participle	He had cried

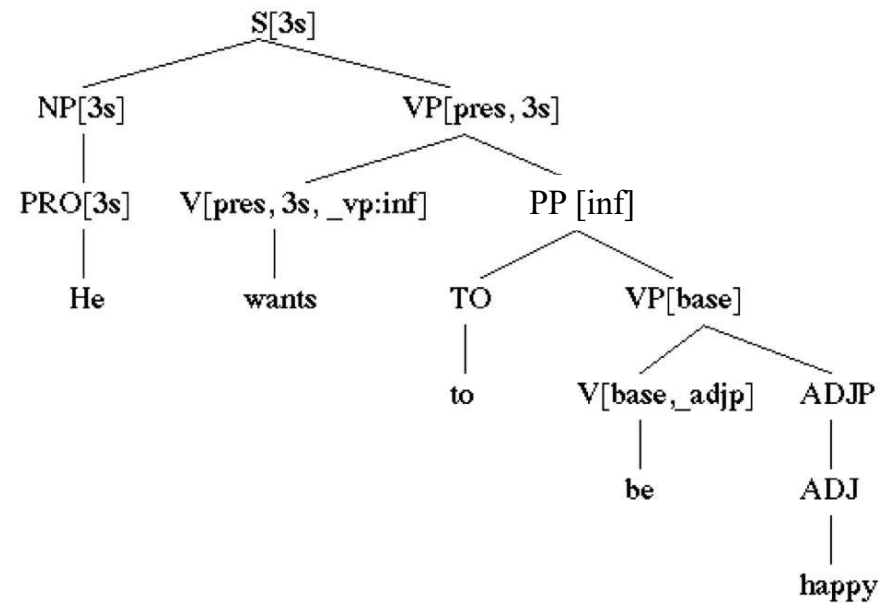
Verb Forms

Progressive	Verb sequence	Example
present	is + present participle	He is crying
past	was + present participle	He was crying
future	will + be + past participle	He will be crying
present perfect	has + been + pres participle	He has been crying
future perfect	will + have + been + past participle	He will have been crying
past perfect	had + been + pres participle	He had been crying

Augmented Context Free Grammars

- Each symbol has a collection of features
- Grammar rules constrain feature values
 - Use unification to enforce constraints, as in Prolog
- Features (mainly) derived from lexical items
- Some also from grammar rules (e.g. Case)
- Simple example
 - $S(\text{number: } N) \rightarrow NP(\text{number: } N) VP(\text{number: } N)$
 - Enforce number agreement by unification (matching)

Example



Note: Not all arguments are specified on tree nodes

Context Dependence

noun --> [cats].

verb --> [hate].

- These grammar rules makes this sentence legal:

the mouse hate the cat.

- Additional constraints must be added to the grammar to ensure that the number of all the parts of speech is consistent.

Context Dependence

```
sentence -->  
    noun_phrase(Number),  
    verb_phrase(Number).
```

```
noun_phrase(Number) -->  
    determiner(Number),  
    noun(Number).
```

```
verb_phrase(Number) -->  
    verb(Number),  
    noun_phrase(_).
```

```
determiner(singular) --> [a].  
determiner(_) --> [the].
```

```
noun(singular) --> [cat].  
noun(singular) --> [mouse].  
noun(plural) --> [mice].
```

```
verb(singular) --> [hates].  
verb(plural) --> [hate].
```

Semantic Interpretation

- Logical form (LF) captures underlying “meaning”
 - Depends on purpose – no one “true” meaning
- Logical form should resolve semantic ambiguity
- Compute LF of sentence from LF of constituents
- Treat logical form as another feature
- Example: John sold a car to Mary

event(e, Sell) \wedge occur(e, past) \wedge agent(e, John) \wedge co-agent(e, Mary) \wedge object(e, {c: car})

Defining the meaning of a sentence

John likes Annie → `like(john, annie)`

transitive verb
(has subject and
object)

Annie paints → `paints(annie)`

intransitive verb
(has subject only)

Defining the meaning of a sentence

```
sentence(VP) -->  
    noun_phrase(Actor),  
    verb_phrase(Actor, VP).
```

```
noun_phrase(NP) -->  
    proper_noun(NP).
```

```
verb_phrase(Actor, VP) -->  
    intrans_verb(Actor, VP).  
verb_phrase(Subject, VP) -->  
    trans_verb(Subject, Object, VP),  
    noun_phrase(Object).
```

```
intrans_verb(Actor, paints(Actor)) --> [paints].  
trans_verb(Subject, Object, likes(Subject, Object)) --> [likes].
```

```
proper_noun(john) --> [john].  
proper_noun(annie) --> [annie].
```

The Determiner 'a'

- 'A person paints' does *not* mean *paints(person)*.
- In this sentence 'person' is not a specific person.

The correct meaning should be:

`exists(X, person(X) & paints(X))`

- The general form for dealing with 'a'

`exists(X, person(X) & Assertion)`

The determiner 'every'

E.g.

Every student studies

$\text{all}(X, \text{student}(X) \rightarrow \text{studies}(X))$

'every' indicates the presence of a *universally* quantified variable.

Relative Clauses

E.g.

Every person that paints admires Monet

Can be expressed in a logical form as:

For all X, if X is a person and X paints then X admires Monet.

in Prolog:

`all(X, person(X) & paints(X) -> admires(X, monet))`

in general:

`all(X, Property1 & Property2 -> Assertion)`

The Complete Grammar

```
?- op(700, xfy, &).  
?- op(800, xfy, ->).
```

```
determiner(X, Property, Assertion, all(X, (Property -> Assertion))) --> [every].  
determiner(X, Property, Assertion, exists(X, (Property & Assertion))) --> [a].
```

```
noun(X, man(X)) --> [man].  
noun(X, woman(X)) --> [woman].  
noun(X, person(X)) --> [person].
```

```
proper_noun(john) --> [john].  
proper_noun(annie) --> [annie].  
proper_noun(monet) --> [monet].
```

```
trans_verb(X, Y, likes(X, Y)) --> [likes].  
trans_verb(X, Y, admires(X, Y)) --> [admires].
```

```
intrans_verb(X, paints(X)) --> [paints].
```

The Complete Grammar

```
sentence(S) -->
    noun_phrase(X, Assertion, S),
    verb_phrase(X, Assertion).

noun_phrase(X, Assertion, S) -->
    determiner(X, Property12, Assertion, S),
    noun(X, Property1),
    rel_clause(X, Property1, Property12).
noun_phrase(X, Assertion, Assertion) -->
    proper_noun(X).

verb_phrase(X, Assertion) -->
    trans_verb(X, Y, Assertion1),
    noun_phrase(Y, Assertion1, Assertion).
verb_phrase(X, Assertion) -->
    intrans_verb(X, Assertion).

rel_clause(X, Property1, (Property1 & Property2)) -->
    [that],
    verb_phrase(X, Property2).
rel_clause(_, Property, Property).
```

The Complete Grammar

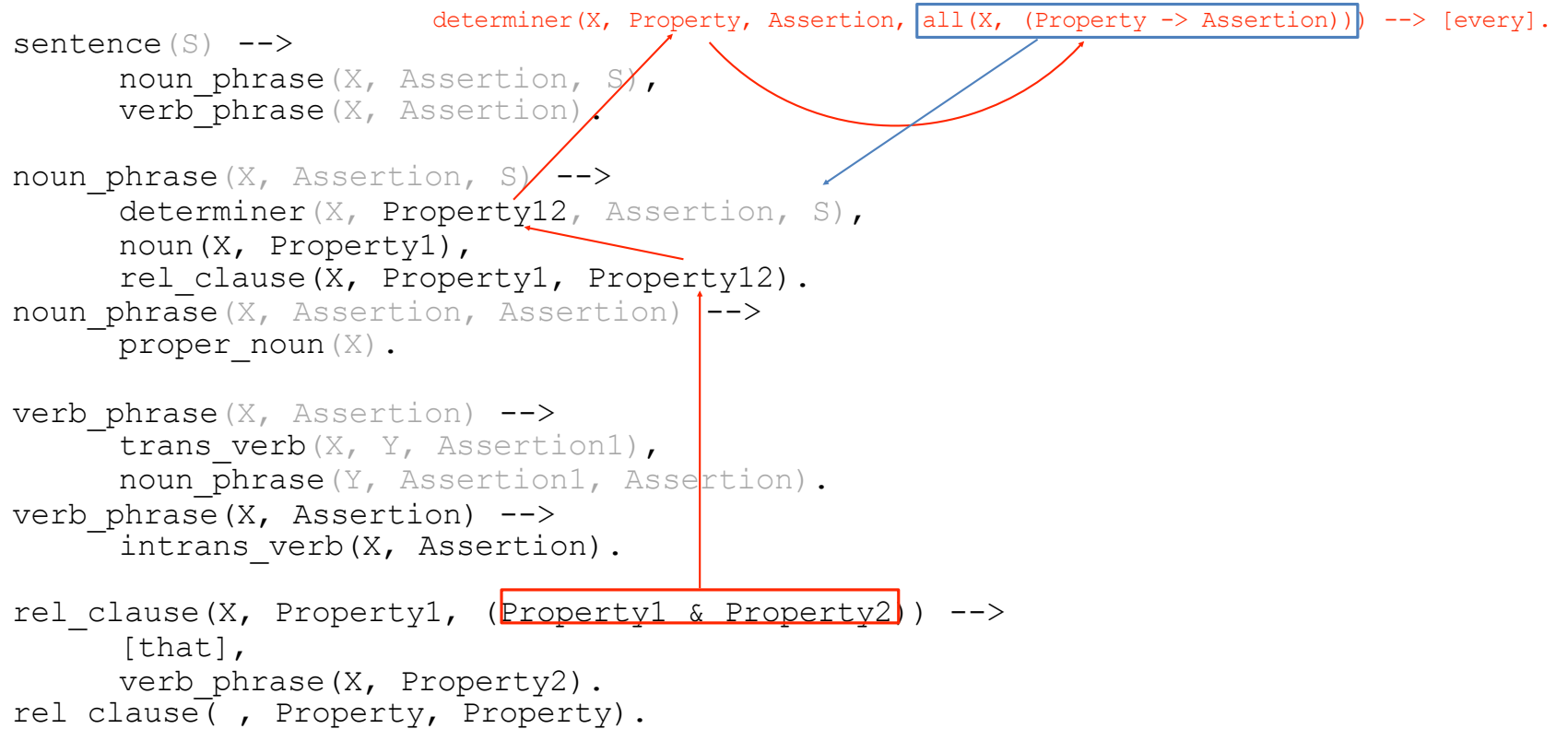
```

sentence(S) -->
    determiner(X, Property, Assertion, all(X, (Property -> Assertion))) --> [every].
    noun_phrase(X, Assertion, S),
    verb_phrase(X, Assertion).

noun_phrase(X, Assertion, S) -->
    determiner(X, Property12, Assertion, S),
    noun(X, Property1),
    rel_clause(X, Property1, Property12).
noun_phrase(X, Assertion, Assertion) -->
    proper_noun(X).

verb_phrase(X, Assertion) -->
    trans_verb(X, Y, Assertion1),
    noun_phrase(Y, Assertion1, Assertion).
verb_phrase(X, Assertion) -->
    intrans_verb(X, Assertion).

rel_clause(X, Property1, (Property1 & Property2)) -->
    [that],
    verb_phrase(X, Property2).
rel_clause(_, Property, Property).
```



The Complete Grammar

```
sentence(S) -->
    noun_phrase(X, Assertion, S),
    verb_phrase(X, Assertion).

noun_phrase(X, Assertion, S) -->
    determiner(X, Property12, Assertion, S),
    noun(X, Property1),
    rel_clause(X, Property1, Property12).
noun_phrase(X, Assertion, Assertion) -->
    proper_noun(X).

verb_phrase(X, Assertion) -->
    trans_verb(X, Y, Assertion1),
    noun_phrase(Y, Assertion1, Assertion).
verb_phrase(X, Assertion) -->
    intrans_verb(X, Assertion).

rel_clause(X, Property1, (Property1 & Property2)) -->
    [that],
    verb_phrase(X, Property2).
rel_clause(_, Property, Property).
```

Result

```
?- sentence(X, [every, person, that, paints, admires, monet], _).
```

```
X = all(_24512, person(_24512) & paints(_24512) -> admires(_24512, monet))
```

Summary

- Disambiguation is central problem in NLP
- Use logical form language to resolve semantic ambiguity
- Augmented grammars can capture agreement and logical form
 - Focus on lexical knowledge
- No one “right” logical form language
 - Case frames, First-order logic, . . .

Pragmatics

Reference Resolution

Jack lost his wallet in his car.
He looked for **it** for several hours.

Jack forgot his wallet.
Sam did too.

Jack forgot his wallet.
He looked for someone to borrow money from.
Sam did too.

I found a red pen and a pencil.
The pen didn't work.

I saw two bears.
Bill saw **some** too.

Reference Resolution

We bought a desk.

The **drawer** was broken.

Reserve a flight to Brisbane for me.

Reserve **one** for Norman too.

Mary gave John five dollars.

It was more than she gave Sue.

One of them was counterfeit.

Each person took a handout.

Then **she** threw **it** away.

John didn't marry a Swedish blonde.

She was Danish/**She** had brown hair/**She's** left with him.

Discourse Entities

- The possible antecedents of pronouns
 - Noun phrases explicitly mentioned in recent discourse
 - A set of (implied) discourse entities
 - e.g. the handouts each person took, the set of people
 - An object related to (evoked by) a discourse entity
 - e.g. the drawer of the desk
 - Fillers of roles in stereotypical scenarios
 - e.g. waiters in restaurants
- Assume discourse is divided into “local contexts”

Focus Hypothesis

- At any time, there is a discourse entity that is the preferred antecedent for pronouns in the current local context – the discourse **focus**
 1. If any object in the local context is referred to by a pronoun in the current sentence, then the focus of the current sentence must also be referred to by a pronoun
 2. The focus is the most preferred discourse entity in the local context that is referred to by a pronoun
 3. Maintaining the focus is preferred to changing the focus
- Order possible antecedents subject > object > rest

Example

Jack left for the party late. (focus = Jack)

When **he** arrived, Sam met **him** at the door. (focus = **he**/Jack)

He decided to leave early. (focus = **he**/Jack)

Jack saw **him** in the park. (focus = **him**)

He was riding a bike. (focus = **he**/**him**)

While Jack was walking in the park, **he** met Sam. (focus = **he**/Jack)

He invited him to the party. (focus = Jack or Sam)

Discourse Structure

- E: So you have the engine assembly finished.
 Now attach the rope to the top of the engine.
 By the way, did you buy petrol today?
- A: Yes. I got some when I bought the new lawnmower wheel.
 I forgot to take my can with me, so I bought a new one.
- E: Did **it** cost much?
- A: No, and I could use another anyway.
- E: OK. Have you got **it** attached yet?

Tracking focus isn't enough

Hierarchical Structure

SEG1

Jack and Sue went to buy a new lawnmower since their old one was stolen.

SEG2

Sue had seen the man who took it and she had chased him down the street, but he'd driven away in a truck.

After looking in the store, they realised they couldn't afford one.

SEG3

By the way, Jack lost his job last month so he's been short of cash recently. He has been looking for a new **one**, but so far hasn't had any luck.

Anyway, they finally found a used **one** at a garage sale.

Discourse Segments

- Recency-based technique for reference resolution
- Fixed time and location or simple progression
- Fixed set of speakers/hearers
- Fixed set of background assumptions
- Intentional view
 - Segment elements contribute to same discourse purpose
- Informational view
 - Segment elements are related temporally, causally, etc.

Attention Stack

- Stack corresponding to segment hierarchy at point in time
 - e.g. [SEG1, SEG2] or [SEG1, SEG3]
- Stack update on starting SEG3
 - Either push new segment
 - giving [SEG1, SEG2, SEG3]
 - Or close current segment and push new segment
 - giving [SEG1, SEG3]

Managing the Attention Stack

- Extending (staying in) a segment
 - All references can be resolved in current segment
 - Same tense or same tense without perfect aspect
- Creating (starting) a new segment
 - Change in tense (progression of discourse)
 - Cue phrase indicating digression
- Closing (finishing) a segment
 - Discourse purpose of new segment part of immediate parent

Speech Acts

- Speech as goal-directed rational activity
 - e.g. promise, threaten, warn, order, advise, state request, inform, assert, deny, apologise, thank, greet, criticise, dare, hope, congratulate, welcome, bless, curse, toast, challenge, announce, declare, question, . . .
- Sometimes explicit in utterances, use of so-called performative verbs
- Speech act affects hearer's actions thoughts, beliefs, etc.
 - Communication involves intention recognition

Indirect Speech Acts

- Use of one kind of speech act to perform another
 - Can you pass the salt?
 - Do you know the time?
 - You are standing on my foot
 - Why don't you leave now?
 - Would you like a game of tennis?
 - If I were you, I'd sell that car
 - Now would you mind getting off my foot!
- Even harder to recognise speaker's intention

Spoken Dialogue Systems

- Feasible now with good speech recognition
 - Speaker dependent or domain specific
- Based on limiting possible dialogue structure
 - Frames with slots that need filling
 - Graphs representing possible transitions
 - Rules for defining actions based on prior context
 - Limited range of sub-dialogues (e.g. clarification)
- Aim is to perform as little reasoning as possible

Initiative

- System Initiative
 - System “controls” dialogue by prompting user for information
 - Useful for specific tasks
 - Booking flights, Ordering pizza, Placing bets
- User Initiative
 - User “controls” dialogue by questions, commands
 - Useful for simple tasks, e.g. web search, training and simulation
- Mixed Initiative
 - Mixture of system initiative and user initiative

Smart Personal Assistant

- Integrated collection of personal (task) assistants
- Each assistant specialises in a task domain
 - E-mail and calendar management
- Users interact through a range of devices
 - PDAs, desktops, iPhone?
- Focus on usability
 - Multi-modal natural language dialogue
 - Adapt to users device, context, preferences

Language Models

Bigrams and Trigrams

- Bigram:
 - Co-occurrence of word pair
 - E.g. the dog, the man, a bike, computer science, electrical engineering
- Trigram:
 - Co-occurrence of word tripple:
 - E.g. the big dog, the old main, a red bike

Predicting the Next Words

- Suppose a sentence is a sequence of words: $w_1 w_2 \dots w_n$
- The probability of the sentence occurring is:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 \dots w_{i-1})$$

Predicting the Next Words

- Bigram probability of the sentence occurring is:

$$P(I \text{ love dogs}) = P(I) \times P(\text{love} | I) \times P(\text{dogs} | \text{love})$$

- Trigram probability of the sentence occurring is:

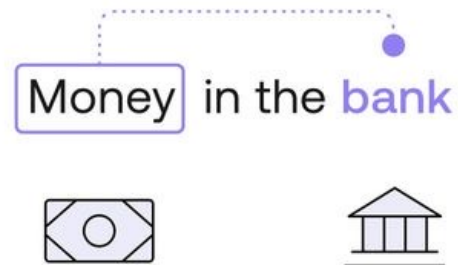
$$P(I \text{ love dogs}) = P(I) \times P(\text{love} | I) \times P(\text{dogs} | I \text{ love})$$

Large Language Models

- Extend to long sequences of words, using ANN to estimate probabilities.
- Problem: word prediction does not necessarily give agreement or pronounce referencing.
- **Attention mechanism** associates words with a **context**

Attention

- **Sentence 1:** The **bank** of the river.
- **Sentence 2:** Money in the **bank**.



Modified sentence 1: The **bank1** of the river.

Modified sentence 2: Money in the **bank2**.

Easy LLM Introductions

- <https://www.analyticsvidhya.com/blog/2022/01/building-language-models-in-nlp/>
- <https://txt.cohere.ai/what-is-attention-in-language-models/>
- [Large Language Models from scratch](#)

Why study syntax, semantics and pragmatics if we have LLMs?

- LLMs are far from prefect.
- Knowledge of computational linguistics helps constrain probabilistic models to make fewer mistakes.

ChatGPT's own assessment of limitations:

- Lack of real-world understanding:
 - can generate coherent and grammatically correct text based on the input it receives
 - does not have a real-world experience and cannot contextualize its responses based on practical experience.
- Biases in the training data:
 - data used to train ChatGPT may contain biases that can be reflected in the generated text
 - biases can be in the form of gender, race, culture, and other factors
- Inability to fact-check:
 - does not have the ability to fact-check or verify the accuracy of the information it generates.
 - It can generate incorrect information, which could be misleading.
- Limited common sense reasoning:
 - has the ability to generate text based on patterns in the training data
 - lacks the ability to reason using common sense, which can result in nonsensical or illogical responses to some prompts.
- Lack of emotional intelligence:
 - cannot comprehend human emotions and hence lacks emotional intelligence.
 - Its generated text may lack empathy, understanding, or the ability to respond appropriately to emotions.

Summary

- The main components of an NLP system are:
 - Syntax
 - Semantics
 - Pragmatics
- Large Language Models are probabilistic generators