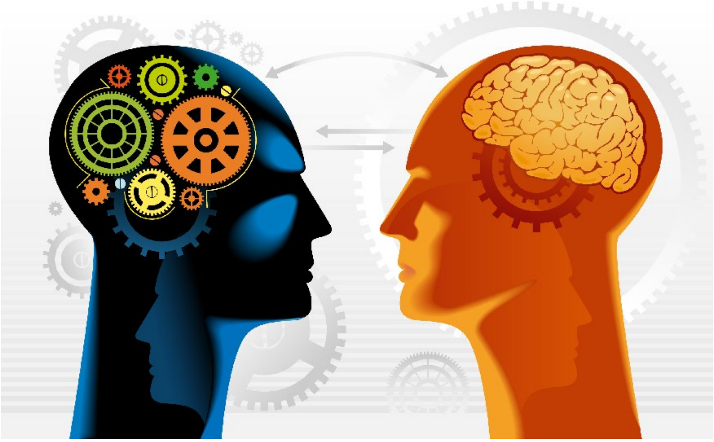
Natural Language Processing
Applied AI – DV2557

Dr. Prashant Goswami
Assistant Professor, BTH (DIDA)
Prashantgos.github.io/

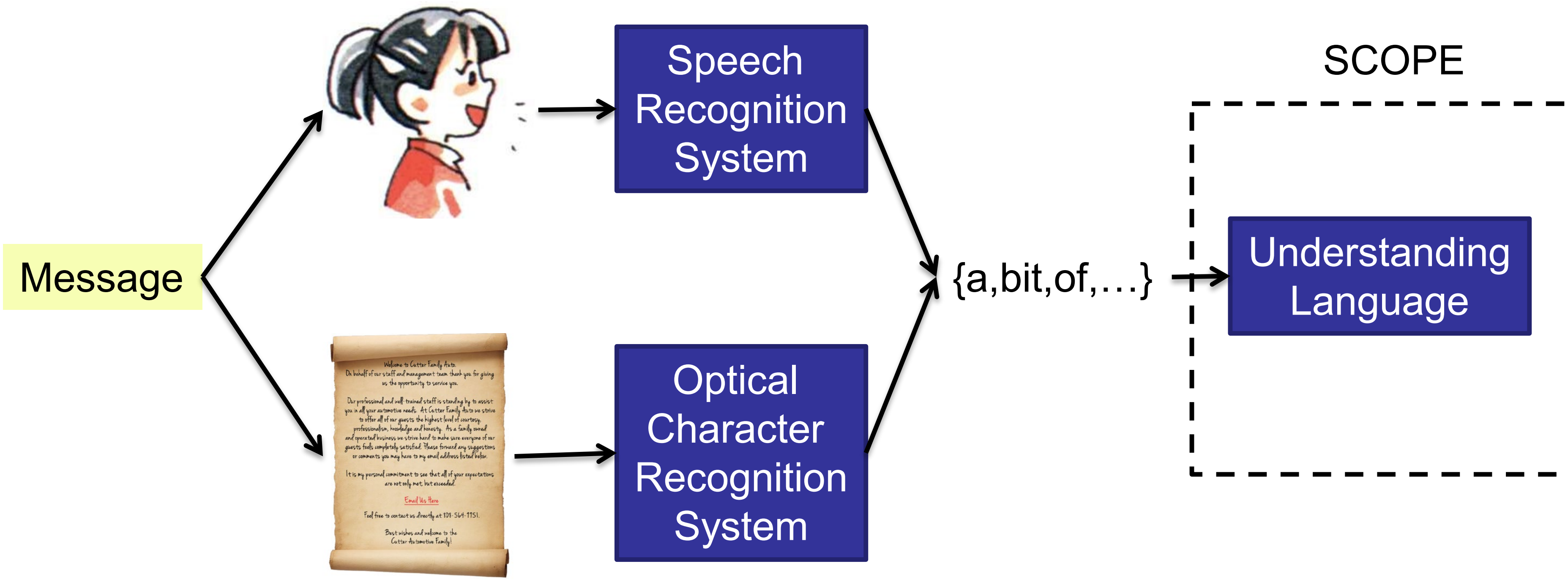
prashant.goswami@bth.se



Communication

- In some cases AI systems need to communicate with humans using natural languages.
- It can be both understanding and producing language…
- … both in text and speech.
- One example is the Jeopardy-playing Watson.

Communication



Fundamentals of language

- A *formal language* is defined as a (possible infinite) set of *strings*.
- Each string is a concatenation of words.
- A word is called a *terminal symbol*.
- Natural language is not as strictly defined as logic, but we will treat it like it is.

Fundamentals of language

- A *grammar* is a finite set of rules that specifies a language.
    - A complete grammar for a natural language is (probably not) possible to create.
    - We can however create one for a subset of a language that covers what we need for a problem.
- *Pragmatics* means the meaning of a string.
    - Situation dependent.
    - ”It is shiny” means different things if you are outside in the sun or in a store looking at jewelry.

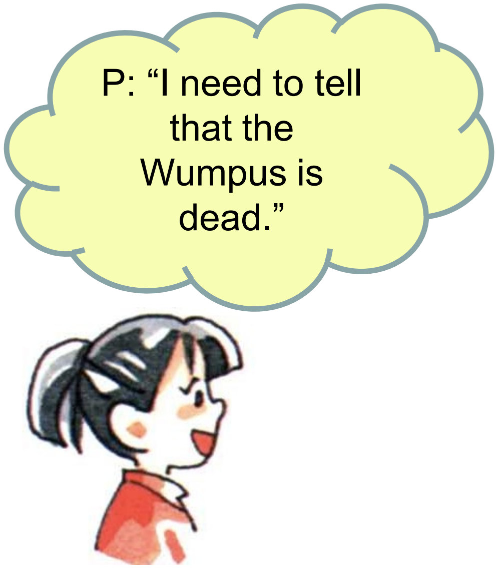
Fundamentals of language

- We assume all languages are based on *phrase structure*.
- It means a string (sentence) is made up of phrases (called subcategories).
- Substring categories:
    - *Noun phrase* (NP): ”The king”, ”The dog”, ”The guy in the corner”
    - *Verb phrase* (VP): ”is dead”, ”is hungry”
- Any NP can be combined with a VP to form a *sentence* (S).

Fundamentals of language

- Category names (NP, VP, S) are called *nonterminal symbols*.
- Nonterminals are defined using *rewrite rules*:
    - S → NP VP
      … is a rewrite rule stating that a sentence S may consist of any NP followed by any VP.

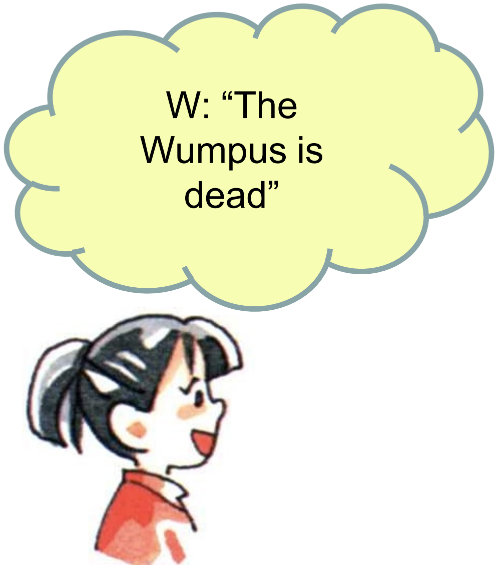
Steps of communication



Intention:

The speaker S decides that there is some proposition P that the hearer H needs to know about.

Steps of communication



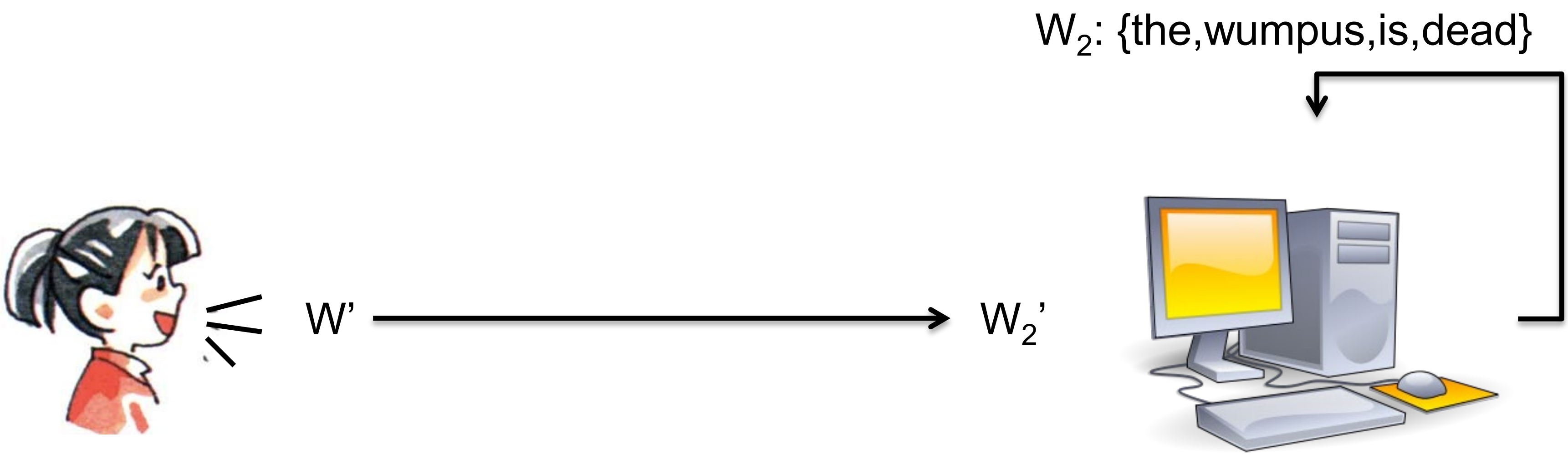
Generation:
The speaker decides how to turn the proposition P into a string W that is likely to be understood by the hearer H.

Steps of communication

W’: [thaxwahmpaxsihzdehd]

<u>Synthesis:</u>
The speaker produces a physical realization W’ of the words W. It can be in ink on a paper, vibrations in the air, ...

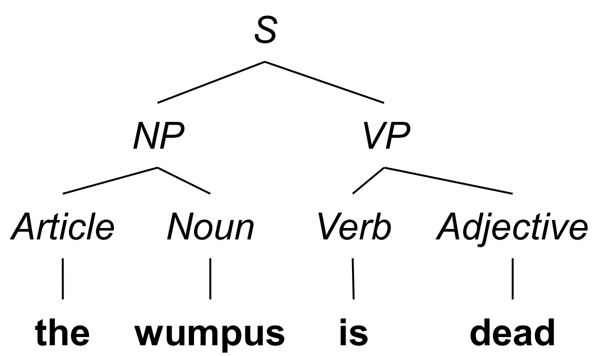
Steps of communication

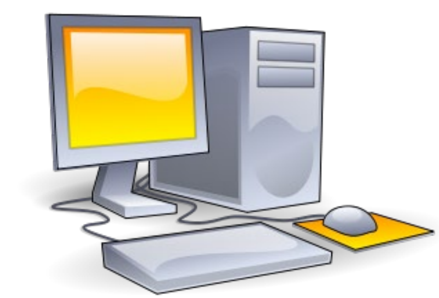


Recognition:
The hearer H receives the physical representation as W_2' and decodes it into the word list W_2.

Steps of communication

W_2: {the,wumpus,is,dead}





Syntactic Analysis (parsing):

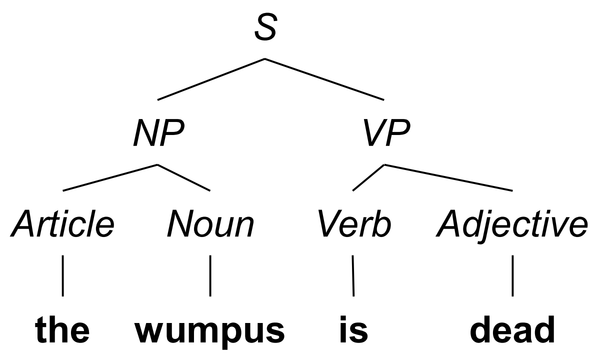
The word list W_2 is turned into a parse tree following the *rewrite rules* for the language:

S → NP VP

NP → Article Noun

VP → Verb Adjective

Steps of communication



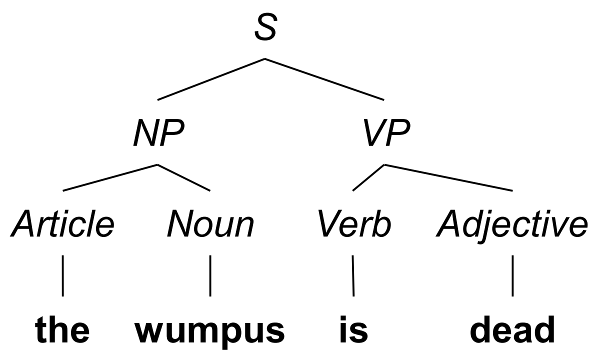
¬Alive(Wumpus,Now)
Tired(Wumpus,Now)

Semantic interpretation:
Understanding the meaning of the sentence, without considering the situation.

Feeling dead is sometimes used to express feeling very tired.

This is called *disambiguation* - the same sentence can have different meanings.

Steps of communication



$\neg Alive(Wumpus_1, S_3)$

~~Tired(Wumpus,Now)~~

<u>Pragmatic interpretation:</u>

Understanding the meaning of the sentence in the current situation. In Wumpus World, the Wumpus can be either dead or alive, not tired…

The general terminals Wumpus and Now are turned into specific terminals matching the game.

Steps of communication

$\neg Alive(Wumpus_1, S_3)$

Incorporation:
The hearer H can choose to believe in the message or not.

Complete vs. Simplified Languages

- Defining terminal symbols (words) and non-terminal symbols (rewrite rules) for real languages like English is nearly impossible.
- But we can however define a fragment of English that covers what we need in our application, for example the Wumpus World.

- The first step is to define a lexicon (a list of allowable words) for our fragment ε_0.

Lexicon for ε_0

Noun	→	stench | breeze | glitter | nothing | player | wumpus | pit | pits | gold | …
Verb	→	is | see | smell | shoot | feel | stinks | go | grab | carry | kill | turn | …
Adjective	→	breezy | dead | back | smelly | stinky | …
Adverb	→	here | there | nearby | ahead | right | left | east | south | west | north | back | …
Pronoun	→	he | she | you | I | it | …
Name	→	Player 1 | Player 2 | …
Article	→	the | a | an | …
Preposition	→	to | in | near | on | …
Conjunction	→	and | or | but | …
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

- Digit is called a *closed class*. We won't add new digits. They are fixed.
- The others are called *open classes*, where we are more likely to add more words (for example the verb jump…)

Grammar

- Step 2 is to define a grammar (rewrite rules) for our fragment ε_0.
- Here we will use five nonterminal symbols:
    - Sentence (*S*)
    - Noun phrase (*NP*)
    - Verb phrase (*VP*)
    - Prepositional phrase (*PP*)
    - Relative clause (*RelClause*)

Grammar for ε_0

S	→	*NP VP*	I + feel a breeze
	|	*S Conjunction S*	I feel a breeze + and + I smell a wumpus
NP	→	*Pronoun*	I
	|	*Name*	Player 1
	|	*Noun*	pits
	|	*Article Noun*	the + wumpus
	|	*Digit Digit*	3 4
	|	*NP PP*	the wumpus + to the east
	|	*NP RelClause*	the wumpus + that is smelly
VP	→	*Verb*	stinks
	|	*VP NP*	feel + a breeze
	|	*VP Adjective*	is + smelly
	|	*VP PP*	turn + to the east
	|	*VP Adverb*	go + ahead
PP	→	*Preposition NP*	to + the east
RelClause	→	**that** *VP*	that + is smelly

Language ε_0

- Our language fragment ε_0 lets us generate useful sentences in good English:
  - Player 1 is in the pit
  - The wumpus that stinks is in 2 2
- Unfortunately it also *overgenerates* – lets us construct grammatically incorrect sentences:
  - I smell pit gold wumpus nothing east
- It also *undergenerates* – rejects grammatically correct sentences:
  - I think the wumpus is smelly

We will get back to this after an example

Syntactic Analysis (Parsing)

- Parsing is the process of finding a parse tree for a sentence.
    - PARSE("the wumpus is dead", ε_0 ,S)
    - … shall return a parse tree with
        - root S
        - leaves "the", "wumpus", "is", "dead"
        - internal nodes are nonterminal symbols from the grammar ε_0
- The parsing process means searching for a parse tree.

Parsing

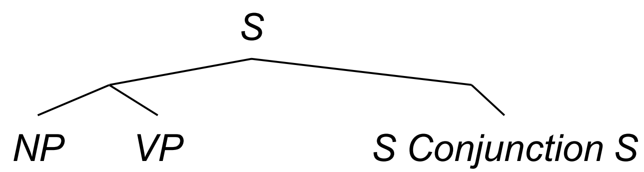
- Two main principles:
  - Top-down parsing
  - Bottom-up parsing

Top-down Parsing

S

Start with the root node S.

Top-down Parsing



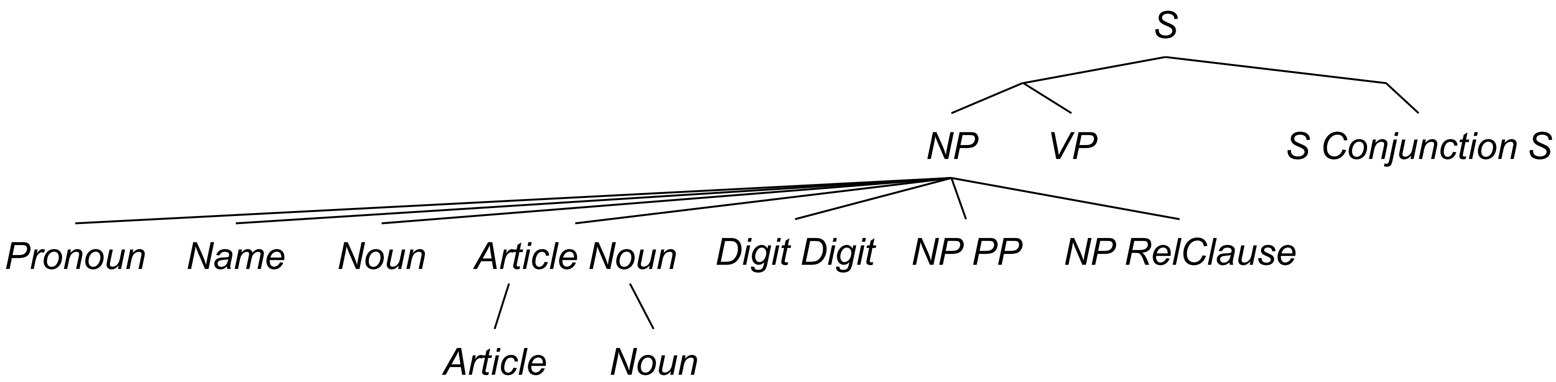
From the grammar, add all rules for S.

S	→	*NP VP*
	|	*S Conjunction S*

Top-down Parsing

From the grammar, add all rules for NP.

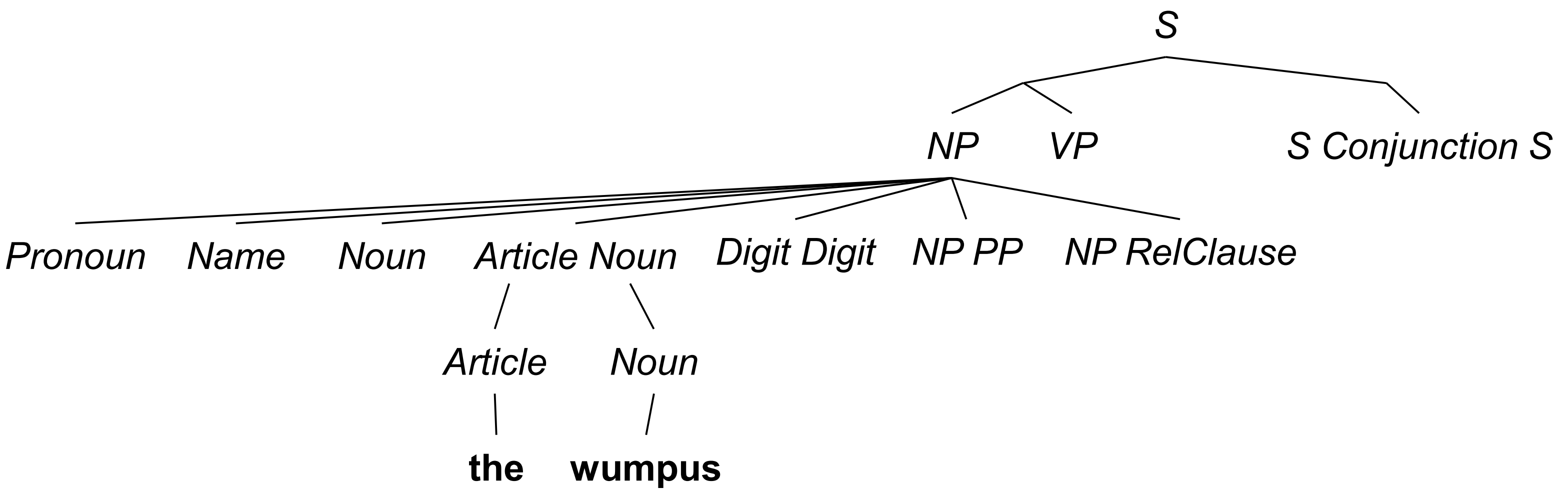
Top-down Parsing



Keep expanding the tree depth-first.

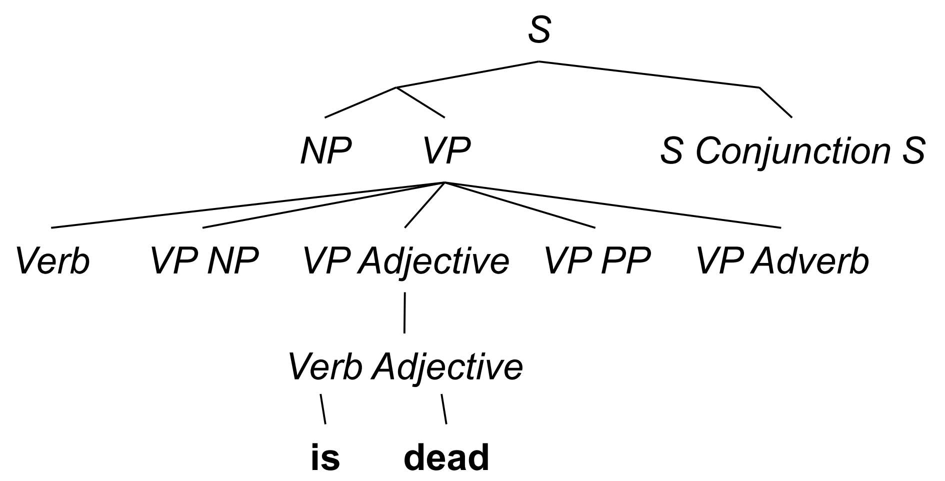
Pronoun: No matching rule or leaf.
Name: No match.
Noun: No match.
Article Noun: Possible to expand.
…

Top-down Parsing



And we have a match!

Top-down Parsing

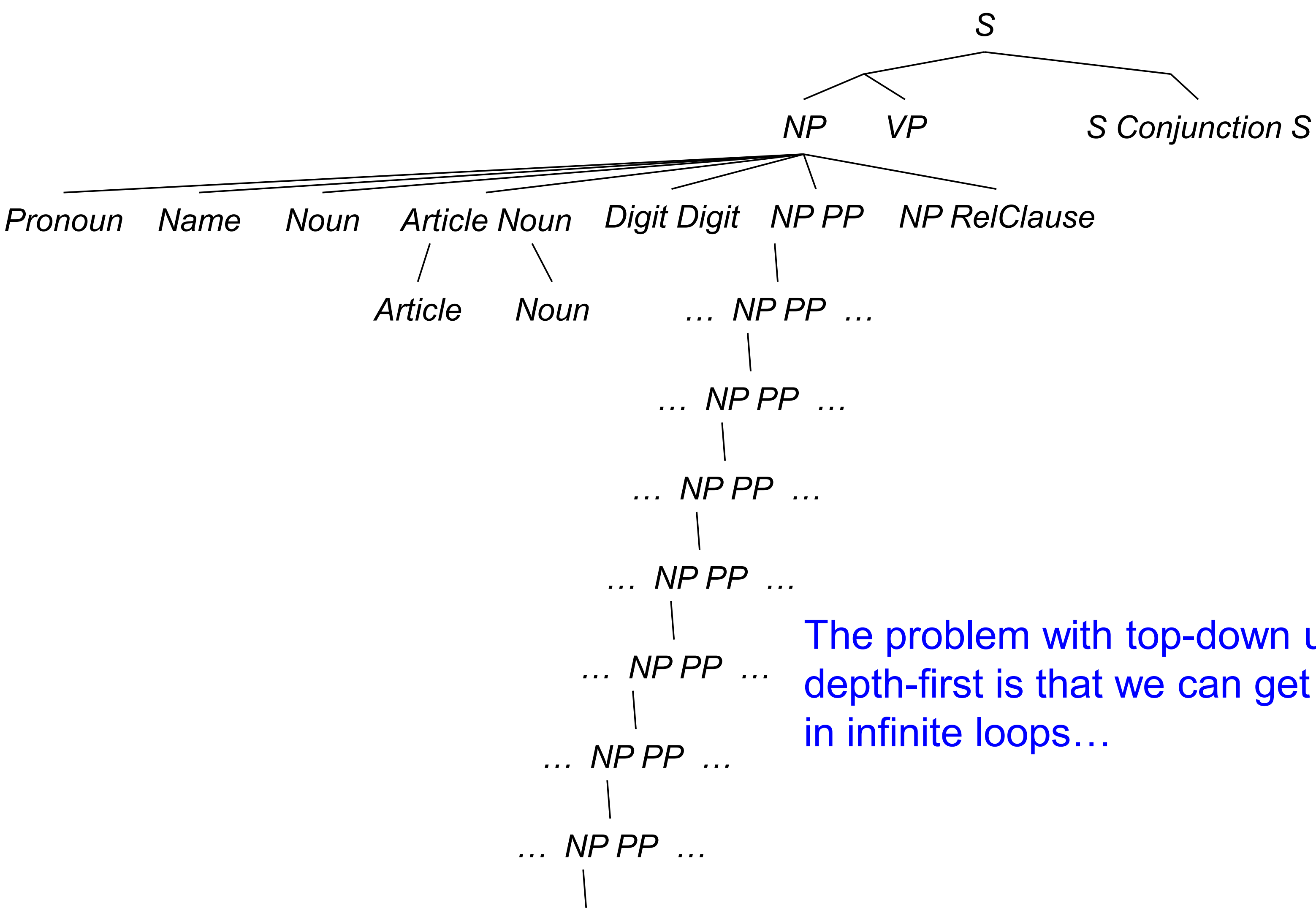


VP	→	*Verb*
	|	*VP NP*
	|	*VP Adjective*
	|	*VP PP*
	|	*VP Adverb*

Do the same for the VP branch, until a match is found or no more expansions can be made.

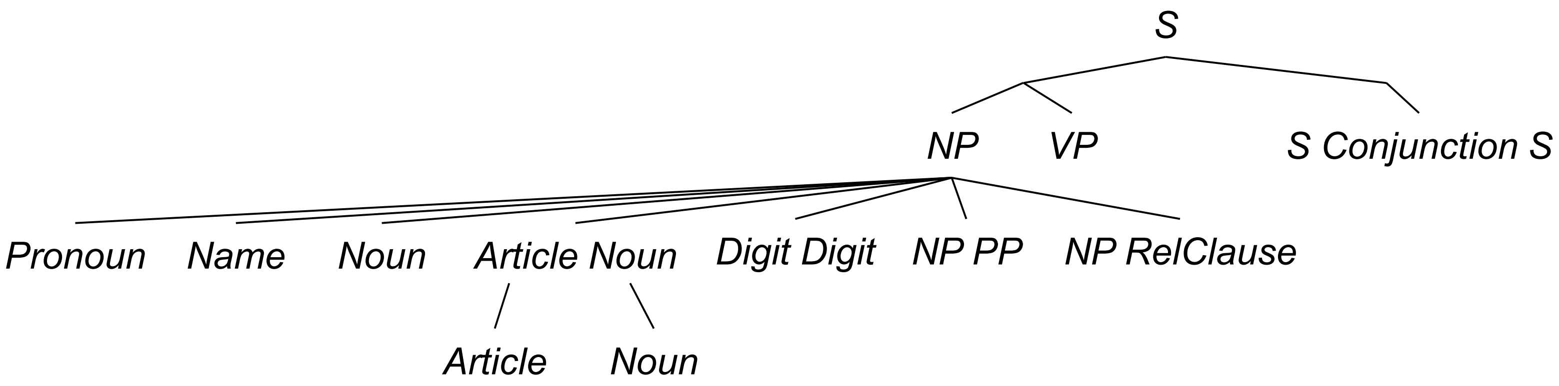
Now we have found a parse tree for our string!

Top-down Parsing



The problem with top-down using depth-first is that we can get stuck in infinite loops…

Top-down Parsing



And if we change to breadth-first, we would get another problem: For invalid sentences the search space will be infinite…

Bottom-up Parsing

Start from the leaf nodes.

the **wumpus** **is** **dead**

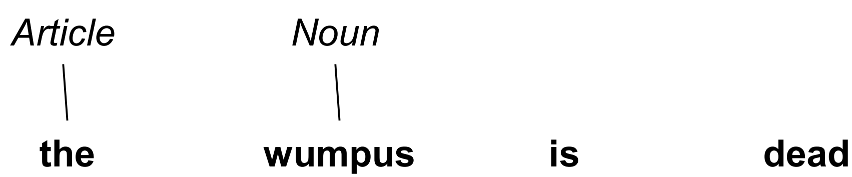
Bottom-up Parsing

Find matches for the leaf nodes from left to right.

Article

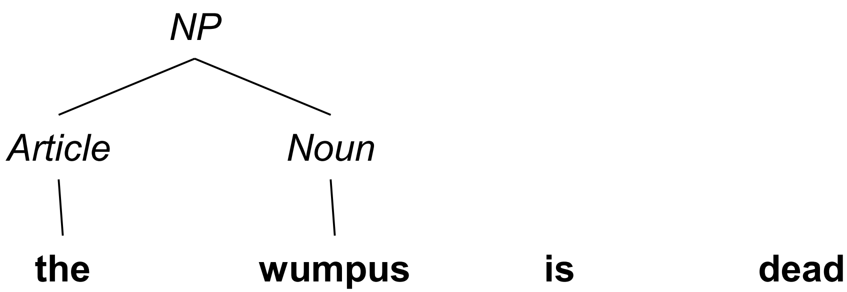
the **wumpus** **is** **dead**

Bottom-up Parsing



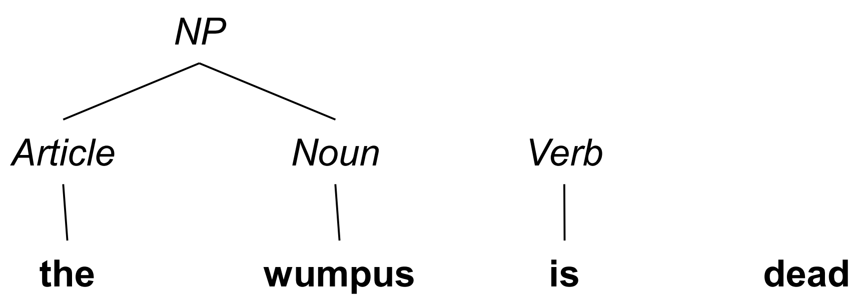
Bottom-up Parsing

Now we have found a match for NP.

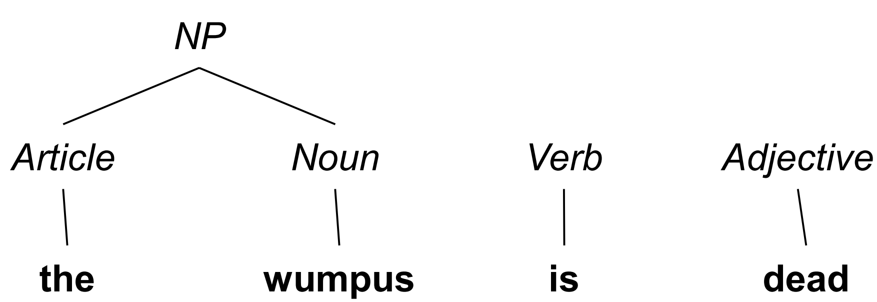


Bottom-up Parsing

Keep working on the rest of the leaf nodes.

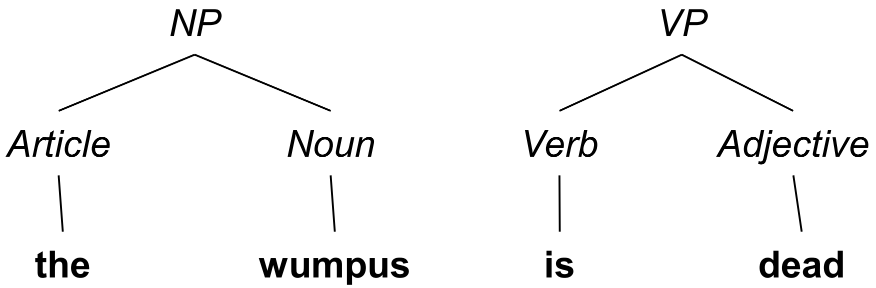


Bottom-up Parsing



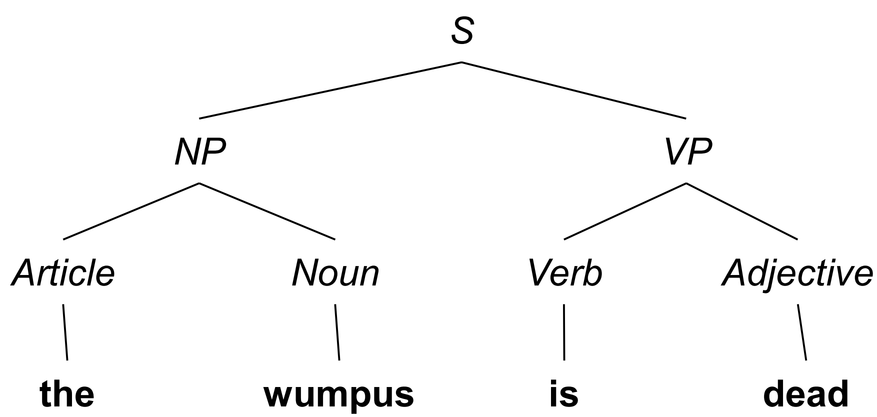
Bottom-up Parsing

Now we have found a match for VP.



Bottom-up Parsing

… and for S. We’re done!



Bottom-up vs. Top-down

- Both bottom-up and top-down can be inefficient, even with good heuristics.
- Top-down often get stuck in infinite loops.
- Bottom-up can generate partial parses that do not fit higher up in the tree, thus spending time searching irrelevant portions of the search space.
  - “the ride the horse gave was wild”
  - … but a VP is not allowed to follow ”the”, so there is no way this parse can fit into S.

VP NP ~~VP~~ → ride the horse ~~gave~~

Chart Parsers

- A more efficient approach is to store already analyzed results, and avoid spending time on re-analysis.
- Once we discover that “the wumpus” is an NP, we store that result in a data structure known as a *chart*.
- Algorithms based on this approach are called *chart parsers*.
- Requires context-free grammar.

Chart Parsers

- Let’s look at an example sentence:
  ”the player feels a breeze”

Chart Parsers

Add 6 vertices (n + 1)
for the 5 word string.

Chart Parsers

Add edges for the words.

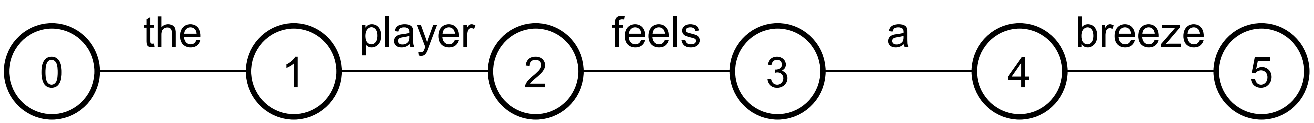


Chart Parsers

0,0, S’ → •S

- Add the start edge stating that
  “if we can find an S, it would complete S’ ”.
- The symbol • separates what we have found so far (left side) and what remains to be found (right side) – We need an S.

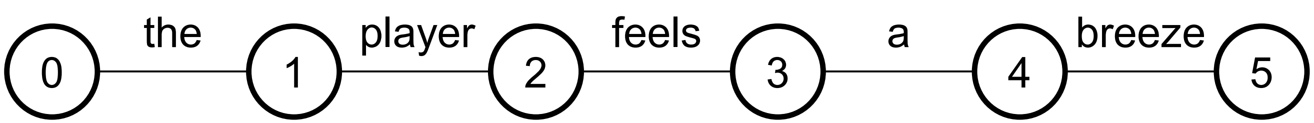


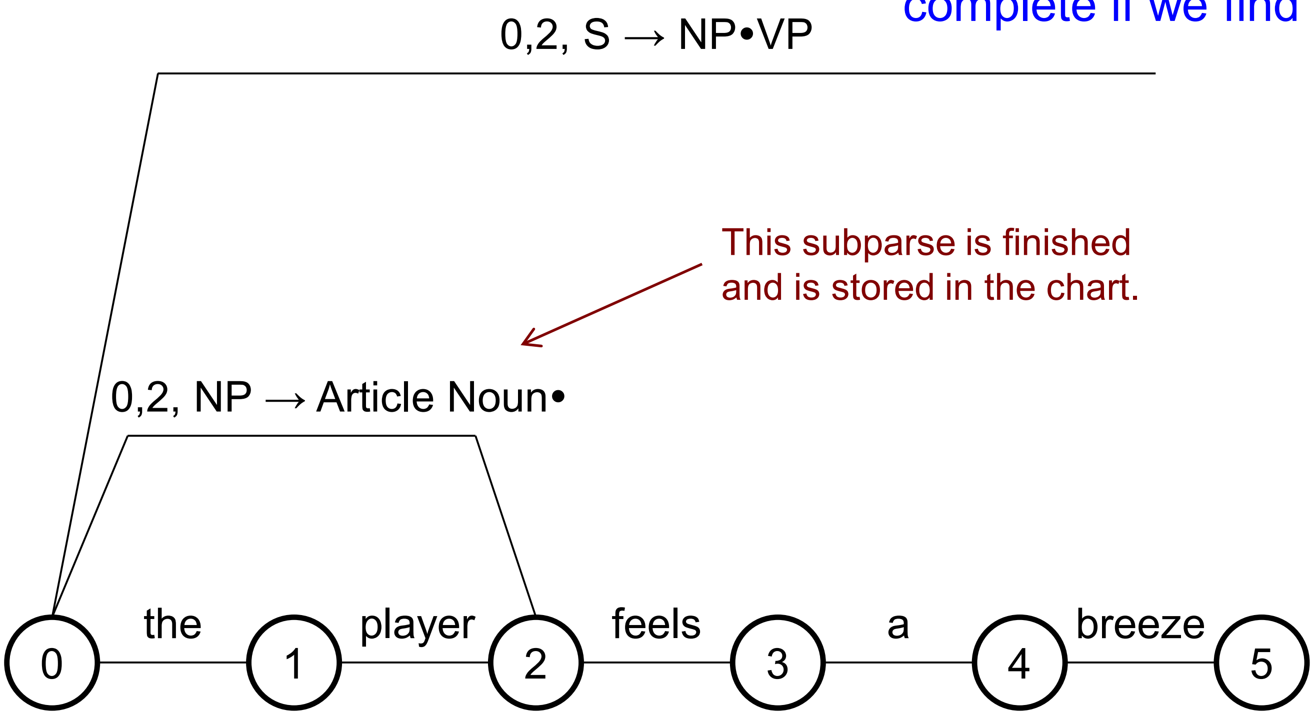
Chart Parsers

0,0, S → •NP VP

From the grammar we can replace the unknown S with an unknown NP and VP.

Chart Parsers

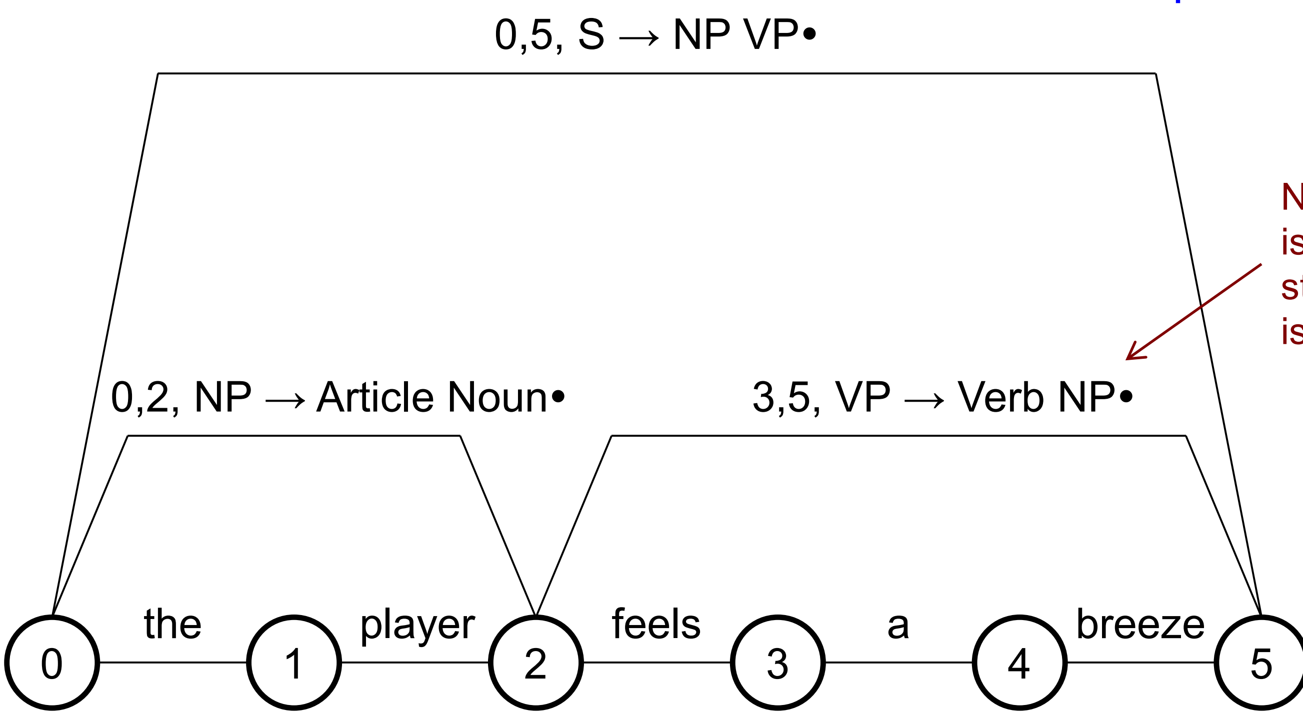
"We have an NP, and S will be complete if we find a VP"



From the rule
NP → Article Noun
we find an NP in [0,2], which satisfies the NP in S.

Chart Parsers

S is now complete. Done!



Note that the NP here is not the same as the stored 0,2,NP since it is in other vertices.

From the rule
VP → Verb NP
we find a VP in [3,5], which satisfies the VP in S.

Chart Parsers

- The pros of chart parsers are:
  - They work both top-down and bottom-up, getting benefits from both approaches.
  - They store finished subparses:
    0,2, NP $\rightarrow$ Article Noun•

Improved Grammar

- We previously noted that the grammar ε_0 overgenerates.
- It can generate “I smell a stench” but also the incorrect “Me smell a stench”.
- The grammatical rules for this states that:
  - “Me” is not a valid NP when it is the subject of a sentence.
- We say that the pronoun “I” is in the *subjective* case and “Me” is in the *objective* case.

Improved Grammar

- To solve this we create a new grammar ε_1 where we split the Pronoun category into two categories:
  - Pronoun_S – Subjective case, “I”
  - Pronoun_O – Objective case, “Me”
- We also need separate subjective and objective nouns:
  - NP_S
  - NP_O

Grammar for ε_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	$\rightarrow$	$Preposition\ NP_O$
$Pronoun_S$	$\rightarrow$	I | you | he | she | it | …
$Pronoun_O$	$\rightarrow$	me | you | him | her | it | …

Note: Only the changes between ε_1 and ε_0 are shown.

Unfortunately…

- ε_1 is better, but still overgenerates…
- English (and many other languages) require an *agreement* between the subject and main verb.
- Consider:
    - I smell - I smells
    - It smell - It smells
    - Third-person vs. other forms.
- Adding new rules for this, and other special cases, would lead to an explosion in number of rules…

Augmented Grammars

- A solution to reduce the number of rules is to use *augmented rules*.
- An *augmented* rule allows parameters on nonterminal categories.
- Example:
  - *NP*(*case*) → *Pronoun*(*case*) | *Name* | *Noun* | …
  - The rule states that NP can be in any case, but if NP is rewritten with a pronoun it must have the same case.

Augmented Grammar for ε_1

S	$\rightarrow$	$NP(Subjective)\ VP \mid \ldots$
$NP(case)$	$\rightarrow$	$Pronoun(case) \mid Name \mid Noun \mid \ldots$
VP	$\rightarrow$	$VP\ NP(Objective) \mid \ldots$
PP	$\rightarrow$	$Preposition\ NP(Objective)$
$Pronoun(Subjective)$	$\rightarrow$	I | you | he | she | it | …
$Pronoun(Objective)$	$\rightarrow$	me | you | him | her | it | …

And to cover subject-verb agreement

S	→	*NP(Subjective, form) VP(form)* | …
NP(case, form)	→	*Pronoun(case, form)* | *Name* | *Noun* | …
VP(form)	→	*VP(form) NP(Objective)* | Verb(form) | …
PP	→	*Preposition NP(Objective)*
Pronoun(Subjective, Third)	→	he | she | it | …
Pronoun(Subjective, Other)		I | you | …
Pronoun(Objective)	→	me | you | him | her | it | …

Unfortunately…

- Even with subject-verb agreement, ε_1 overgenerates.
- A problem is construction of verb phrases:
  - "give me the gold"
  - "go me the gold"
- Both are accepted by ε_1.
- The solution is to state which phrases can follow which verbs in a *subcategorization* (subcat) list.
    - = the category *Verb* is broken down into subcategories.

Subcategorization list

Verb	Subcat	Example phrase
give	[NP, PP]	give the gold to me
	[NP, NP]	give me the gold
smell	[NP]	smell a wumpus
	[Adjective]	smell awful
	[PP]	smell like a wumpus
is	[Adjective]	is smelly
	[PP]	smell awful
	[NP]	is a pit
died	[ ]	died
believe	[S]	believe the wumpus is dead

Subcategorization

- Example: give [NP,PP]
  - ”Give” can be made into a complete VP by adding [NP,PP].
  - ”Give the gold” can be made complete by adding [PP].
  - ”Give the gold to me” is complete, and therefore has an empty subcat list [ ].

Syntactic Analysis

- With our lexicon and grammar ε_0 extended with:
  - Subjective and Objective case
  - Subject-Verb agreement
  - Verb subcategorization
- … we have a pretty good grammar for the Wumpus World.
- These rules form a solid base for syntactic analysis in most AI applications.
- …but are still too simple for complete English.

Next step is understanding a sentence using

SEMANTIC ANALYSIS

Semantic Interpretation

- The next step in language processing is *semantic interpretation* – extracting the meaning of a sentence.
- We need to translate a sentence to First-Order Logic, for example:

"The wumpus is dead"	$Dead(Wumpus_1)$
"Go to 1 2"	$Move(Player_1, 1, 2)$

Example

- We have the sentence ”John loves Mary”
- ”loves” is a verb, referring to two objects:
  $\lambda y\ \lambda x\ Loves(x,y)$
- The rules
  - VP → Verb NP NP → Name
- … tells us that the verb relates to Mary:
  $\lambda x\ Loves(x,Mary)$
- And the rules
  - S → NP VP NP → Name
- … tells us that John must be the object:
  $Loves(John,Mary)$

Note: λ is used when we need a new variable on-the-fly.

Example

- We can create a grammar for the semantic interpretation:

$S(pred(obj))$	$\rightarrow$	$NP(obj)\ VP(pred)$
$VP(pred(obj))$	$\rightarrow$	$Verb(pred)\ NP(obj)$
$NP(obj)$	$\rightarrow$	$Name(obj)$
$Name(John)$	$\rightarrow$	**John**
$Name(Mary)$	$\rightarrow$	**Mary**
$Verb(\lambda y\ \lambda x\ Loves(x,y))$	$\rightarrow$	**loves**

- … which we can use to create a parse tree:

Example

Start with the words (bottom-up).

John **loves** **Mary**

Example

Find an interpretation for the verb and the nouns.

Name(*John*)	*Verb*(λy λx *Loves*(x,y))	*Name*(*Mary*)
John	**loves**	**Mary**

Example

A name is a complete NP
according to the grammar.

NP(*John*)

Name(*John*)

John

Verb(λy λx *Loves*(x,y))

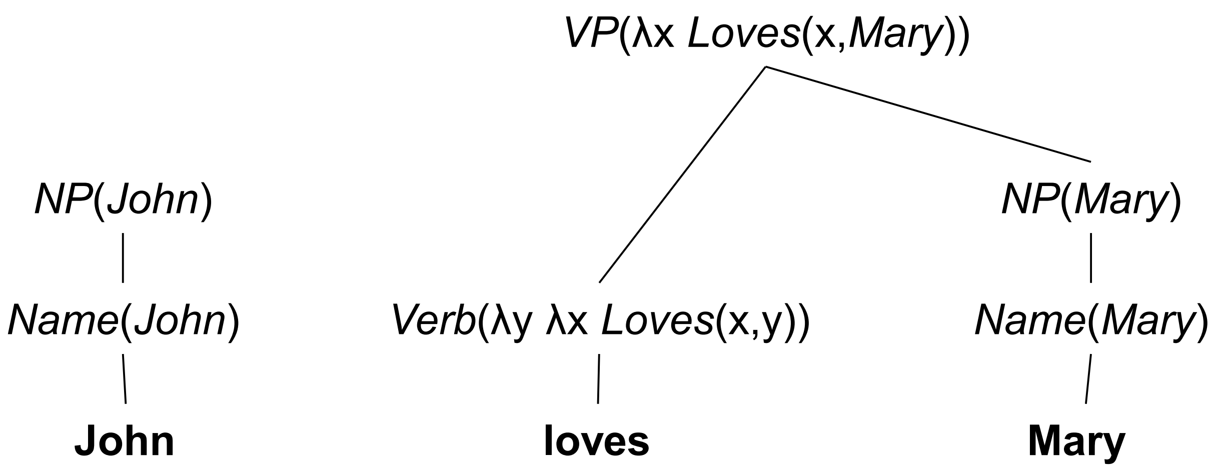
loves

NP(*Mary*)

Name(*Mary*)

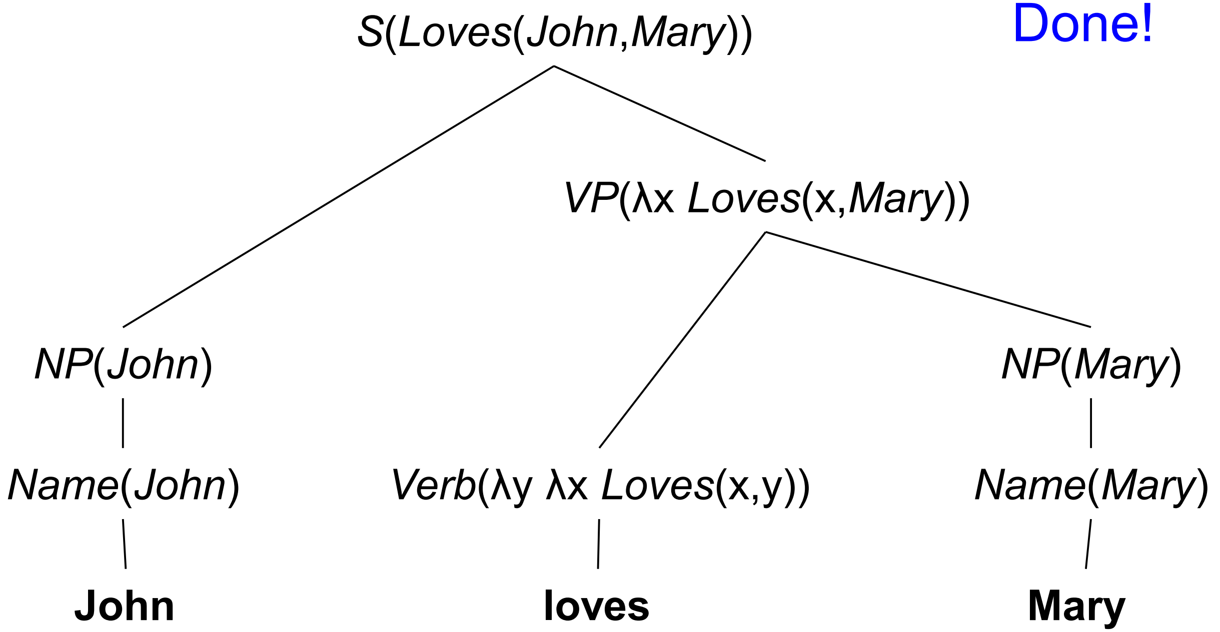
Mary

Example



A VP is made up from a Verb(rel) and an NP(obj). NP(Mary) is interpreted as the NP(obj).

Example



Done!

And NP(John) is interpreted as the NP(obj) in S:
S(pred(obj)) → NP(obj) VP(pred)

Summary

- Now we have a basic set of theories to deal with syntactic and semantic analysis of quite complex subsets of natural language.
- There are far more about natural language processing than mentioned today:
  - Language generation
  - Ambiguity – several possible interpretations of a sentence.
  - Discourse - text with more than one sentence.
  - Grammar induction – learning grammar from data.
  - Probabilistic language models
  - …
- This is out of scope for this course.

That was all for this lecture

Acknowledgements

Dr. Johan Hagelbäck

Linnæus University

johan.hagelback@lnu.se

http://aiguy.org