Artificial Intelligence

Natural Language Processing II

Lecture 11

(10 November, 1999)

Tralvex (Rex) Yeap MAAI MSCS
University of Leeds

Content: Natural Language Processing II

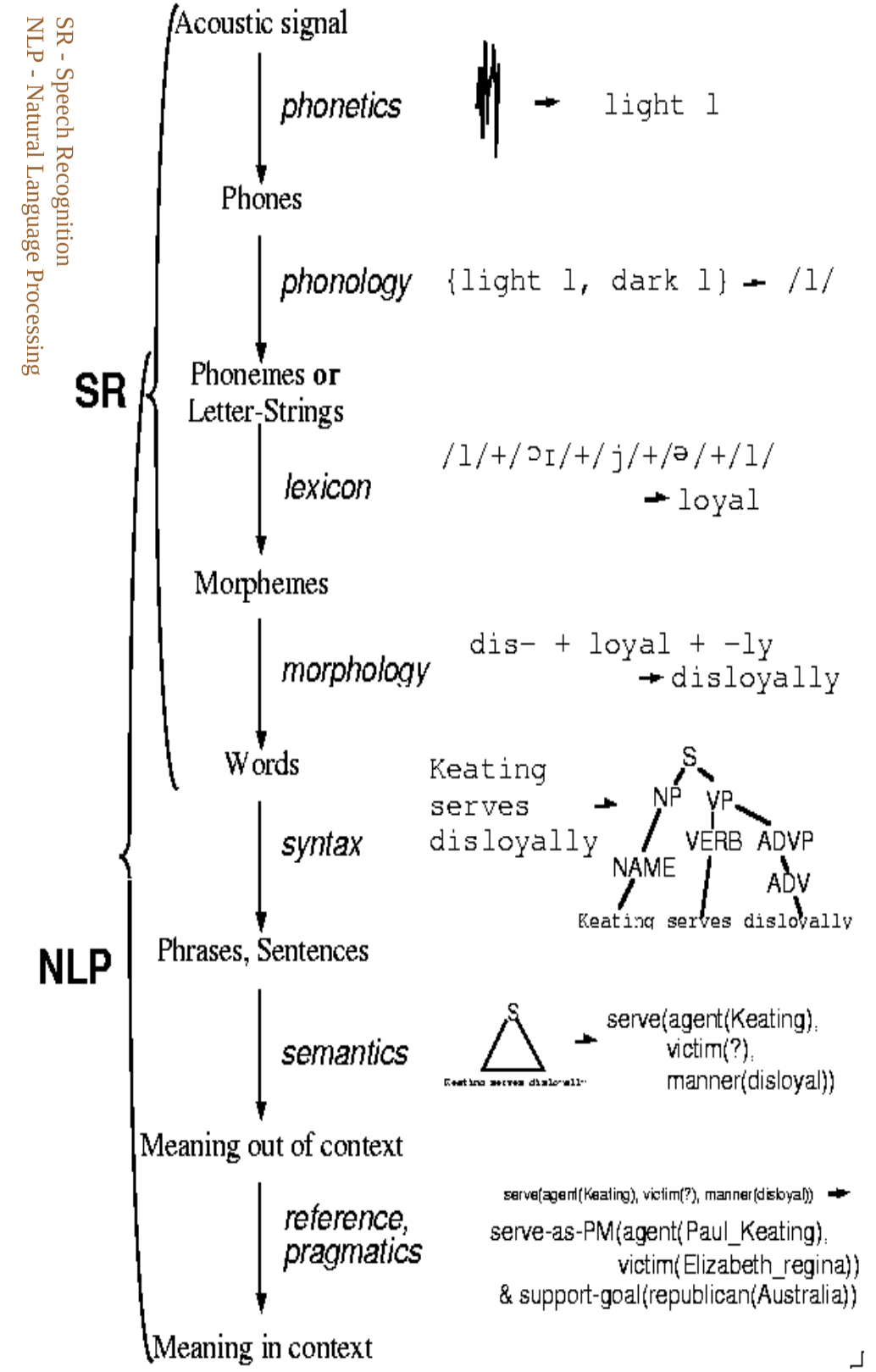
- Quick Review of Lecture 10
- Overall Picture of Linguistics and Language Processing
- NLP Stages in Producing an Internal Representation of a Sentence
- Context Free Grammars
  - Components of a Natural Language
  - SMC Example: Sue, mouse & the cat
  - Top-down parsing (A)
  - Bottom-up parsing (A)
  - Top-down parsing (B) - using parse tree
  - Bottom-up parsing (B) - using parse tree
  - JA Example: John and apple
  - Alternative CFG Visualization
  - Proving that a list of words is a sentence
- Class Workout 1, 2 & 3: AI Exam 1995, 1996 & 1997 NLP questions
- Chomsky's Grammar Hierarchy
  - Type 0, 1, 2 & 3
- Grammar for Natural Language
  - Attribute-Value Grammar
- Semantics Interpretation
  - Logical Form
  - Logical Form of Phrases Examples
  - Lambda Calculus
  - Reduction Rules
  - α-reduction
  - β-reduction
  - Semantics Annotations
  - Semantics Ambiguity
- Conclusion
- Class Activity 1, 2 & 3
- What's in Store for Lecture 12

Quick Review on Lecture 10

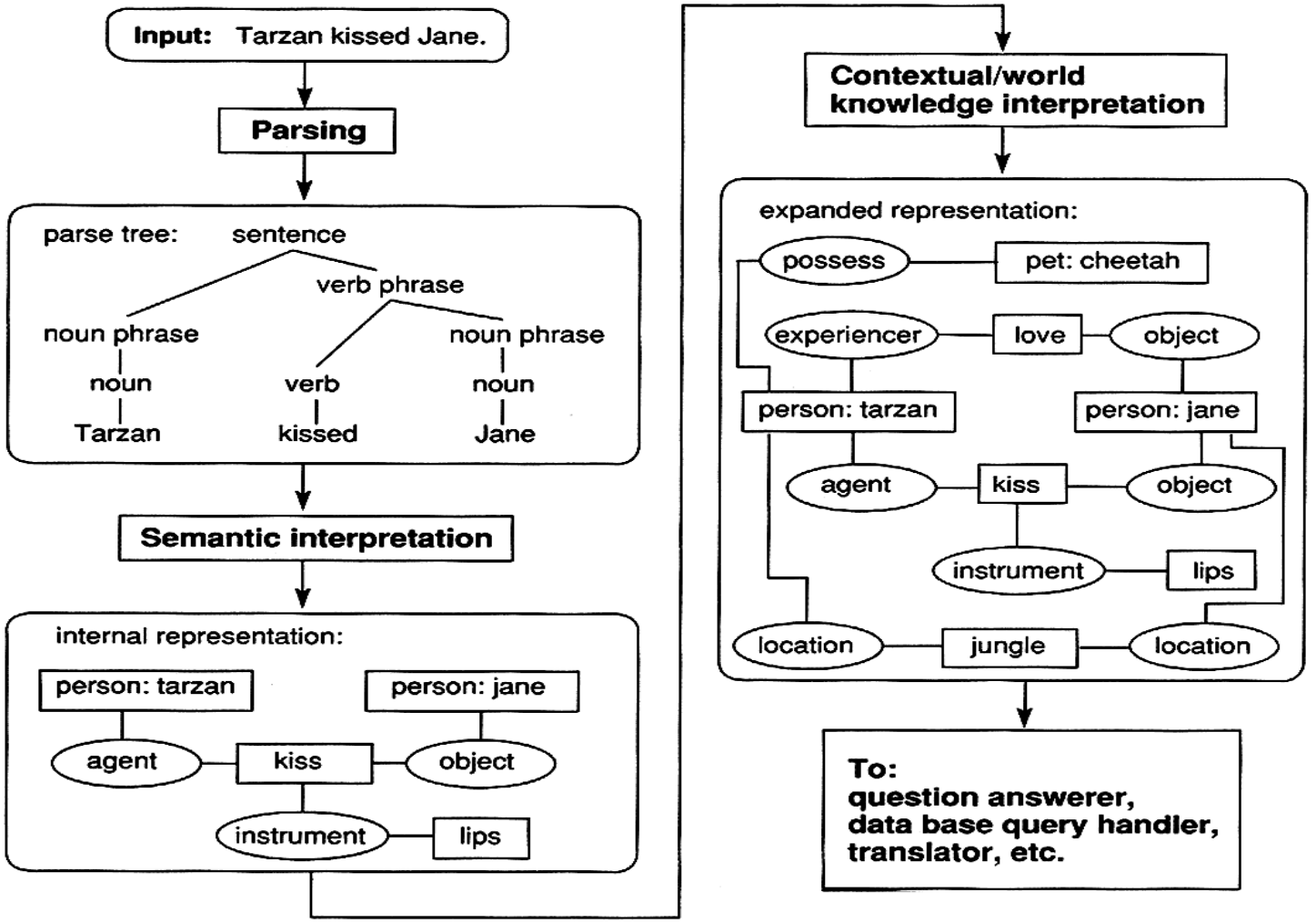
- Introduction to NLP
- NL and Computer Language
- Motivations for NLP
- NLP History
- Major NLP Accomplishments
- Real World NLP Applications
  - MT: Deluxe Universal Translator
  - IR: Buzzcity
  - IR: Altavista Search Engine
  - IV: Cartia's Themescape
  - Autonomous interacting bots: Eliza's grand-daughter - Lisa
  - Grammer Checking Systems: MS Word Grammer Checker
- A Generic NL System Architecture
- Language and Knowledge
- Five Processing Stages in a NLP System
  (1) Phonological Analysis
  (2) Morphological Analysis
  (3) Syntactic Analysis
  (4) Semantic Analysis
  (5) Pragmatic Analysis
- Class Activity: Real-world Paper Reading
- Students' Mini Research Presentation by Group E

Overall Picture of Linguistics and Language Processing

SR - Speech Recognition
NLP - Natural Language Processing



NLP Stages in Producing an Internal Representation of a Sentence



Context Free Grammars

- A **context-free grammar** (CFG) is a formal system that describes a language by **specifying how any legal text** can be derived from a distinguished symbol called the **axiom**, or **sentence symbol**.
- It consists of a **set of productions**, each of which states that a given **symbol** can be replaced by a given **sequence of symbols**.
- An alternative shorter explanation of CFG is that CFG specifies **rewrite rules** for how a **symbol** can be replaced by a **series of symbols and words**.
- Parsing tries to determine how a **sentence can be generated from a grammar**.

Context Free Grammars

Components of a Natural Language: Types of Words

Content Words: Identifies a part of the world

- **Nouns**: person, place, or thing (child, room, hammer)
- **Adjectives**: properties of objects (tall, dark, pretty)
- **Verbs**: actions, relationships (edit, load, debug, link)
- **Adverbs**: properties of verbs (slowly, frequently, nally)

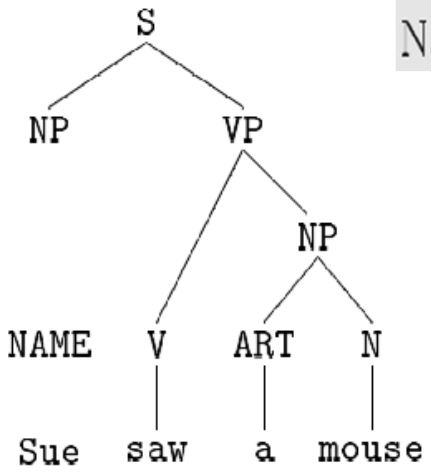
Function Words: Glues words, phrases together

- **Determiners**: indicate specific object (a, the, that)
- **Quantifiers**: how many are identified (all, some, none)
- **Prepositions**: relates phrases (at, on, of, about)
- **Connectives**: connect words, phrases (and, but, when)

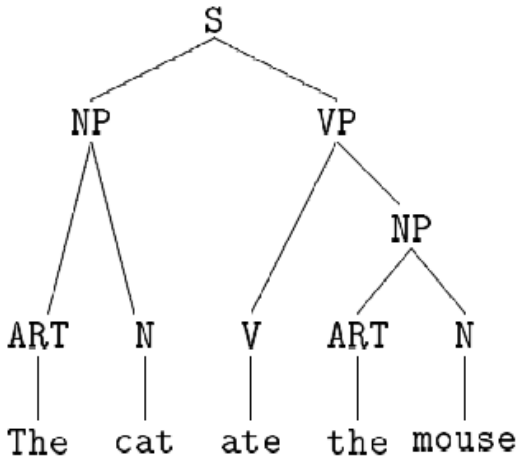
Context Free Grammars

SMC Example: Sue, mouse & the cat

Sentence	S → NP VP
Verb Phrase	VP → V NP
Noun Phrase	NP → NAME
	NP → ART N
Article	ART → a | the
Verb	V → ate | saw
Noun	N → cat | mouse
Name	N → Sue | Zak



Parse tree 1 (Parse tree is also known as derivation tree)



Parse tree 2

Context Free Grammars

SMC Example: Top-down parsing (A)

- **Top-down** parsing **starts with the S symbol** and tries to rewrite it into the **sentence**.

S →	NP VP	using S rule
→	NAME VP	using 1st NP rule
→	Sue VP	using NAME rule
→	Sue V NP	using VP rule
→	Sue saw NP	using V rule
→	Sue saw ART N	using 2nd NP rule
→	Sue saw the N	using ART rule
→	Sue saw the mouse	using N rule

Context Free Grammars

SMC Example: Bottom-up parsing (A)

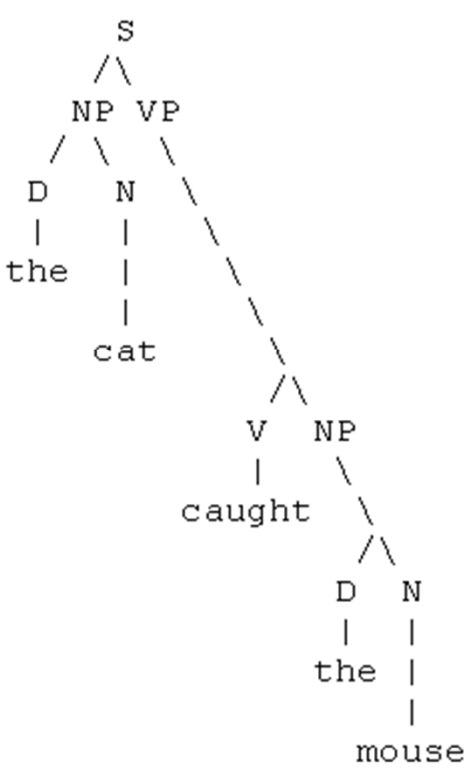
- **Bottom-up** parsing **starts with the words** and tries to **find symbols** that generate them.

```
ART (1)   → The (1)
N (2)     → cat (2)
NP (1-2)  → ART (1) N (2)
V (3)     → ate (3)
ART (4)   → the (4)
N (5)     → mouse (5)
NP (4-5)  → ART (4) N (5)
VP (3-5)  → V (3) NP (4-5)
S (1-5)   → NP (1-2) VP (3-5)
```

Context Free Grammars

SMC Example: Top-down parsing (B) - using parse tree

PARSE TREE

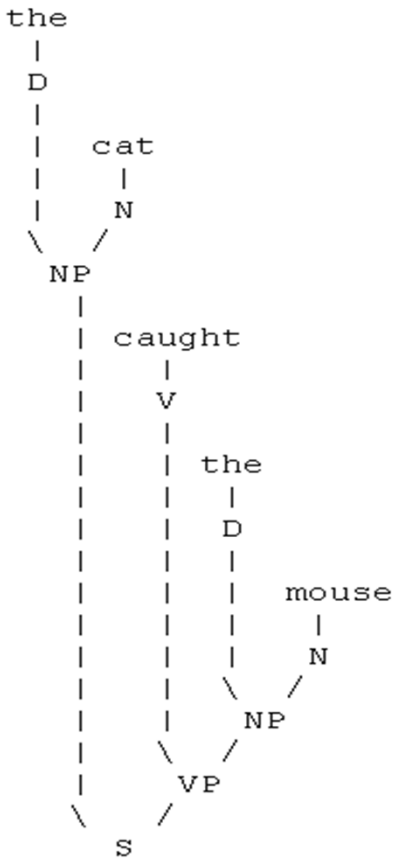


UNRECOGNIZED INPUT
the cat caught the mouse
the cat caught the mouse
the cat caught the mouse
cat caught the mouse
caught the mouse
caught the mouse
the mouse
the mouse
mouse

Context Free Grammars

SMC Example: Bottom-up parsing (B) - using parse tree

PARSE TREE



UNRECOGNIZED INPUT
the cat caught the mouse
cat caught the mouse
cat caught the mouse
caught the mouse
caught the mouse
caught the mouse
the mouse
the mouse
mouse
mouse

Context Free Grammars

JA Example: John and apple

```
S -> NP VP
VP -> VERB NP
NP -> PROPER_NAME
NP -> ART NOUN
ART -> the
PROPER_NAME -> John
VERB -> ate
NOUN -> apple
```

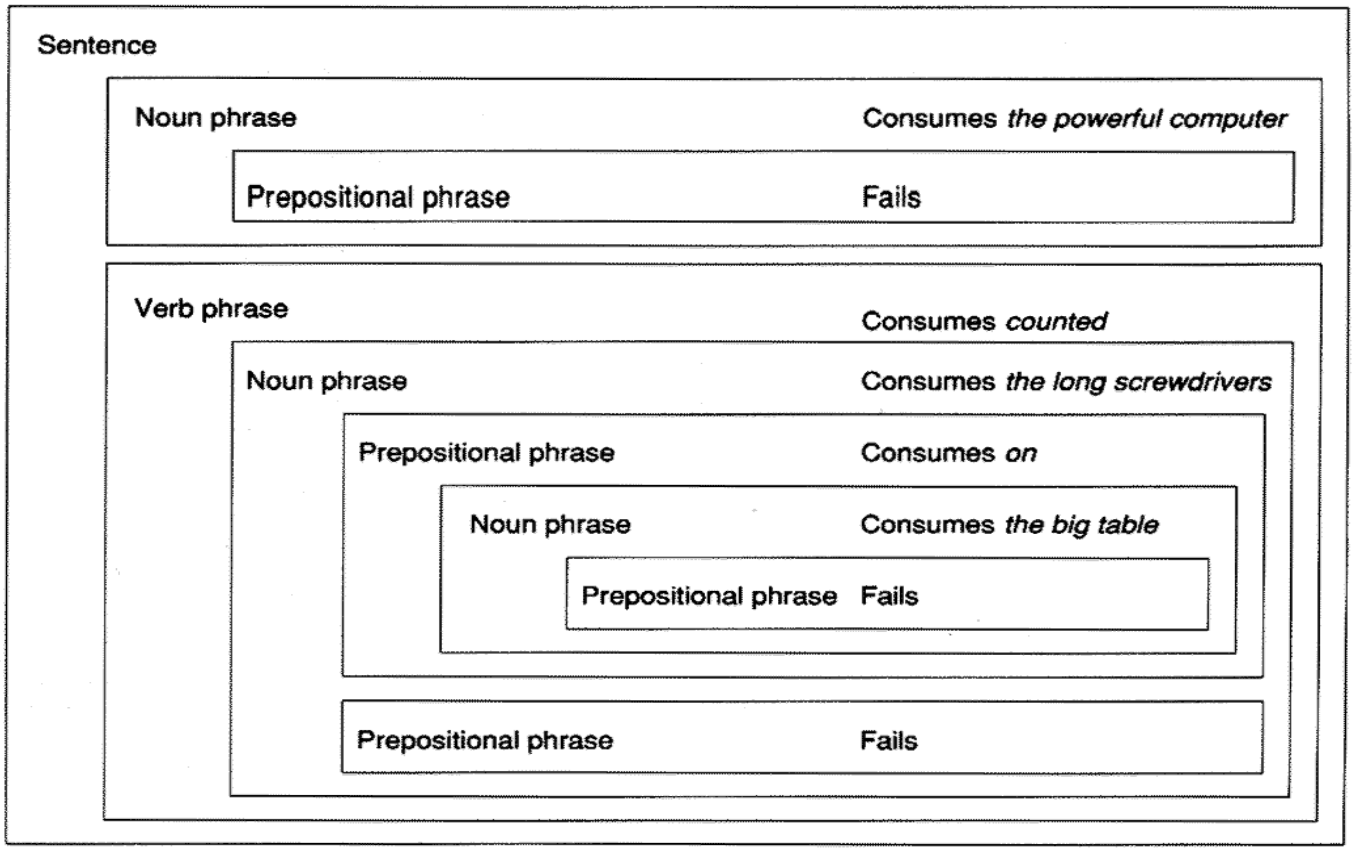
Top-down Parsing

```
S
NP VP
PROPER_NAME VP
John VP
John VERB NP
John ate NP
John ate ART NOUN
John ate the NOUN
John ate the apple
```

Bottom-up Parsing

```
John ate the apple
PROPER_NAME ate the apple
NP ate the the apple
NP VERB the apple
NP VERB ART apple
NP VERB ART NOUN
NP VERB NP
NP VP
S
```

Context Free Grammars
Alternative CFG Visualization



Analysis of the sentence, "**The powerful computer counted the long screwdrivers on the big table**", using a semantic transition tree grammar.

Context Free Grammars

Proving that a list of words is a sentence

```
S -> NP VP
VP -> VERB NP
NP -> PROPER_NAME
NP -> ART NOUN
ART -> the
PROPER_NAME -> john
VERB -> ate
NOUN -> apple
```

- Break the **sentence into phrases**, and prove that the **phrases are non-terminal S**.

Class Workout 1

AI Exam 1995, Q4b

sentence	-->	noun_phrase, verb_phrase.
verb_phrase	-->	verb, noun_phrase.
noun_phrase	-->	determiner, noun.
verb	-->	[teach].
determiner	-->	[the].
determiner	-->	[some].
noun	-->	[students].
noun	-->	[tutors].

GrpA: 4b(i)
GrpB: 4b(ii)
GrpC: 4b(iii)
GrpD: 4b(iv)
GrpE: 4b(v)

(b) Which of the following sentences will parse successfully according to the grammar above? Give a brief explanation for those that will not.

[3]

(i) the tutors teach the students.
(ii) the students teach the tutors.
(iii) some students teach tutors.
(iv) the tutors teach.
(v) the tutors teach the tutors.

Class Workout 1

AI Exam 1995, Q4c

GrpA: 4c
GrpB: 4c

(c) Give examples of two sentences which will not parse according to the grammar above, but will parse with the addition of the following: [2]

noun	-->	[student]
noun	-->	[tutor]
verb	-->	[teaches]

Class Workout 2

AI Exam 1996, Q4a

GrpC: 4a(i)
GrpD: 4a(ii)
GrpE: 4a(ii)

Question 4 A natural language system has the following grammatical and lexical rules:

determiner	⟶	[a]
determiner	⟶	[every]
noun	⟶	[man]
noun	⟶	[woman]
noun	⟶	[telescope]
verb	⟶	[saw]
preposition	⟶	[with]
sentence	⟶	noun-phrase verb-phrase
noun-phrase	⟶	determiner noun
verb-phrase	⟶	verb noun-phrase
verb-phrase	⟶	verb-phrase prepositional-phrase
prepositional-phrase	⟶	preposition noun-phrase

(a) Using syntactic trees, or otherwise, show the analyses that the grammar gives for the following sentences:

(i) 'Every woman saw a man'

(ii) 'A man saw every woman with a telescope'

[7]

Class Workout 2

AI Exam 1996, Q4b & Q4c

(b) Suggest additional rules that would enable the grammar to parse the following sentences, giving the resultant syntactic analysis in each case:

(i) 'a man laughed'

(ii) 'the women saw a man'

[4]

(c) A better grammar would be able to parse: "the man walks" and "the men walk" but should find mistakes with "the man walk" and "the men walks". Describe how you might elaborate the above grammar and lexicon formalism to capture these syntactic details. [6]

GrpA: 4b(i)
GrpB: 4b(ii)
GrpC: 4c
GrpD: 4c
GrpE: 4c

Class Workout 3

AI Exam 1997, Q4a & c

Question 4 A natural language system has the following grammatical and lexical rules:

sentence	⟶	noun-phrase verb-phrase
verb-phrase	⟶	verb noun-phrase
noun-phrase	⟶	determiner noun
noun	⟶	adjective noun
proper-noun	⟶	John
determiner	⟶	[a]
determiner	⟶	[every]
noun	⟶	[lecturer]
noun	⟶	[student]
noun	⟶	[course]
adjective	⟶	[bright]
verb	⟶	[teaches]
verb	⟶	[likes]

Class Workout 3

AI Exam 1997, Q4a & c

(a) Which of the following are sentences according to the above grammar? Using syntactic trees, or otherwise, show the analyses that the grammar gives for those that are sentences:

(i) 'A lecturer teaches every student'

(ii) 'Every lecturer likes a bright student'

(iii) 'John teaches a student'

[7]

(c) Suggest additional rules that would enable the grammar to parse the following sentences, giving the resultant syntactic analysis in each case:

(i) 'a student passed'

(ii) 'the student passed the course'

[4]

GrpA: 4a(i)
GrpB: 4a(ii)
GrpC: 4a(iii)
GrpD: 4c(i)
GrpE: 4c(ii)

Chomsky's Grammar Hierarchy

- **Rules** are needed to express the **correct grammatical structure of language** for analysis and synthesis.

- In Chomsky's Grammar Hierarchy format, it is expressed as

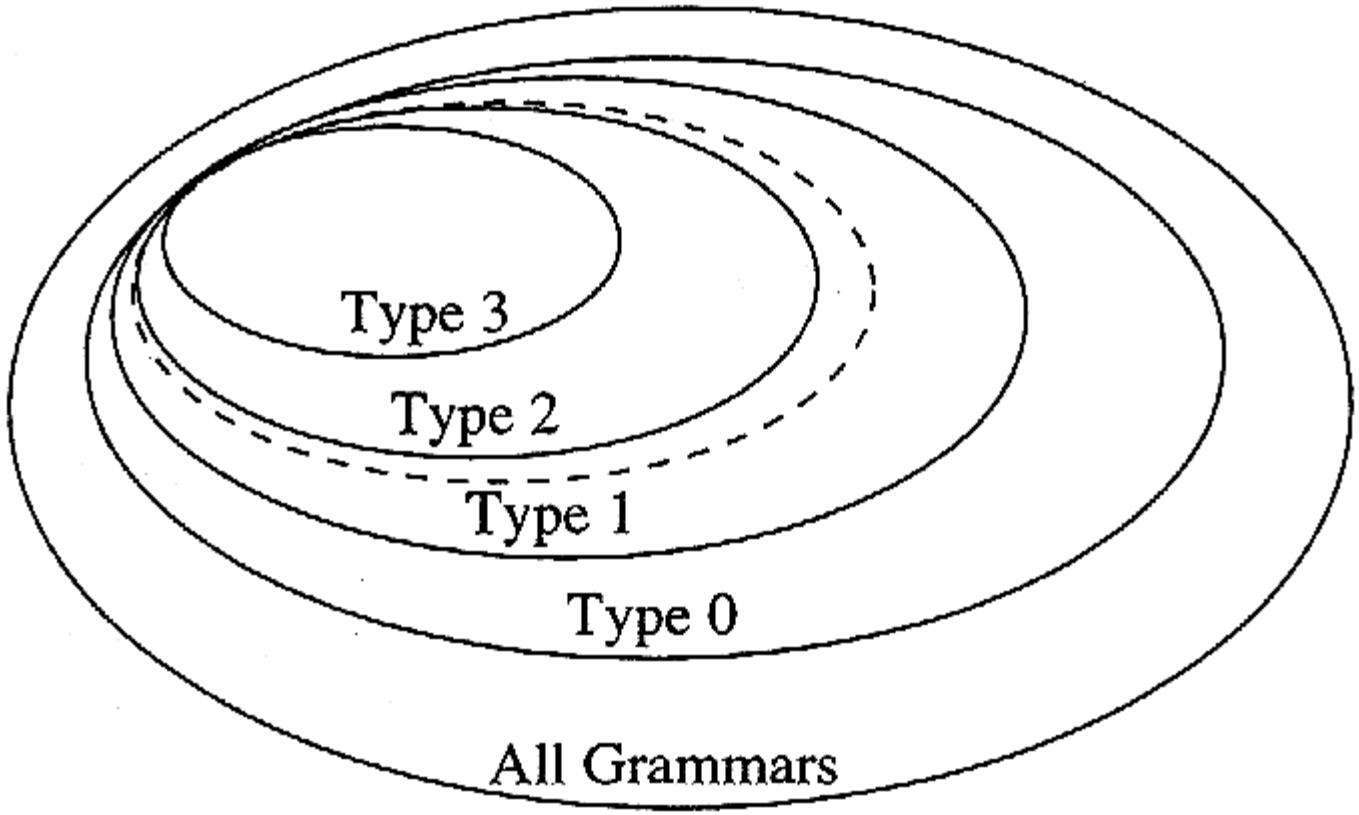
$$A\ B\ C\ \ldots \rightarrow E\ F\ G\ \ldots$$

- This means that the sequences of categories A B C …**can be produced from** the sequences of categories E F G…
- For a string of words to be a sentence in the language, there must be some way of **using these rules to produce the category *S* from the words in the sentences.**

Chomsky's Grammar Hierarchy (cont)

- What are the kind of rules we need?
- We can **classify grammars** according to the **kinds of rules that appear** in them.
- Next, we proceed to classify the languages into **families according to the kinds of rules** that are needed to express their grammar.
- And one such means of classifying grammars and languages is call **Chomsky's Grammar Hierarchy**.

Chomsky's Grammar Hierarchy (cont)



Chomsky's Grammar Hierarchy

Type 0

- Type 0
  Called: Transformation Grammar
  Rules of the form: *anything* → *anything*
  Computational Power: General Turing Machine
  Characteristics String: (can capture any computable dependencies)

Chomsky's Grammar Hierarchy

Type 1

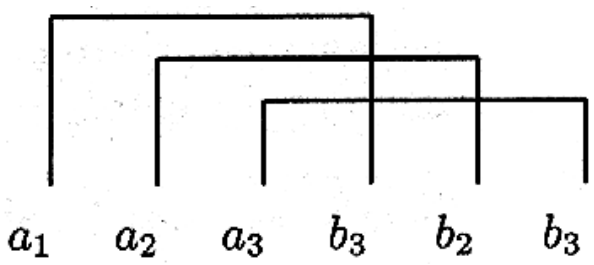
- Type 1
  Called: Context Sensitive Grammar
  Rules of the form: $A\ B\ C \rightarrow A\ D\ C$
  Computational Power: Linear Bound Automata
  Characteristics String: $a^n b^n c^n$

- Type 1 can capture strings with **crossing dependencies.**

- A category a^n is dependent upon another category b^n if **some aspects of it changes according to the nature** of b^n.

Chomsky's Grammar Hierarchy

Type 1 (cont)



- Swiss. German and Dutch allow sentences equivalent to
  $John_1$, $Mary_2$, $Peter_3$, $helped_4$, $teach_5$, $swim_6$.
  (John helped Mary teach Peter to swim.)

- A example in English
  The men and the woman danced and sang respectively.

Chomsky's Grammar Hierarchy

Type 2

- Type 2
  Called: Context Free Grammar
  Rules of the form: $A \rightarrow B\ C\ D...$
  Computational Power: Push Down Stack Automata
  Characteristics String: $a^n b^n$

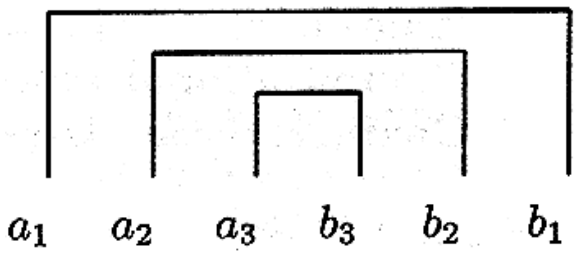
- Context-free analysis can be represented as **tree** (as seen in the earlier section on CFG).

- In such a grammar, we always have **one category going to** one or more categories or a word; like one level of branching in a tree. Combining grammars rules is then like building up a tree.

Chomsky's Grammar Hierarchy

Type 2 (cont)

- Context free grammars can capture nested dependencies of the form:



- **All natural languages display** such nested dependencies.
- Eg. John$_1$, who works for a company$_2$ that makes$_2$ cheese is$_1$ going home.
- This suggest that we **need grammars** that are **at least context free**.

Chomsky's Grammar Hierarchy

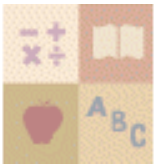
Type 3

- Type 3
  Called: Regular or Right Linear
  Rules of the form: $A \rightarrow x\ B$ and $A \rightarrow x$ where x is a terminal category.
  Computational Power: Finite State Automata
  Characteristics String: a^*b^*

- Such **grammars cannot capture** the **commonest dependencies** present in natural language.

- They have been used for **grammars of morphology**.

Grammar for Natural Language



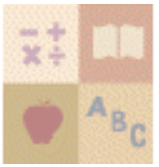
- It is generally accepted that **grammars for natural language** should be **at least context free**.
- Procedures to recognize sentences from context free languages are **guaranteed to terminate** and tell us if the sequence of words is a sentence of the language or not.
- The grammar up till now only deals with **transitive verbs** (verbs that requires a noun phrase to produce a verb phrase). To handle **intransitive verbs**, this can be done by:

  verb-phrase → verb-intrans

  verb-intrans → 'died'

Grammar for Natural Language

Attribute-Value Grammar



```
sentence ← noun_phrase verb_phrase
noun_phrase ← article number noun
noun_phrase ← number noun
number ← singular
number ← plural
article singular ← a singular
article singular ← the singular
article plural ← some plural
article plural ← the plural
singular noun ← dog singular
singular noun ← man singular
plural noun ← men plural
plural noun ← dogs plural
singular verb_phrase ← singular verb
plural verb_phrase ← plural verb
singular verb ← bites
singular verb ← likes
plural verb ← bite
plural verb ← like
```

- In this grammar, the non-terminals **singular** and **plural** provide a context to determine when different article, noun, and verb_phrase rules can be applied.

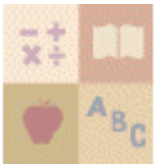
- The derivation of the sentence "The dogs bite" using this grammar is given by

```
The dogs bite.
The dogs plural verb.
The dogs plural verb_phrase.
The plural noun verb_phrase.
article plural noun verb_phrase.
noun_phrase verb_phrase.
sentence.
```

- Similarly, **we could disallow sentences** such as "man bites dog" by **adding a non-terminal, act_of_biting**, to the grammar **which only allows dog** to be the noun.

Grammar for Natural Language

Attribute-Value Grammar (cont)



- Additionally, we also want to capture the **person** agreements that are exemplified by:

  I see the cat.
  We see the cat.
  John sees the cat.
  They see the cat.

- The noun 'I' and 'We' are said to be in the **first person**, whereas 'John' and 'They' are in the **third person**.

- A **naive approach** is to use **many similar rules** or many categories.

- A **better approach** is the use of a much more expressive naming categories available in **attribute-value grammars**.

- Here, we give category names **more structure**, thus rather than

  noun-plural-third,

  we have, $\begin{bmatrix} \text{noun} \\ \text{number} = \text{plural} \\ \text{person} = 3 \end{bmatrix}$

- The equations form **attribute-value pairs**.

- By having variables for the attribute values, we can express **more general grammar rules** for these categories.

$$\begin{bmatrix} \text{np} \\ \text{number}=x \\ \text{person}=y \end{bmatrix} \longrightarrow \begin{bmatrix} \text{det} \\ \text{number}=x \end{bmatrix} \begin{bmatrix} \text{noun} \\ \text{number}=x \\ \text{person}=y \end{bmatrix}$$

Semantics Interpretation

- Our goal is to **mechanically derive some representation in logic** of the meaning of natural language sentences.
- Language is **generative**[1], we **cannot list all the possible sentences**, and **neither can we list all the sentences and their translation** into a formal language.
- Language syntax is **captured by a finite set of (context free) grammar rules**.
- The meaning of a sentence then depends upon the **meaning of its parts**, as analyzed by the grammar. This is known as **compositionality**.

[1] generative \Gen"er*a*tive\, a. [Cf. F. g['e]n['e]ratif.] Having the power of generating, propagating, originating, or producing. ``That generative particle." --Bentley.

Semantics Interpretation
Logical Form

- A **common way** to model the semantics of NL is by means of a **logical form** which **captures the truth conditions** associated with the sentences.

- eg1. John sleeps

  $sleeps(john)$

- eg2. Every student wrote a program

  $\forall s\ student(s) \rightarrow \exists p\ (program(p) \wedge wrote\ (s,p))$

- In order to construct the logical form systematically, we appeal to the **Principle of Compositionity** according to which the **logical form** of a complex phrase is a **function** of the **logical forms of its sub-phrases**.

Semantics Interpretation

Logical Form of Phrases Examples

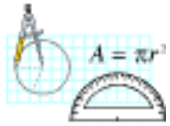
Constituent	Logical Form
proper noun Mike	logical constant mike
common noun cow	1-place predicate cow(x)
adjective red	1-place predicate red(x)

Semantics Interpretation

Logical Form of Phrases Examples (cont)

Constituent	Logical Form
noun vs adjectives big, red car	conjunction car(x) $\wedge$ big(x) $\wedge$ red(x)
intrans. verb phrase john sleeps	1-place predicate sleeps(john)
trans. verb phrase john kicked the dog	2-place predicate kick(john, fido)

Semantics Interpretation
Lambda Calculus



- The logical form of most non-sentential[1] phrases are mostly **incomplete logical propositions** with **certain parts missing**.

- For example: a verb phrase such as **sleeps** can be thought of as a **incomplete proposition** that is missing an argument. More usefully, we can treat it as a function that **maps from** (the logical form of) **noun phrases to a complete proposition**.

- We can model this behaviour by extending the logical form language to include **lambda expressions**

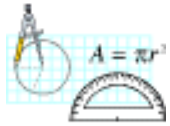
  $\lambda x.\phi$

  where **x is the variable marking the argument** of the function.

[1] non-sentential - not of or not pertaining to a sentence, or full period; as, a sentential pause.

Semantics Interpretation

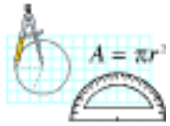
Lambda Calculus: Reduction Rules



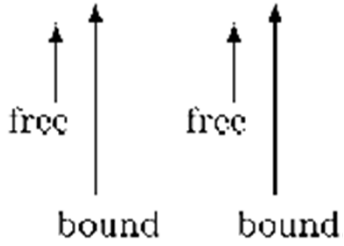
- λ-expressions can be used to **represent data objects** like numbers, strings, etc.
- For example, an arithmetic expression like (2+3)*5 can be represented as a λ-expression and its ‘value’ 25 can also be represented by a λ-expression.
- The process or ‘simplifying’ (2+3)*5 is done by a process call **reduction** (or conversion).
- There are **three kinds** of λ-reduction call α-reduction, β-reduction and η-reduction.

Semantics Interpretation

Lambda Calculus: Reduction Rules (cont)



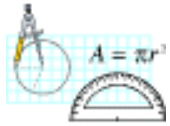
- Here, we will only be covering α-reduction and β-reduction
- An occurrence of a variable V in a λ-expression is **free** if it is **not within the scope** of a `λV ', **otherwise** it is **bound**. For example $(\lambda x.\ y\ x)(\lambda y.\ x\ y)$



- The following notation will be used:
  - $E_1 =_\alpha E_2$ means E_1 α-reduces to E_2
  - $E_1 =_\beta E_2$ means E_1 β-reduces to E_2

Semantics Interpretation

Lambda Calculus: α-reduction



- α-reduction says that **two abstracts are the same** if we can **rename the abstracted variable systematically**, ie.

  $\lambda x.t =_\alpha \lambda y.t[x := y]$ provided that y is not already free in t

- For example, if we use the λ-calculus together **with arithmetic expressions**, then we can have:

  $\lambda x.(x + 5) =_\alpha \lambda y.(y + 5)$

  We can think of this as a function that adds five.

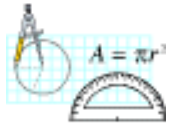
- The following equivalence **does not hold**

  $\lambda x.(x + y) \neq \lambda y.(y + y)$ (as *y* is free in '*t*')

  Informally, we can notice that both are indeed different functions, one **adds *x* to *y***, while the others just **doubles its parameter**.

Semantics Interpretation

Lambda Calculus: β-reduction



- The **most important** kind of reduction among α, β and η-reduction is **β-reduction**.

- Essentially, **substitution is done** for the argument in **every position** where the **abstracted variable occurred**.

$$(\lambda x.t)s =_\beta t[x := s]$$

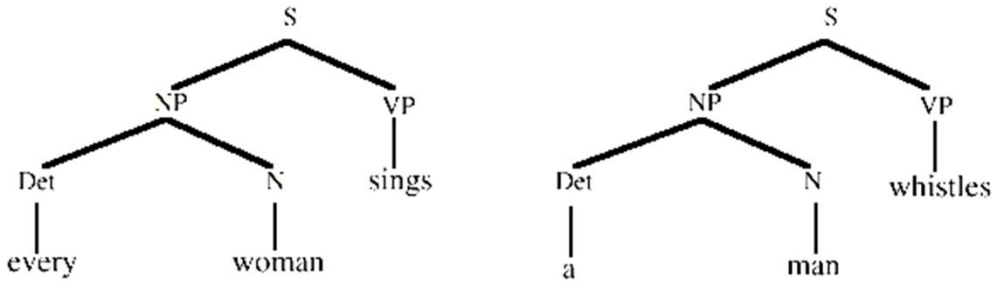
- For example,

$$(\lambda x.(x + 5))6 =_\beta (x + 5)[x := 6]$$
$$= 6 + 5$$
$$= 11$$

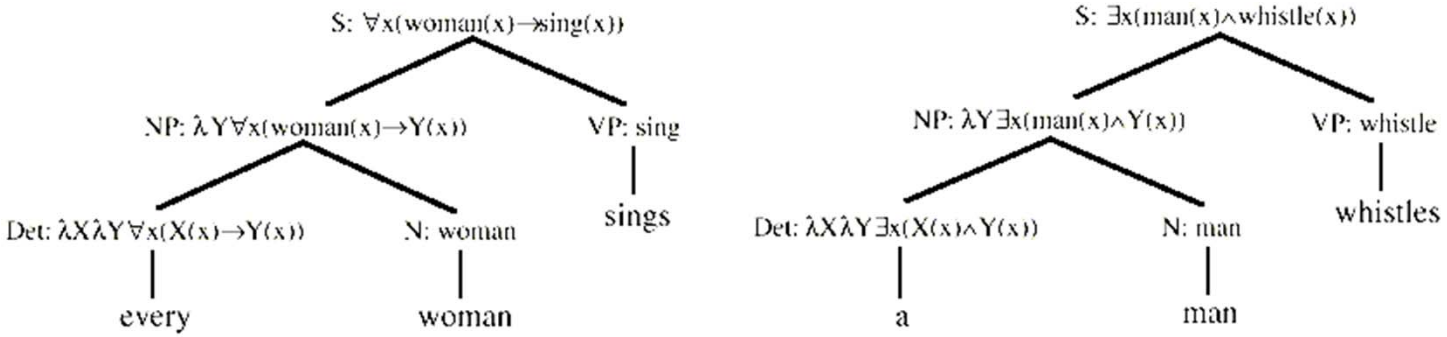
Semantics Interpretation

Semantics Annotations

- Below are **two derivation trees** from a simple paragraph:



- Using λ-calculus, we can **annotate** the above tree as follows:



Semantics Interpretation

Semantics Ambiguity

- When there is **more than one syntactic analysis** of a sentence, then we would expect **more than one semantic interpretation**.

- There are occasions when **one syntactic analysis** of a sentence seems to require **more than one semantic interpretation**. A trivial example of this is when there is **lexical ambiguity**, and a word of one category (eg. noun) has two meanings, as with ‘bank’ (which could mean either a financial institution, or an earth barrier).

Semantics Interpretation

Semantics Ambiguity (cont)

- For example:

  Every man loves a woman.

  could be interpreted **to mean either**:

  $\forall m\ man(m) \rightarrow \exists w\ (woman(w) \wedge loves\ (m, w))$

  or

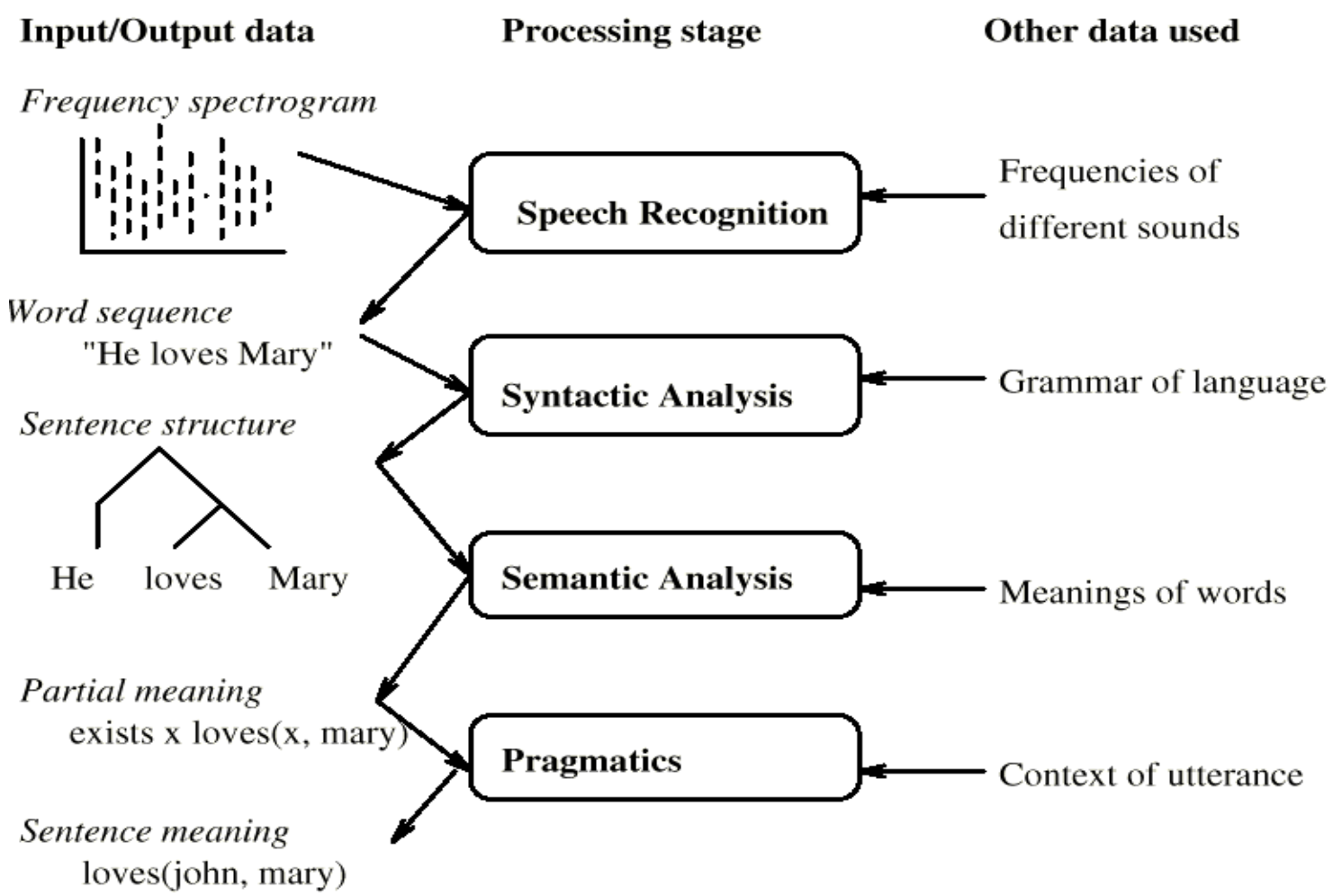
  $\exists w\ woman(w) \wedge \forall m\ (man(m)) \rightarrow\ loves\ (m, w))$

- A **strictly compositional analysis** will only find **one quantifier scoping**, as different quantifier scopings arise for each syntactic analysis.

- **Extra machinery** is required to obtain the additional readings, and to **use the context** to **rule-out inappropriate interpretations**.

Conclusion

Holistic Picture including Speech Component



Class Activity 1: Real-world Article Reading

Article 1. “Natural Language Processing and Windows 2000”

Natural language processing (NLP) is a hybrid field combining Artificial Intelligence (AI) and linguistics, and is a subject that has recently come under scrutiny mostly because it is a highly touted feature of the up-coming Microsoft 'Windows 2000’ operating system. I am not sure how far Microsoft is going to be pushing towards true natural language processing, which involves much more than programs that analyze and construct syntactic structures...

Class Activity 2: Real-world Paper Reading

Paper 1. “An Overview of Empirical Natural Language Processing”

- A Brief History of Natural Language Processing
- Reasons for the Resurgence of Empiricism
- Categories of Empirical Methods
- Categories of Language Task
- Machine Translation

Class Activity 3: Real-world Paper Reading

Paper 2. “GRAMMAR: Words and Their Arrangement”

- Introduction
  - Some Preliminaries
- The Clause Rank
  - The Subject
  - The Verb
  - The Direct Object
  - The Indirect Object
  - The Object Complement
  - The Subject Complement
  - The Adverbial and Adverbial Complement
- Grammatical Ambiguity
- References

What's in Store for Lecture 12

- Why Philosophy is a Weak Topic Among Earlier Batches?
- Introduction to Philosophical Aspects of AI
- List of Papers for Philosophy I
  1. Computing Machinery and Intelligence
  2. Glossary of Philosophical Terms
  3. Mapping Great Debates: Can Computers Think?
  4. The Turing Test & Chinese Room Experiment
  5. Misguided Artificial Intelligence: The Turing Test
  6. Twelve Reasons to Toss the Turing Test
- List of Papers for Philosophy I (cont)
  7. Philosophy of AI: Part of Contemporary Philosophy of Mind - An Annotated Bibliography
  8. How to Pass the Turing Test by Cheating
  9. Lessons from a Restricted Turing Test
  10. Chatterbots, Tinymuds, And The Turing Test: Entering the Loebner Prize Competition
  11. Introducing MegaHal
  12. HeX Template
  13. Joseph Weintraub & His Therapist
  14. Chess is Too Easy

End of Lecture 11

Good Night.