Applied Human Language Technology

Lecture 4

Part 2: Sentences, Rewrite Rules, Parsing and Syntax

Irene Murtagh

Continuing on from last week……

Example of Dative Shift

a. Van Helsing [$_{VP}$ sent [the stake] [to Count Dracula]].

b. V [$_{NP}$ DIRECT OBJECT] [$_{PP}$ INDIRECT OBJECT]].

vs.

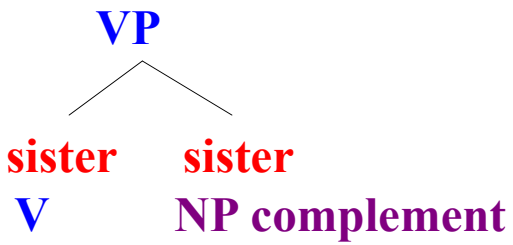
a. Van Helsing [$_{VP}$ sent [Count Dracula] [the stake]].

b. V [$_{NP}$ INDIRECT OBJECT] [$_{NP}$ DIRECT OBJECT]].

Domain

a. Only sisters are relevant for **subcategorisation** ; hence,
b. For any **X**, the frame is **X**P

i.e A head only governs within its phrase.
[$_{VP}$ [V NP]]



Verb Phrases – General Issues

Copula Verb Complements

a. The castle [*is very spooky*]. → COP AP
b. He [*became a haematologist*]. → COP NP_{pred}

Auxiliary Verb Complements

a. They [had *to sleep in the day*]. → AUX VP_{Infin}
b. I [am *hiding in the wardrobe*]. → AUX $VP_{PresPart}$
c. We [must *get out of here before dark*]. → AUX VP_{Stem}

Lexical Verb Complements

a. Something [flew *past my window*]. → V PP_{Mot}

b. She [dropped *the garlic*]. → V NP

c. Someone [put *a coffin in the cellar*]. → V NP PP_{Loc}

d. Who [pushed *the butler into the grave*]. → V NP PP_{Mot}

e. Van Helsing [ordered *the girl to hammer the stake*]. → V NP VP_{Infin}

Transitivity and Verbs

Problem 1 : Knowing when something is an argument of the verb .

Arguments can be:

a. NP: She hated [$_{NP}$ red cars]

b. PP: He put the loot [$_{PP}$ [$_{P}$ in] [$_{NP}$ a sack]]

c. Clause : He wanted [$_{INFL-CL}$ to win the race]

Definition :

If the verb needs the element,
then the element is an argument

a. *she hated.

b. *He put the loot.

c. *He wanted.

In these, the verb wants its missing arguments

Problem 2 : What counts as an argument .

Elements can be either **arguments** or **adjuncts** , like adverbials .

Adverbials are :

a. optional or
b. less attached to the verb.

Any verb can have adverbials.

Intransitive

a. She **died** (yesterday).
b. (Yesterday,) she **died**.

Transitive

a. She **invented** a new drug (yesterday).
b. (Yesterday,) She **invented** a new drug.

Ditransitive (also called **bitransitive**)

a. She **put** her foot on a live wire (yesterday).

b. (Yesterday,) She **put** her foot on a live wire.

The same elements can be arguments or adjuncts in different contexts.

a. She **read** a book [**Adjunct** in the library]

b. She **put** a book [**Argument** in the library]

- The verb **read** is transitive .
- The verb **put** is ditransitive .

Dependency

Heads can be

Verbs
Nouns
Prepositions
Adjectives

The complement depends on the particular head chosen

a. proximity: [_ **To**-P]

b. faith: [_ **In**-P]

Note the use of specific w ords in the frame.

Sometimes specific words need to be mentioned in stating the complement possibilities

Paul [**gave** *the food* ***to*** *the cat*]

GIVE [_ NP **To**-P]

Valence refers to the number of dependents that a head can take, i.e. arguments. Valency includes all arguments including the subject.

Arguments count as dependents of a head, adjuncts do not.

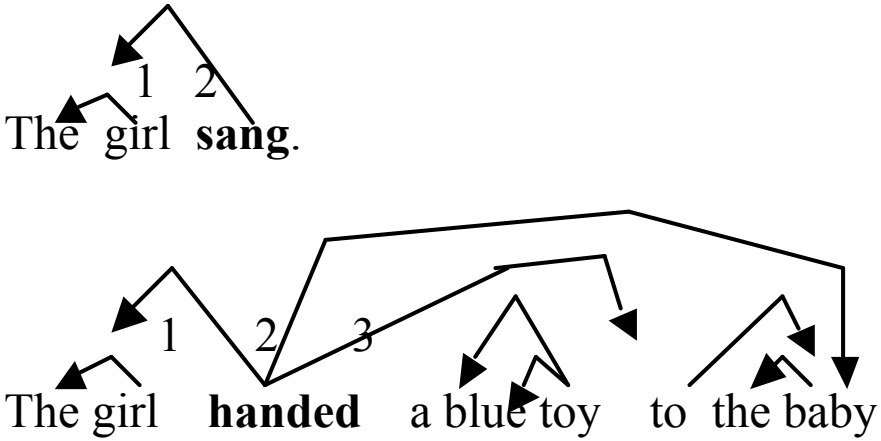


Diagram of Dependency Relations.

Transitive Complements

Some verbs can combine with a **noun phrase**.

Aisling [broke *the bottle*].
BREAK [_ NP]

Some verbs can combine with a **noun phrase** & a **predicate noun phrase**.

Aisling [considers *Sorcha* *a good friend*].
CONSIDER [_ NP NP $_{\textbf{Pred}}$]

Some verbs can combine with a **noun phrase** & an **adjectival phrase**.

Paul [kept *the beer* *cold*].
KEEP [_ NP AdjP]

Some verbs can combine with a **noun phrase** and a **locative phrase** .

Mairéad [**kept** *the cat* ***in*** *the garage*]
KEEP [_ NP LocP]

Some verbs can combine with a **motion phrase** .

She [**moved** *the lamp* ***into*** *the room*]
MOVE [_ NP MotP]

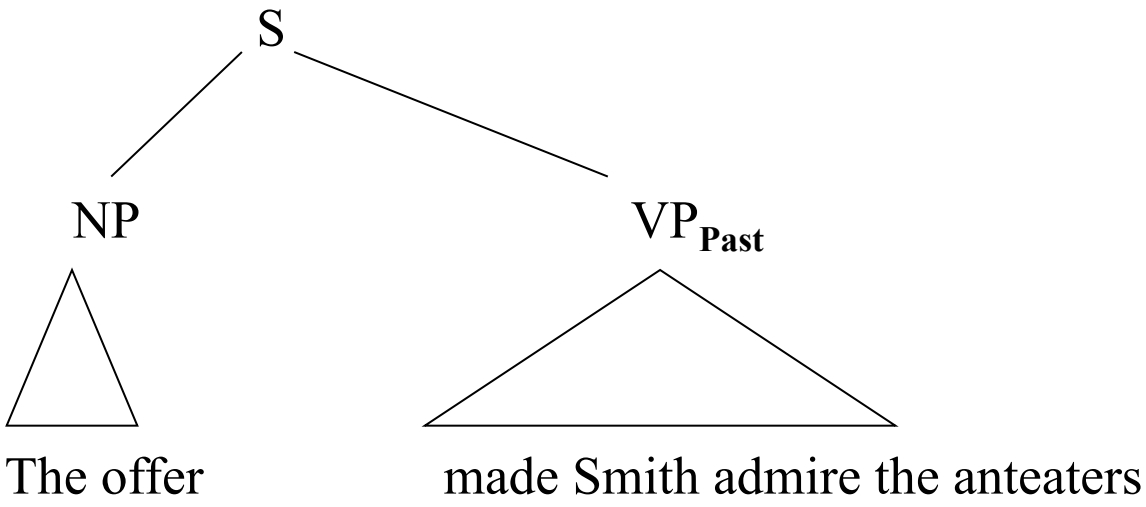
Ditransitive Complements

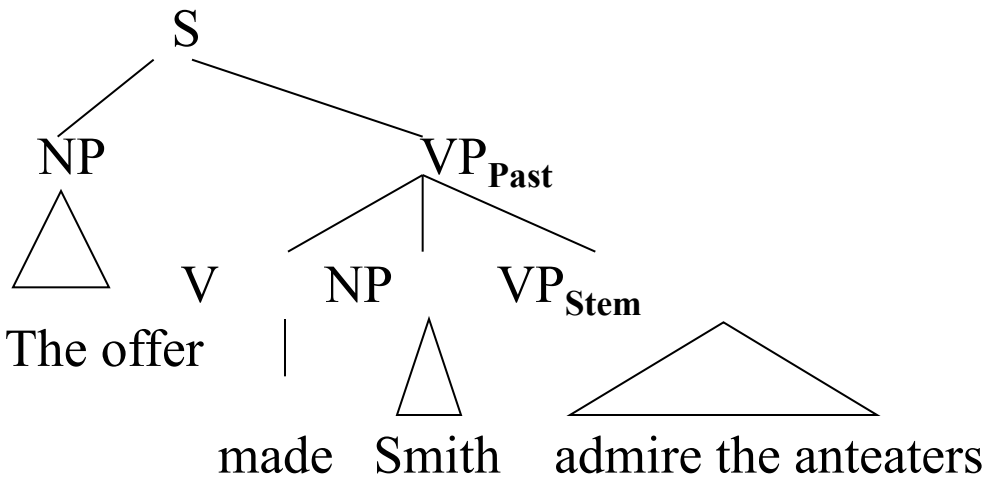
Some verbs can combine with **two noun phrases,** and both of these are arguments.

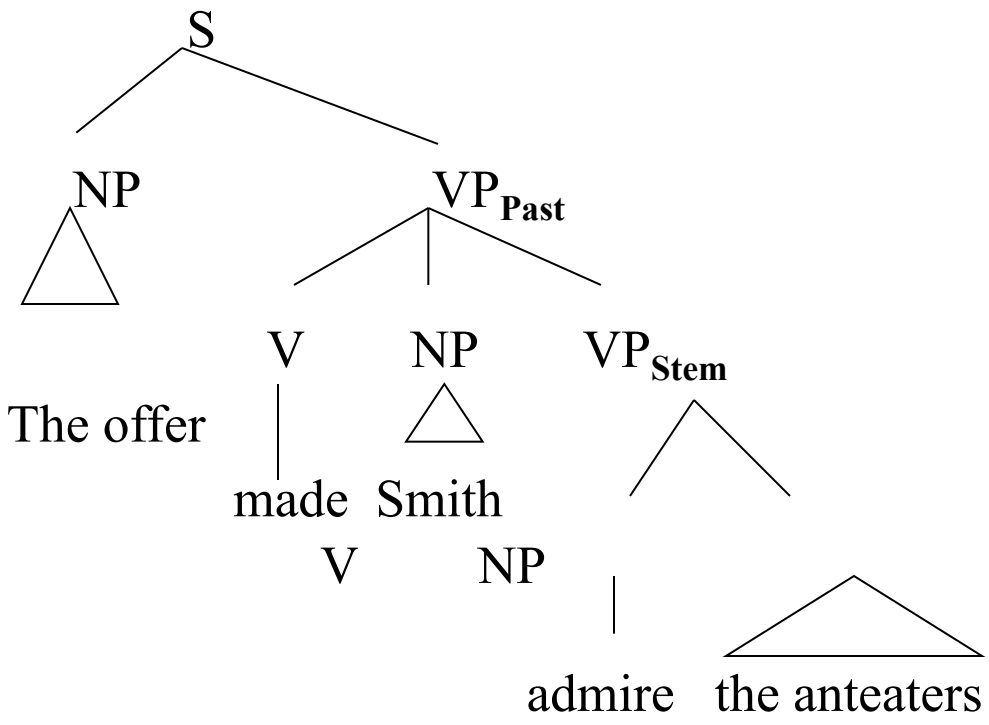
Olga [sent *Brian* *the book*].
SENT [_ NP NP]

A Method of Representing Phrase Structure

The offer made Smith admire the anteaters.

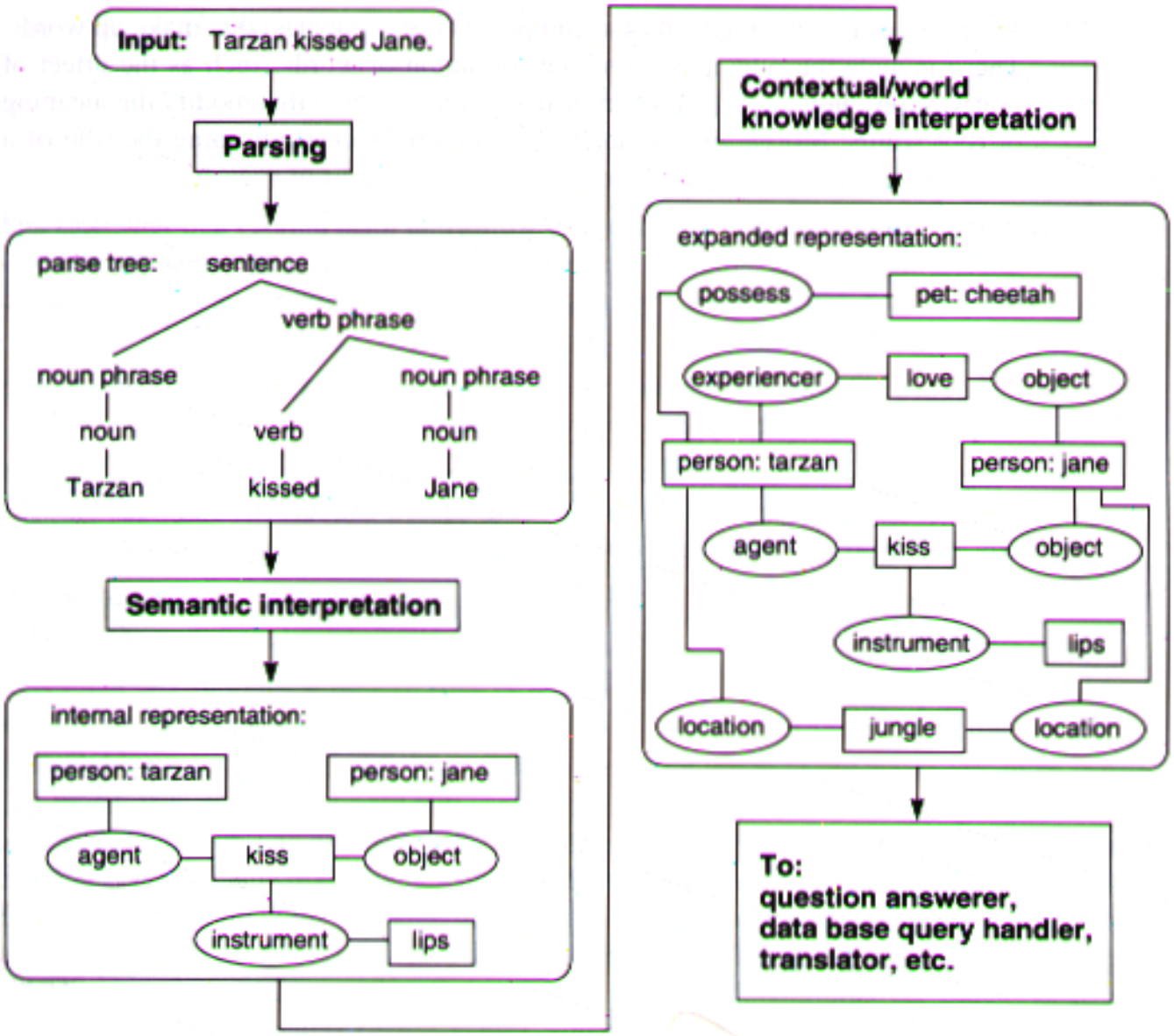



S
NP
VP Past
V
NP
VP Stem
The offer
made
Smith
admire the anteaters


S
NP
VP Past
V
NP
VP Stem
The offer
made
Smith
V
NP
admire
the anteaters

[S [NP the offer] [VPPast [V made] [NP Smith] [VPStem [V admire] [NP the anteaters]]]]

Stages in producing an internal representation if a sentence



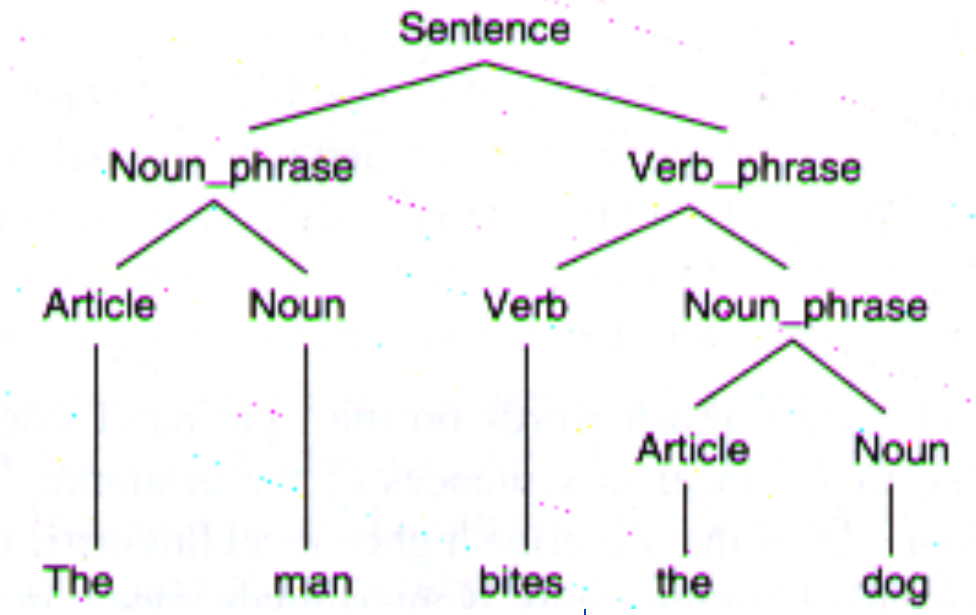
Specification and parsing using context-free grammars

1. sentence -> noun-phrase verb-phrase
2. noun-phrase -> noun
3. noun-phrase -> article noun
4. verb-phrase -> verb
5. verb-phrase -> verb noun-phrase
6. article - > a
7. article -> the
8. noun -> man
9. noun -> dog
10. verb -> likes
11. verb -> bites

"the man bites the dog"

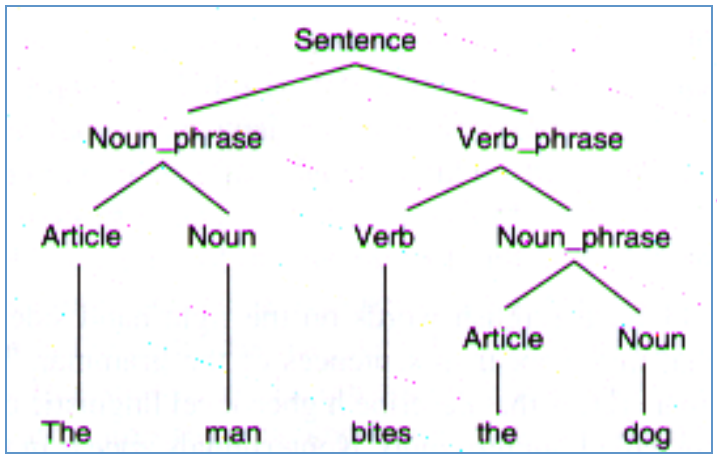
"the man bites the dog"

1. sentence -> noun-phrase verb-phrase
2. noun-phrase -> noun
3. noun-phrase -> article noun
4. verb-phrase -> verb
5. verb-phrase -> verb noun-phrase
6. article - > a
7. article -> the
8. noun -> man
9. noun -> dog
10. verb -> likes
11. verb -> bites

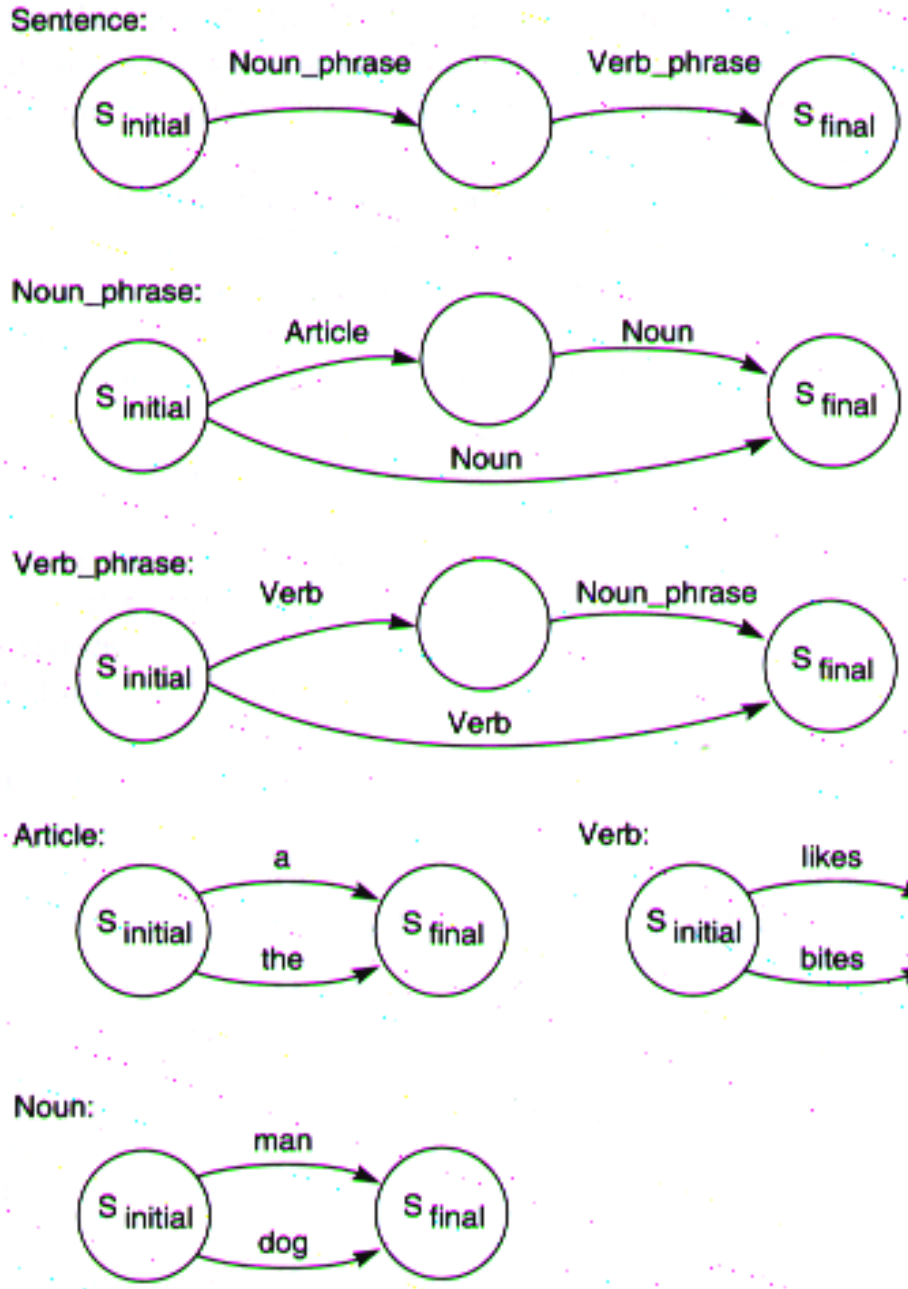


PARSING

sentence	apply R1
noun-phrase verb-phrase	apply R3
article noun verb-phrase	apply R7
The noun verb-phrase	apply R8
The **man** verb-phrase	apply R5
The man **bites** noun-phrase	apply R11
The man bites article noun	apply R3
The man bites **the** noun	apply R7
The man bites the **dog**	apply R9



Represented as a transition network



Lexicon entries for a simple parser

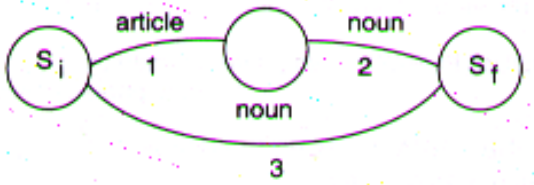
Word	Definition
a	PART_OF_SPEECH: article ROOT: a NUMBER: singular
bite	PART_OF_SPEECH: verb ROOT: bite NUMBER: plural
bites	PART_OF_SPEECH: verb ROOT: bite NUMBER: singular
dog	PART_OF_SPEECH: noun ROOT: dog NUMBER: singular
dogs	PART_OF_SPEECH: noun ROOT: dog NUMBER: plural
like	PART_OF_SPEECH: verb ROOT: like NUMBER: plural
likes	PART_OF_SPEECH: verb ROOT: like NUMBER: singular
man	PART_OF_SPEECH: noun ROOT: man NUMBER: singular
men	PART_OF_SPEECH: noun ROOT: man NUMBER: plural
the	PART_OF_SPEECH: article ROOT: the NUMBER: plural or singular

sentence:


```
procedure sentence-1;
   begin
      NOUN_PHRASE := structure returned by
              noun_phrase network;

      SENTENCE.SUBJECT := NOUN_PHRASE;
   end.
```

```
procedure sentence-2;
   begin
      VERB_PHRASE := structure returned by
              verb_phrase network;

      if NOUN_PHRASE.NUMBER =
                                  VERB_PHRASE.NUMBER
         then begin
            SENTENCE.VERB_PHRASE := VERB_PHRASE;
            return SENTENCE
         end
         else fail
   end.
```

noun_phrase:



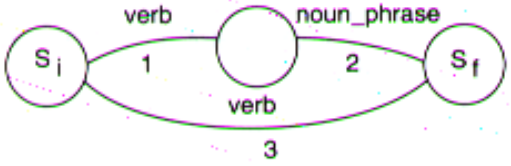

```
procedure noun_phrase-1;
   begin
      ARTICLE := definition frame for next word of input;

      if ARTICLE.PART_OF_SPEECH=article
         then NOUN_PHRASE.DETERMINER := ARTICLE
         else fail
   end.
```

```
procedure noun_phrase-2;
   begin
      NOUN := definition frame for next word of input;

      if NOUN.PART_OF_SPEECH=verb and
            NOUN.NUMBER agrees with
                  NOUN_PHRASE.DETERMINER.NUMBER
         then begin
            NOUN_PHRASE.NOUN := NOUN;
            NOUN_PHRASE.NUMBER := NOUN.NUMBER
            return NOUN_PHRASE
         end
         else fail
   end.
```

Psuedo-code for a grammar that checks number agreement and builds a parse tree

```
procedure noun_phrase-3
   begin
      NOUN := definition frame for next word of input;

      if NOUN.PART_OF_SPEECH=noun
         then begin
            NOUN_PHRASE.DETERMINER := unspecified;
            NOUN_PHRASE.NOUN := NOUN
            NOUN_PHRASE.NUMBER := NOUN.NUMBER
         end
         else fail
   end.
```

verb_phrase:

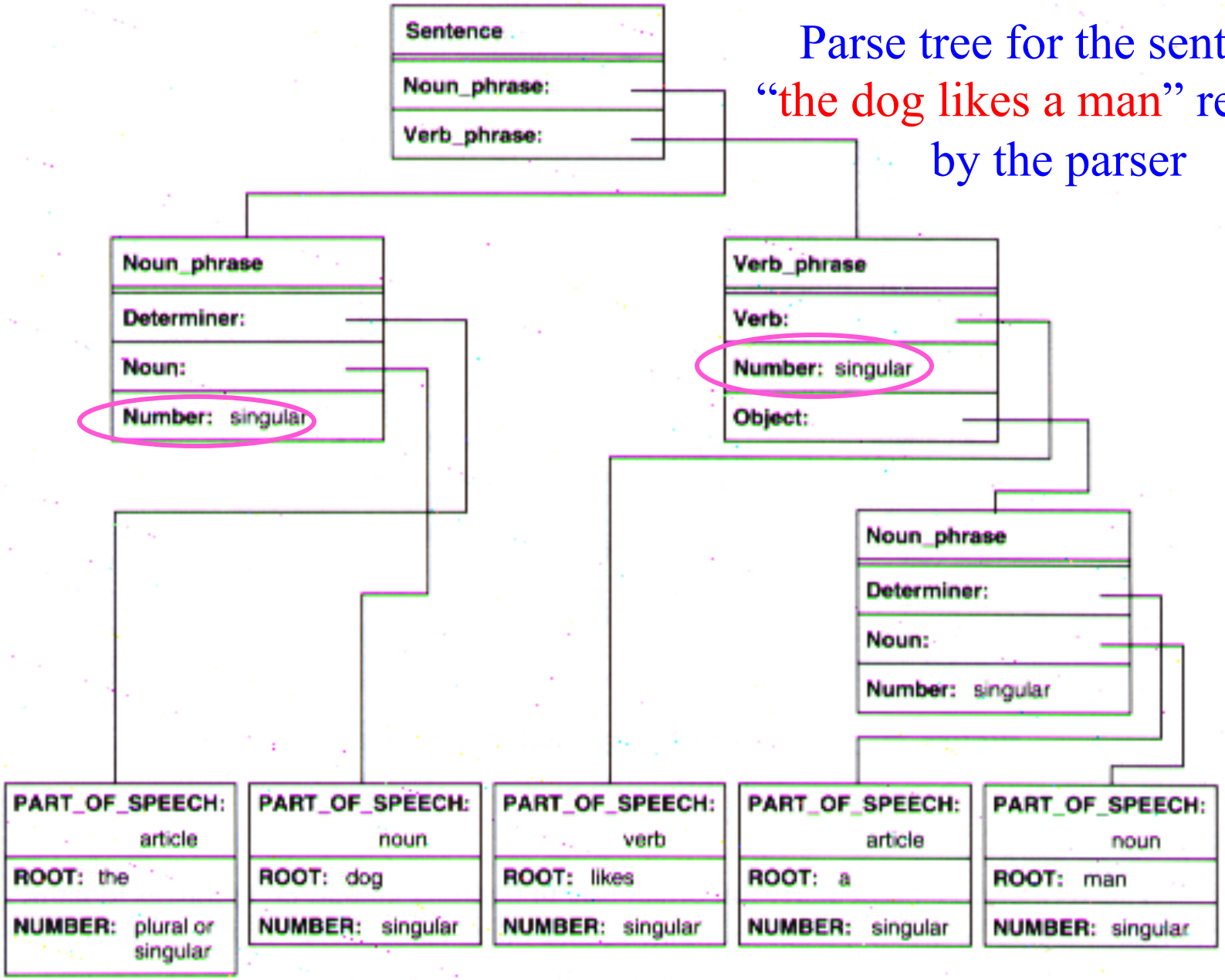



```
procedure verb_phrase-1
   begin
      VERB := definition frame for next word of input;

      if VERB.PART_OF_SPEECH=verb
         then begin
            VERB_PHRASE.VERB := VERB;
            VERB_PHRASE.NUMBER := VERB.NUMBER;
         end;
   end.
```

```
procedure verb_phrase-2
   begin
      NOUN_PHRASE := structure returned by
                                        noun_phrase network;

      VERB_PHRASE.OBJECT := NOUN_PHRASE;
      return VERB_PHRASE
   end.
```

```
procedure verb_phrase-3
   begin
      VERB := definition frame for next word of input;

      if VERB.PART_OF_SPEECH=verb
         then begin
            VERB_PHRASE.VERB := VERB;
            VERB_PHRASE.NUMBER := VERB.NUMBER;
            VERB_PHRASE.OBJECT := unspecified;
            return VERB_PHRASE;
         end;
   end.
```

Parse tree for the sentence "the dog likes a man" returned by the parser



1. sentence -> noun-phrase verb-phrase
2. noun-phrase -> noun
3. noun-phrase -> article noun
4. verb-phrase -> verb
5. verb-phrase -> verb noun-phrase
6. article - > a
7. article -> the
8. noun -> man
9. noun -> dog
10. verb -> likes
11. verb -> bites

S -> NP VP
NP -> N
NP -> DET N
VP -> V
VP -> V NP

DET - > a
DET -> the

N -> man
N -> dog

V -> likes
V -> bites

sample rewrite rules

TO DO THIS WEEK

Read: Ch.8 & 9: *Speech and Language Processing* by Jurafsky & Martin