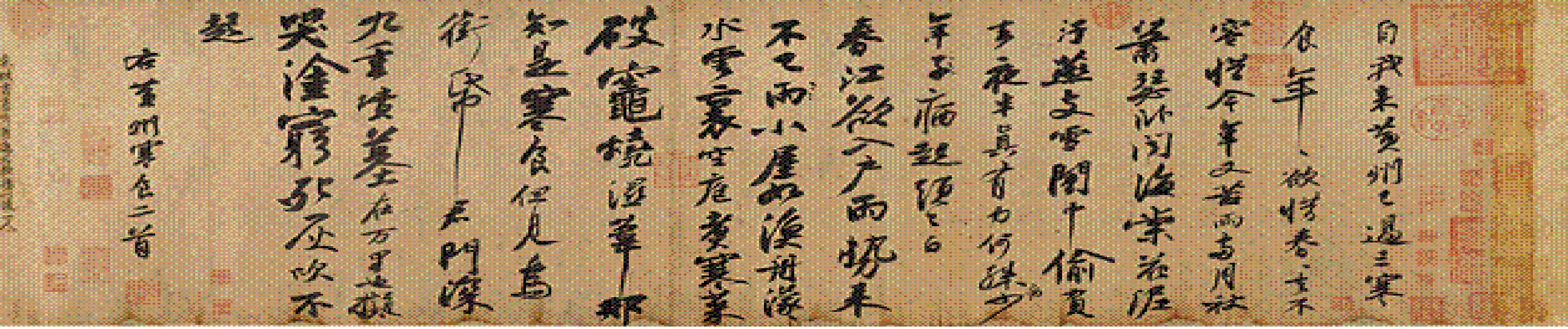


计算语言学
Computational Linguistics

教师：孙茂松
Tel:62781286
Email:sms@tsinghua.edu.cn
TA：林衍凯
Email:linyankai423@qq.com

郑重声明

- 此课件仅供选修清华大学计算机系研究生课《计算语言学》(70240052)的学生个人学习使用，所以只允许学生将其下载、存贮在自己的电脑中。未经孙茂松本人同意，任何人不得以任何方式扩散之（包括不得放到任何服务器上）。否则，由此可能引起的一切涉及知识产权的法律责任，概由该人负责。
- 此课件仅限孙茂松本人讲课使用。除孙茂松本人外，凡授课过程中，PTT文件显示此《郑重声明》之情形，即为侵权使用。

第七章
特征与扩充语法

8.1. Feature Systems and Augmented Grammars

- The number agreement restriction of English
  "*a men*"
  Subject-verb agreement,
  Gender agreement for pronouns,
  Restrictions between the head of a phrase
  ……
  NP -> ART N
  only when NUMBER1 agrees with NUMBER2

  NP-SING -> ART-SING N-SING
  NP-PLURAL -> ART-PLURAL N-PLURAL

  Doubling the size of the grammar.
  失去简洁性，可理解性

8.1. Feature Systems and Augmented Grammars

- Feature structure - a mapping from **features** to **values** that defines the relevant properties of the constituent.

```
ART1:
(CAT ART
 ROOT a
 NUMBER s)
```

The abbreviated form:

```
ART1: (ART ROOT a NUMBER s)
```

8.1. Feature Systems and Augmented Grammars

- Feature structures themselves can occur as values.

"a fish"

```
NP1:(NP NUMBER s
        1 (ART ROOT a
               NUMBER s)
        2 (N ROOT fish
             NUMBER s))
```

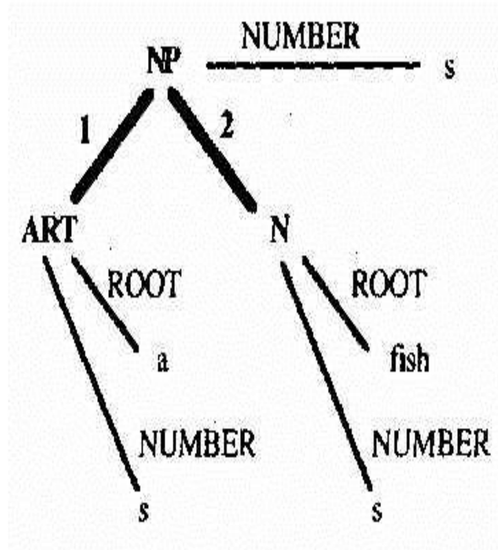


Figure 4.1 Viewing a feature structure as an extended parse tree

1, 2, 3, and so on - will stand for the first subconstituent, second subconstituent, and so on, as needed.

8.1. Feature Systems and Augmented Grammars

- Variables are allowed as feature values so that a rule can apply to a wide range of situations.

  For example, a rule for simple noun phrases would be as follows:

```
(NP NUMBER ?n) ->
            (ART NUMBER ?n) (N NUMBER ?n)
```

This says that an NP constituent can consist of two subconstituents, the first being an ART and the second being an N, in which the NUMBER feature in all three constituents is identical.

8.1. Feature Systems and Augmented Grammars

```
*  (NP 1  (ART NUMBER s)
        2  (N NUMBER s))

*(NP NUMBER s
      1  (ART NUMBER s)
      2  (N NUMBER p))


 (N ROOT fish NUMBER ?n{s p})
```

Or, more simply as

```
 (N ROOT fish NUMBER {s p})
```

8.1. Feature Systems and Augmented Grammars

- An interesting issue: whether an augmented context-free grammar can describe languages that cannot be described by a simple context-free grammar.

The answer depends on the constraints on what can be a feature value：

If the set of feature values is finite, then it would always be possible to create new constituent categories for every combination of features. Thus it is expressively equivalent to a context-free grammar.

If the set of feature values is unconstrained, then such grammars have arbitrary computational power.

8.2. Some Basic Feature Systems for English

- **Person and Number Features**

First Person (1): The noun phrase refers to the speaker, or a group of people including the speaker (for example, I, we, you and I).

Second Person (2): The noun phrase refers to the listener, or a group including the listener but not including the speaker (for example, you, all of you).

Third Person (3): The noun phrase refers to one or more objects, not including the speaker or hearer.

8.2. Some Basic Feature Systems for English

Since number and person features always co-occur, it is convenient to combine the two into a single feature, **AGR**, that has six possible values:

first person singular (1s),
second person singular (2s),
third person singular (3s),
and first, second and third person plural (1p, 2p, and 3p, respectively).

‘is’: its AGR feature would be 3s.
‘are’, its AGR feature would be a variable ranging over the values {2s 1p 2p 3p}.

8.2. Some Basic Feature Systems for English

- **Verb-Form Features and Verb Subcategorization**

Feature values for the feature **`VFORM`** **(关于动词自身)**

`base` - base form (for example, *go*, *be*, *say*, *decide*)
`pres` - simple present tense (for example, *go*, *goes*, *am*, *is*, *say*, *says*, *decide*)
`past` - simple past tense (for example, *went*, *was*, *said*, *decided*)
`fin` - finite (that is, a tensed form, equivalent to {pres past})
`ing` - present participle (for example, *going*, *being*, *saying*, *deciding*)
`pastprt` - past participle (for example, *gone*, *been*, *said*, *decided*)
`inf` - a special feature value that is used for infinitive forms with the word *to*

8.2. Some Basic Feature Systems for English

- **SUBCAT**

If the category is restricted by a feature value, then the feature value follows the constituent separated by a colon.

Value	Example Verb	Example
_none	laugh	Jack laughed.
_np	find	Jack found a key.
_np_np	give	Jack gave Sue the paper.
_vp:inf	want	Jack wants to fly.
_np_vp:inf	tell	Jack told the man to go.
_vp:ing	keep	Jack keeps hoping for the best.
_np_vp:ing	catch	Jack caught Sam looking at his desk.
_np_vp:base	watch	Jack watched Sam look at his desk.

8.2. Some Basic Feature Systems for English

The rule for verbs with a SUBCAT value of _np_vp:inf would be:

```
(VP) -> (V SUBCAT _np_vp:inf)
        (NP)
        (VP VFORM inf)
```

8.2. Some Basic Feature Systems for English

- **PFORM** on prepositional phrases

Value	Example Prepositions	Example
TO	to	I gave it to the bank.
LOC	in, on, by, inside, on top of	I put it on the desk.
MOT	to, from, along, ...	I walked to the store.

Value	Example Verb	Example
_np_pp:to	give	Jack gave the key to the man.
_pp:loc	be	Jack is at the store.
_np_pp:loc	put	Jack put the box in the corner.
_pp:mot	go	Jack went to the store.
_np_pp:mot	take	Jack took the hat to the party.
_adjp	be, seem	Jack is happy.
_np_adjp	keep	Jack kept the dinner hot.
_s:that	believe	Jack believed that the world was flat.
_s:for	hope	Jack hoped for the man to win the prize.

Figure 4.4 Additional SUBCAT values

8.2. Some Basic Feature Systems for English

```
(VP) -> (V SUBCAT _np_pp:loc)
        (NP)
        (PP PFORM LOC)
```

For **embedded sentences**, a complementizer is often needed and must be subcategorized for. Thus a **`COMP`** feature with possible values *for*, *that*, *and nocomp* will be useful.

8.2. Some Basic Feature Systems for English

- **Binary Features**

For example, the **INV** feature is a binary feature that indicates whether or not an S structure has an inverted subject (as in a yes/no question).

The S structure for the sentence *Jack laughed.* will have an INV value - ,

for the sentence *Did Jack laugh?* will have the INV value.

Often, the value is used as a prefix, and we would say that a structure has the feature +INV or -INV.

- **The Default Value for Features**

. Binary features: the default value -.

8.3. Morphological Analysis and the Lexicon

- Before you can specify a grammar, you must define the lexicon.
- The lexicon must contain information about all the different words that can be used, including all the relevant feature value restrictions. When a word is ambiguous, it may be described by multiple entries in the lexicon, one for each different use.

- The verb *want* may require six entries, for *want* (both in base and present form), *wants*, *wanting*, and *wanted* (both in past and past participle form).

8.3. Morphological Analysis and the Lexicon

```
(V ROOT ?r SUBCAT ?s VFORM pres AGR 3s) ->
(V ROOT ?r SUBCAT ?s VFORM base) (+S)
```

where +S is a new lexical category that contains only the suffix morpheme -s. This rule, coupled with the lexicon entry

```
want:
(V ROOT want
   SUBCAT {_np_vp:inf}
   VFORM base)
```

would produce the following constituent given the input string want -s

```
wants:
(V ROOT want
   SUBCAT {_np_vp:inf}
   VFORM pres
   AGR 3s)
```

8.3. Morphological Analysis and the Lexicon

Another rule would generate the constituents for the present tense form not in third person singular, which for most verbs is identical to the root form:

(V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** {1s 2s 1p 2p 3p})

—> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base)

Irregular verbs: * We be at the store.

A feature is introduced to identify irregular forms:

verbs with +IRREG-PRES have irregular present tense forms.

(V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** (1s 2s 1p 2p 3p))

—> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES** -)

Present Tense

1. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** 3s) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES** –) +S
2. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** {1s 2s 1p 2p 3p}) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES** –)

Past Tense

3. (V **ROOT** ?r **SUBCAT** ?s **VFORM** past **AGR** {1s 2s 3s 1p 2p 3p}) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PAST** –) +ED

Past Participle

4. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pastprt) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **EN-PASTPRT** –) +ED
5. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pastprt) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **EN-PASTPRT** +) +EN

Present Participle

6. (V **ROOT** ?r **SUBCAT** ?s **VFORM** ing) →
   (V **ROOT** ?r **SUBCAT** ?s **VFORM** base) +ING

Plural Nouns

7. (N **ROOT** ?r **AGR** 3p) →
   (N **ROOT** ?r **AGR** 3s **IRREG-PL** –) +S

Grammar 4.5 Some lexical rules for common suffixes on verbs and nouns

```
a:      (CAT ART
         ROOT A1
         AGR 3s)
be:     (CAT V
         ROOT BE1
         VFORM base
         IRREG-PRES +
         IRREG-PAST +
         SUBCAT {_adjp _np})
cry:    (CAT V
         ROOT CRY1
         VFORM base
         SUBCAT _none)
dog:    (CAT N
         ROOT DOG1
         AGR 3s)
fish:   (CAT N
         ROOT FISH1
         AGR {3s 3p}
         IRREG-PL +)
happy:  (CAT ADJ
         SUBCAT _vp:inf)
he:     (CAT PRO
         ROOT HE1
         AGR 3s)
is:     (CAT V
         ROOT BE1
         VFORM pres
         SUBCAT {_adjp _np}
         AGR 3s)
Jack:   (CAT NAME
         AGR 3s)
man:    (CAT N1
         ROOT MAN1
         AGR 3s)
men:    (CAT N
         ROOT MAN1
```

```
saw:    (CAT N
         ROOT SAW1
         AGR 3s)
saw:    (CAT V
         ROOT SAW2
         VFORM base
         SUBCAT _np)
saw:    (CAT V
         ROOT SEE1
         VFORM past
         SUBCAT _np)
see:    (CAT V
         ROOT SEE1
         VFORM base
         SUBCAT _np
         IRREG-PAST +
         EN-PASTPRT +)
seed:   (CAT N
         ROOT SEED1
         AGR 3s)
the:    (CAT ART
         ROOT THE1
         AGR {3s 3p})
to:     (CAT TO)
want:   (CAT V
         ROOT WANT1
         VFORM base
         SUBCAT {_np _vp:inf _np_vp:inf})
was:    (CAT V
         ROOT BE1
         VFORM past
         AGR {1s 3s}
         SUBCAT {_adjp _np})
were:   (CAT V
         ROOT BE
         VFORM past
         AGR {2s 1p 2p 3p}
```

8.3. Morphological Analysis and the Lexicon

The saw was broken.
Jack wanted me to saw the board in half.
I saw Jack eat the pizza.

With the lexicon in Figure 4.6 and Grammar 4.5, correct constituents for the following words can be derived: *been*, *being*, *cries*, *cried*, *crying*, *dogs*, *saws* (two interpretations), *sawed*, *sawing*, *seen*, *seeing*, *seeds*, *wants*, *wanting*, and *wanted*.

For example, the word *cries* would be transformed into the sequence *cry +s*, and then rule 1 would produce the present tense entry from the base form in the lexicon.

8.4. A Simple Grammar Using Features

- A simple grammar using the feature systems and lexicon developed earlier.

The man cries.
The men cry.
The man saw the dogs.
He wants the dog.
He wants to be happy.
He wants the man to see the dog.
He is happy to be a dog.
* The men cries.
* The man cry.
* The man saw to be happy.
* He wants.
* He wants the man saw the dog.

8.4. A Simple Grammar Using Features

Additional conventions in writing the grammar:

Many feature values are unique to a feature :
(VP SUBCAT inf) will be abbreviated as VP[inf].

For a binary feature B, the constituent C[+B] indicates the constituent (C B +).

Default values

Define clusters of features, and then indicate a cluster with a single symbol rather than listing them all.

The inheritance of features in a feature hierarchy.

8.4. A Simple Grammar Using Features

- Many features are constrained so that the value on the mother must be identical to the value on its **head subconstituent**. These are called **head features**.

```
(VP VFORM ?v AGR ?a) ->
     (V VFORM ?v AGR ?a SUBCAT _np_vp:inf)
     (NP)
     (VP VFORM inf)

If declared separately:
VP -> (V SUBCAT _np_vp:inf) NP (VP VFORM inf)

VP —> V [_np_vp:inf] NP VP[inf]
```

1. S[-inv] → (NP AGR ?a) (*VP* [{*pres past*}] ***AGR*** *?a*)
2. NP → (ART AGR ?a) (*N* ***AGR*** *?a*)
3. NP → *PRO*
4. VP → *V*[*_none*]
5. VP → *V*[*_np*] NP
6. VP → *V*[*_vp:inf*] VP[inf]
7. VP → *V*[*_np_vp:inf*] NP VP[inf]
8. VP → *V*[*_adjp*] ADJP
9. VP[inf] → *TO* VP[base]
10. ADJP → *ADJ*
11. ADJP → *ADJ* [*_vp:inf*] VP[inf]

Head features for S, VP: **VFORM**, **AGR**

Head features for NP: **AGR**

Grammar 4.7 A simple grammar in abbreviated form

1. (S INV – VFORM ?v{pres past} AGR ?a) →
   (NP AGR ?a) (*VP VFORM ?v{pres past} AGR ?a*)
2. (NP AGR ?a) → (ART AGR ?a) (*N AGR ?a*)
3. (NP AGR ?a) → (*PRO AGR ?a*)
4. (VP AGR ?a VFORM ?v) → (*V SUBCAT _none AGR ?a VFORM ?v*)
5. (VP AGR ?a VFORM ?v) → (*V SUBCAT _np AGR ?a VFORM ?v*) NP
6. (VP AGR ?a VFORM ?v) →
   (*V SUBCAT _vp:inf AGR ?a VFORM ?v*) (VP VFORM inf)
7. (VP AGR ?a VFORM ?v) →
   (*V SUBCAT _np_vp:inf AGR ?a VFORM ?v*) NP (VP VFORM inf)
8. (VP AGR ?a VFORM ?v) →
   (*V SUBCAT _adjp AGR ?a VFORM ?v*) ADJP
9. (VP SUBCAT inf AGR ?a VFORM inf) →
   (*TO AGR ?a VFORM inf*) (VP VFORM base)
10. ADJP → *ADJ*
11. ADJP → (*ADJ SUBCAT _inf*) (VP VFORM inf)

Grammar 4.8 The expanded grammar showing all features

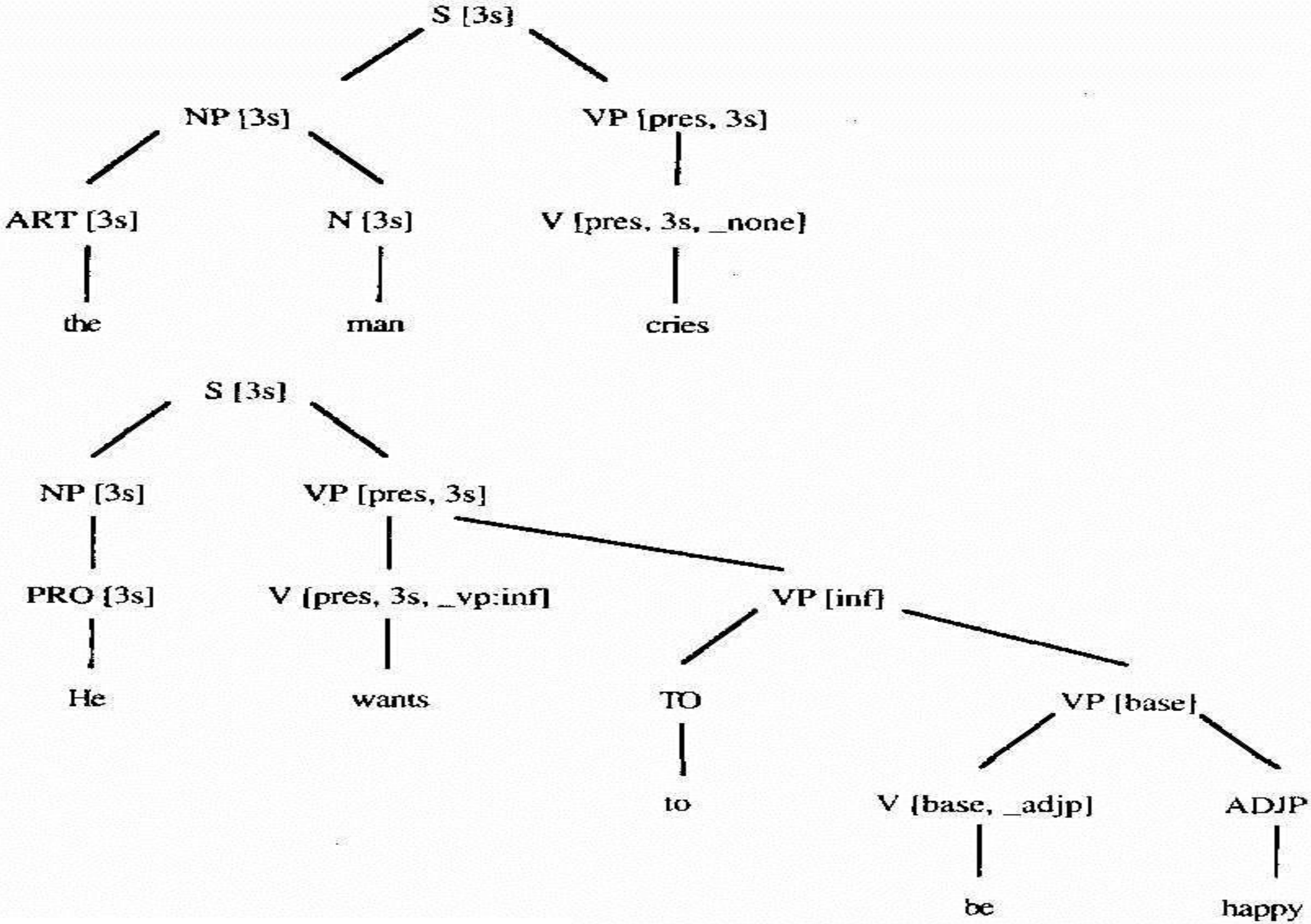


Figure 4.9 Two sample parse trees with feature values

8.5. Parsing with Features

```
C -> C1 ... Ci o X ... Cn

C -> C1 ... Ci X o ... Cn
```

A similar operation can be used for grammars with features, but the parser may have to instantiate variables in the original arc before it can be extended by X.

8.5. Parsing with Features

arc 1. (NP **AGR** ?a) -> o (ART **AGR** ?a) (N **AGR** ?a)

Consider extending arc 1 with the constituent 2:
Constituent 2. (**ART** **ROOT** A **AGR** 3s)

To make arc 1 applicable, the variable ?a must be instantiated to 3s, producing:
arc 3. (NP **AGR** 3s) -> o (**ART** **AGR** 3s) (N **AGR** 3s)
This arc can now be extended because every feature in the rule is in constituent 2:
arc 4. (NP **AGR** 3s) -> (**ART** **AGR** 3s) o (N **AGR** 3s)

Constituent 5. (N **ROOT** DOG1 **AGR** 3s)
arc 6. (NP **AGR** 3s) —> (**ART** **AGR** 3s) (N **AGR** 3s) o

8.5. Parsing with Features

- Algorithm

Given an arc A, where the constituent following the dot is called NEXT, and a new constituent X, which is being used to extend the arc,

a. Find an instantiation of the variables such that all the features specified in NEXT are found in X.

b. Create a new arc A', which is a copy of A except for the instantiations of the variables determined in step (a).

c. Update A' as usual in a chart parser.

For instance, let A be arc 1, and X be the ART constituent 2. Then NEXT will be (ART AGR ?a). In step a, NEXT is matched against X, and you find that ?a must be instantiated to 3s. In step b, a new copy of A is made, which is shown as arc 3. In step c, the arc is updated to produce the new arc shown as arc 4.

8.5. Parsing with Features

Consider extending arc 1 with the constituent:

`(ART` **`ROOT`** `the` **`AGR`** `?v{3s 3p})`

`(NP` **`AGR`** `?v{3s 3p})` —>

`(`**`ART`** **`AGR`** `?v{3s 3p}) o (N` **`AGR`** `?v{3s 3p})`

with `(N` **`ROOT`** `dog` **`AGR`** `3s):`

arc 6. `(NP` **`AGR`** `3s)` —>

`(`**`ART`** **`AGR`** `3s) (N` **`AGR`** `3s) o`

S1 CAT S
AGR 3s
VFORM pres
INV –
1 NP1
2 VP3

VP3 CAT VP
VFORM pres
AGR 3s
1 V1
2 VP2

VP2 CAT VP
VFORM inf
1 TO1
2 VP1

NP1 CAT NP
AGR 3s
1 PRO1

VP1 CAT VP
VFORM base
1 V2

PRO1 CAT PRO
AGR 3s

V1 CAT V
ROOT want
VFORM pres
AGR 3s
SUBCAT
{_np,_vp:inf,
_np_vp:inf}

TO1 CAT TO

V2 CAT V
ROOT cry
VFORM base
SUBCAT _none

He wants to cry

8.6. Augmented Transition Networks

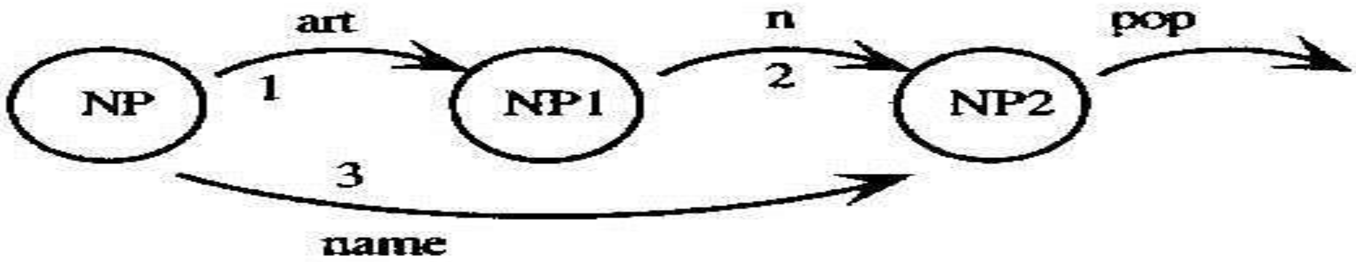
- Recursive transition network + Features:

    Augmented transition network (ATN)

Features in an ATN are traditionally called **registers**.

Constituent structures are created by allowing each network to have **a set of registers**.

Each time a new network is pushed, a new set of registers is created. As the network is traversed, these registers are set to values by **actions** associated with each arc. When the network is popped, the registers are assembled to form a constituent structure, with the **CAT slot being the network name**.



Arc	Test	Actions
1	none	DET := * AGR := AGR_*
2	$AGR \cap AGR_*$	HEAD := * AGR := $AGR \cap AGR_*$
3	none	NAME := * AGR := AGR_*

Grammar 4.11 A simple NP network

Arc	Test	Actions
4	none	SUBJ := *
5	$AGR_{SUBJ} \cap AGR_*$	MAIN-V := * AGR := $AGR_{SUBJ} \cap AGR_*$
6	none	OBJ := *

Grammar 4.12 A simple S network

Trace of S Network

Step	Node	Position	Arc Followed	Registers Set
1.	S	1	arc 4 succeeds (for recursive call see trace below)	SUBJ ← (NP DET the HEAD dog AGR 3s)
5.	S1	3	arc 5 (checks if $3p \cap 3p$)	MAIN-V ← saw AGR ← 3p
6.	S2	4	arc 6 (for recursive call trace, see below)	OBJ ← (NP NAME Jack AGR 3s)
9.	S3	5	pop arc succeeds	returns (S SUBJ (NP DET the HEAD dog AGR 3s) MAIN-V saw AGR 3p OBJ (NP NAME Jack AGR 3s))

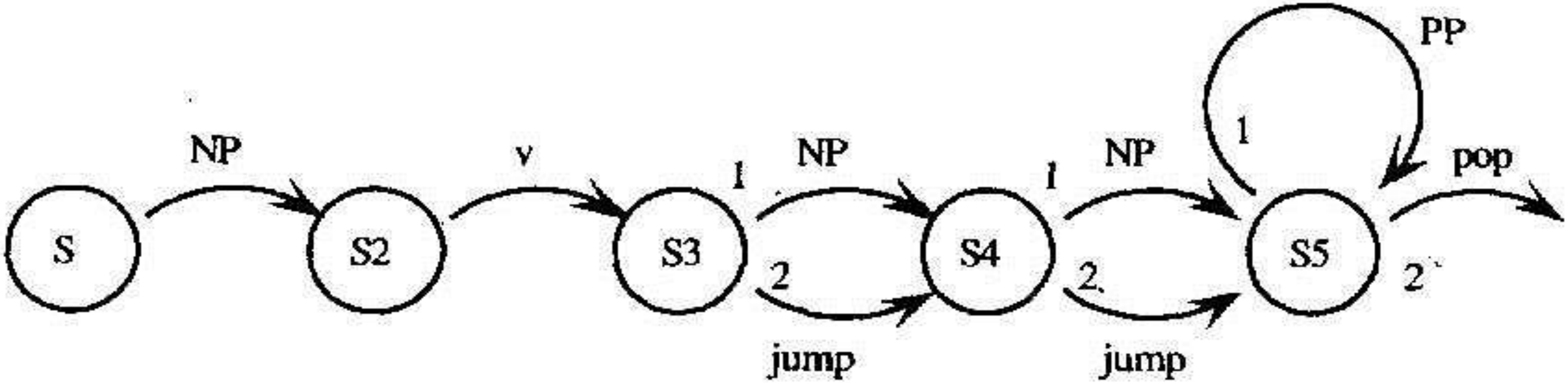
Trace of First NP Call: Arc 4

Step	Node	Position	Arc Followed	Registers Set
2.	NP	1	1	DET ← the AGR ← {3s 3p}
3.	NP1	2	2 (checks if $\{3s\ 3p\} \cap 3p$)	HEAD ← dog
4.	NP2	3	pop	returns (NP DET the HEAD dog AGR 3s)

Trace of Second NP Call: Arc 6

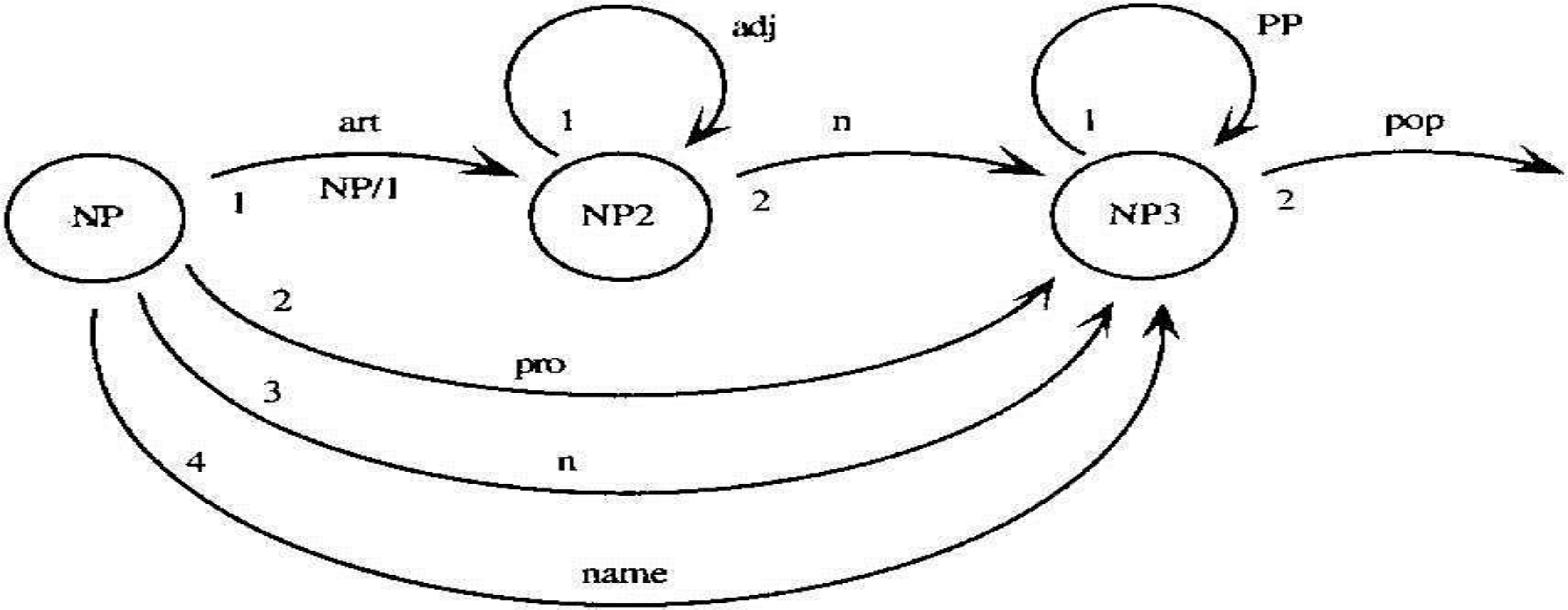
Step	Node	Position	Arc Followed	Registers Set
7.	NP	4	3	NAME ← John AGR ← 3s
8.	NP2	5	pop	returns (NP NAME John AGR 3s)

Figure 4.13 Trace tests and actions used with $_1$ *The* $_2$ *dog* $_3$ *saw* $_4$ *Jack* $_5$



Arc	Test	Actions
S/1		**SUBJ** := *
		MOOD := DECL
S2/1	$\mathbf{AGR}_{\mathbf{SUBJ}} \cap \mathbf{AGR}_{*}$	**MAIN-V** := *
S3/1	$\mathbf{SUBCAT}_{\mathbf{MAIN\text{-}V}} \cap$ {_np _np_np}	**OBJ** := *
		$\mathbf{SUBCAT} := \mathbf{SUBCAT}_{\mathbf{MAIN\text{-}V}} \cap$ {_np _np_np}
S3/2	$\mathbf{SUBCAT}_{\mathbf{MAIN\text{-}V}} \cap$ _none	**SUBCAT** := _none
S4/1	$\mathbf{SUBCAT}_{\mathbf{MAIN\text{-}V}} \cap$ _np_np	**IOBJ** := **OBJ**
		OBJ := *
S5/1	----------	**MODS** := Append(**MODS**, *)

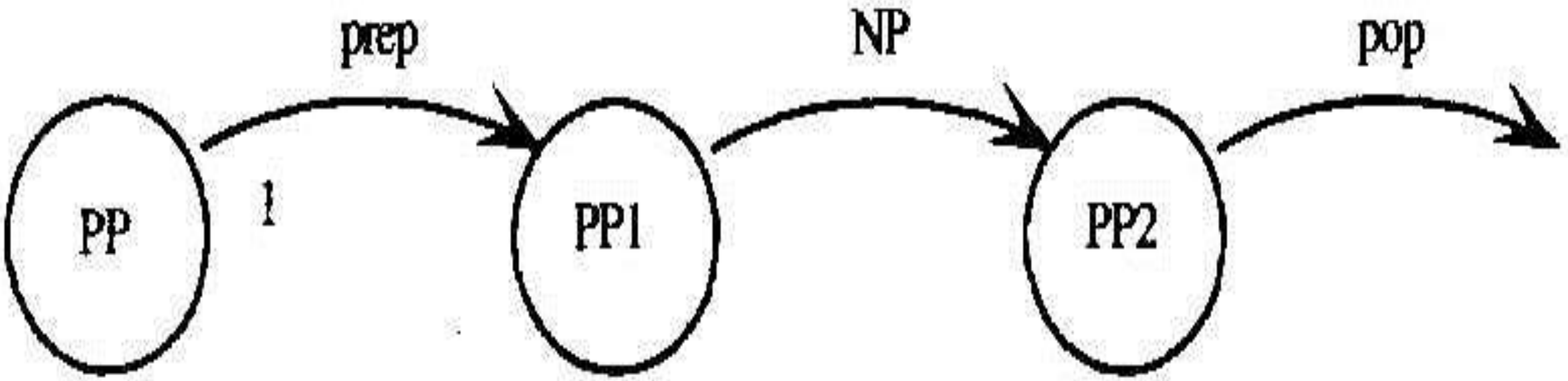
Grammar 4.14 An S network for assertions



Arc	Test	Actions
NP/1	----------	**DET** := *
NP/2	----------	**PRO** := *
NP/3	**AGR**$_*$ $\cap$ 3p	**HEAD** := * **AGR** := **AGR**$_*$
NP/4	----------	**NAME** := *
NP2/1	----------	**ADJS** := Append(**ADJS**, *)
NP2/2	**AGR** $\cap$ **AGR**$_*$	**HEAD** := * **AGR** := **AGR** $\cap$ **AGR**$_*$
NP3/1	----------	**MODS** := Append(**MODS**, *)

Grammar 4.15 The NP network

8.6. Augmented Transition Networks



Arc	Test	Action
PP/1	----------	P := *
PP1/1	----------	POBJ := *

Grammar 4.16 The PP network

思考题（不用交）

设计可接受汉语基本动词句型的ATN:

不带宾语 (她哭了)

带一个体词性宾语 (小王吃饺子)

带双宾语 (老师送我一支笔)

带一个谓词性宾语 (李四认为没事了)

带一个小句 (他认为敌人不再来了)

兼语句 (我叫他别来)

需要对应定义一个微型的lexicon. 给出能接受的例句集和不能被接受的例句集.

并且给出一个最能说明问题的例句的trace.

复杂的句子：漆黑的夜里，妻儿在等他回去想办法买米烧夜饭吃。