Chapter IV

Analysis of corpus

There are 50 sentences in the corpus of this study (see Appendix A). These sentences are selected to illustrate different linguistic phenomena, such as a relative clause, nominalized clause, syntactic paraphrase, lexical ambiguity, non-projective sentence, etc. All sentences in the corpus will be analyzed by CUPARSE into D-trees and conceptual networks, using rules and links based on manual analysis of the corpus sentences. This chapter describes in details the analysis of the corpus. It is divided into four sections. Three sections are the analysis for each module in CUPARSE: LD, DTC, and CCA. The last section describes the details of the output of analysis in the corpus.

4.1 LD analysis

The main function of LD analysis is to solve ambiguity at the lexical level. Since the number of lexical ambiguity in this study is quite small, LD analysis is not the main part of analysis in this study; therefore, only two rules are needed to analyze sentences in our corpus: NCompV and PrefV. Rule NCompV solves ambiguity of a wordform which can be a complementizer as well as other categories. Rule PrefV solves ambiguity of a wordform which can be a verb as well as other categories. The prefix "การ" or "ความ", if exists, helps determine that the wordform belongs to the verb category. Rule NCompV states that if a sequence of lexemes [N], [COMP], and [V] is found as three continuous segments, the lexemes in these segments which are not [N], [COMP], [V] respectively will be deleted. Rule PrefV states that if a sequence of lexemes [PREF] [V] is found at two continuous segments, the lexemes in these segments which are not [PREF] [V] will be deleted.

```
(1) a) Rule : NCompV
        if (-)=[N] and (*)=[COMP] and (+)=[V]
          then SELECT (-) which is [N]
               SELECT (*) which is [COMP]
               SELECT (+) which is [V]
        endif
    b) Rule : PrefV
        if (*)=[PREF] and (+)=[V]
          then SELECT (*) which is [PREF]
          then SELECT (+) which is [V]
        endif
```

4.2 DTC analysis

The main function of DTC analysis is to construct a D-tree from a sequence of lexemes and assign syntactic cases. To construct a D-tree from an input sentence, the priorities of relation are used as the basis to determine the order of relation to be constructed. These priorities are based on manual analysis of the corpus sentences, which is also used as a basis to postulate rules in various modules of CUPARSE. This manual analysis is presented in 4.2.1. This is followed by postulation of rules in 4.2.2. The grouping of rules in various phases is in 4.2.3. Ordering of links is discussed in 4.2.4.

4.2.1 D-tree analysis of corpus

Before any priorities of rules can be postulated, adequate linguistic information is required for the structure and dependency relation of the 50 sentences in our corpus. This information is derived from a detailed manual analysis of the sentence. Each sentence will be analyzed into a D-tree, such as the sentence "เขา ใช้ คอมพิวเตอร์ ที่ ห้องแล็บ" is analyzed into a D-tree as follows:

```
(2) [                ใช้               ]
       |           |            |
    (SUBR) (     FOBR      ) ( LATPR  )
       |           |            |
    [เขา ] [ คอมพิวเตอร์ ] [ ห้องแล็ป ]
```

Manually analyzed D-trees of the corpus are on Appendix B.

4.2.2 <u>Postulation of rules</u>

This section is divided into two topics: rules for syntactic cases and priorities of rules.

4.2.2.1 <u>Rules for syntactic cases</u>

Syntactic cases are assigned from category sequence of adjacent lexemes in a sentence. For example, syntactic case NUMR is assigned when the sequence [CRDN][CLSS] is found. This sequence is considered as the condition of rule NumR for dependency construction between noun and classifier. Each rule consists of condition and action commands used for syntactic case construction. The following is the example of rule NumR. (For the entire list and formulation of rules, see Appendix D)

```
(3) Rule: NumR
    if (*)=[CRDN] and (+)=[CLSS]
    then allocate node n1
         link (*) to be left depender of n1
         link n1 to be left depender of (+)
         assign [NUMR] to n1
         design (*) to be [N]; design (+) to be [N]
    endif
```

A rule can be used for one or more syntactic cases. For example, rule NumR, which considers the condition [CRDN][CLSS], is used for only syntactic case NUMR. Rule VRaux, which considers the condition [V][RAUX], is used for both syntactic cases RATTR and RASPR. In addition, a syntactic case can be assigned from more than one condition, such as syntactic case COMPR which is assigned from the condition [N][COMP][V] as well as [N][ADJ]. In this case, more than one rule is required. These are rules NCompV and NAdj. The list of rules used for syntactic cases and their conditions are as follows:

(4)	Rule	Condition	Syntactic cases
	TopSubR	[N][V]	TOPR, SUBR
	FobSobR	[V][N]	FOBR
	ClssR	[N][CLSS]	CLSSR
	NumR	[CRDN][CLSS]	NUMR
	LdetR	[LDET][N]	LDETR
	RdetR	[N][RDET]	RDETR
	LauxV	[LAUX][V]	LMDR, LASPR, LTNSR, LATTR
	VRaux	[V][RAUX]	RATTR, RASPR
	AdjR	[V][VADJ]	ADJR
	NCompV	[N][COMP][V]	COMPR
	NAdj	[N][VADJ]	COMPR
	NpnN	[N][PREPN][N]	POSSPR
	NpN	[N][PREP][N]	LINPR
	VpvN	[V][PREPV][N]	MWITHPR, BENPR, COMPPR
	VpN	[V][PREP][N]	LATPR, LINPR, RFROMPR, RTOPR, LONPR, LFRTPR, ABOUTPR
	pNV	[PREP][N][V]	LINPR

In the corpus, these rules cannot readily apply to sentence 33. This sentence is non-projective (see 2.2.1, 3.5) An additional rule is then needed to solve this non-projective problem before other rules can apply. Rule NClssDist in (5) is used to adjust the sequence of lexemes for this purpose. It checks whether or not the classifier, which is adjacent and to the right of the noun, belongs to the noun. If it does not belong to that noun, a new noun will be searched so that the classifier can be moved to connect with this new noun.

(5) a) S33: บริษัท บริจาค คอมพิวเตอร์ แก่ มหาวิทยาลัย 10 เครื่อง

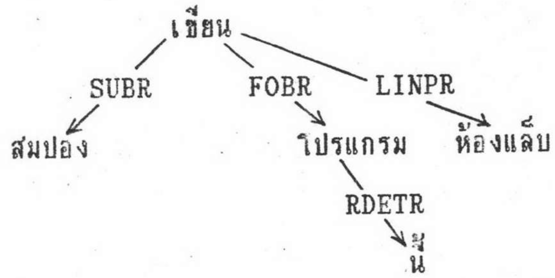
b) Rule: NClssDist

```
if  (*)=[N] (+)=[CLSS] and *.CLSSG <> +.MORPH then
   if LSEARCH for X that X.CLSSG = +.MORPH then
      MOVE (+) to connect to X endif
endif
```

4.2.2.2 <u>Priorities of rules</u>

The priorities as described in 2.2.4 are used as the basis of dependency construction. A relation which has higher priority should be constructed before that with the lower priority. Since a relation is constructed by rules, the priorities concerned are priorities of rules. The priorities of rules are obtained directly from the desired output D-trees. For example, the D-tree of sentence 21 is shown as (6a).

(6) a) S21: สมปอง เขียน โปรแกรม นี้ ใน ห้องแล็บ



The priorities between syntactic cases in this sentence obtained from this D-tree is shown in (6b). There is only one PPT, FOBR > LINPR because the relations N<-prep-N, RDET->N, and RDET<-N are not possible. Priorities of syntactic cases in (6b) are changed to be priorities of rule as shown in (6c). Example (7) shows another D-tree and priorities of rules obtained from sentence 26.

(6) b) IPT : FOBR > LINPR

BPT : RDETR > FOBR

PPT : FOBR > [N<-prep->N] (โปรแกรม ใน ห้องแล็บ) => LINPR

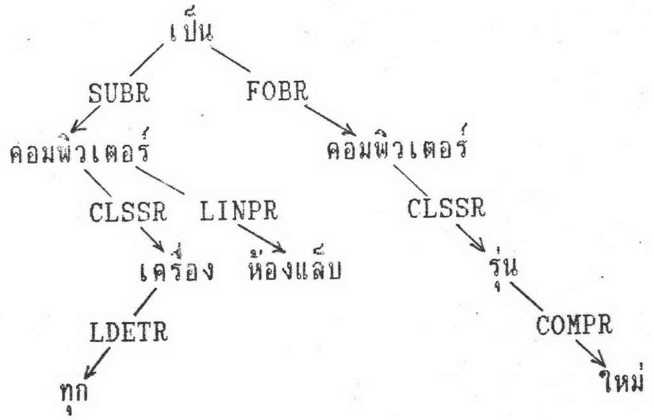
RDET > [RDET<->N] (นี้ ห้องแล็บ)

c) IPT : FobSobR > VpN

BPT : RdetR > FobSobR

PPT : FobSobR > NpN

(7) a) S26: คอมพิวเตอร์ ทุก เครื่อง ใน ห้องแล็บ เป็น คอมพิวเตอร์ รุ่น ใหม่



b) IPT : CLSSR > LINPR

BPT : LINPR > SUBR

LDETR > CLSSR > SUBR

COMPR > CLSSR > FOBR

PPT : LINPR > [N<->VEQU] (ห้องแล็บ เป็น) => SUBR

CLSSR > [N<-prep->N] (เครื่อง ใน ห้องแล็บ) => LINPR

LDETR > [N<->LDET] (คอมพิวเตอร์ ทุก)

c) IPT : ClssR > NpN

BPT : NpN > TopSubR

LdetR > ClssR > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

Priorities of rules extracted from the study of D-trees of the 50 sentences are listed as follows:

(8) S2: BPT : FobSobR > NCompV > FobSobR

S3: IPT : LauxV > TopSubR

S4: IPT : LauxV > LauxV > TopSubR

S5: BPT : FobSobR > NCompV > TopSubR

S6: IPT : FobSobR > VpN

PPT : FobSobR > NpN

S7: BPT : AdjR > NCompV > FobSobR

S8: BPT : NumR > FobSobR

RdetR > ClssR > TopSubR

PPT : ClssR > TopSubR

S9: BPT : RdetR > ClssR > TopSubR

PPT : ClssR > TopSubR

S10: BPT : RdetR > ClssR > NpnN > TopSubR

PPT : NpnN > TopSubR

NpnN > NAdj

ClssR > TopSubR

ClssR > NAdj

S11: BPT : TopSubR > NCompV > TopSubR

RdetR > TopSubR

S12: BPT : RdetR > TopSubR

S13: IPT : FobSobR > VRaux

BPT : RdetR > FobSobR

S14: IPT : FobSobR > VpvN

BPT : RdetR > FobSobR

S15: IPT : NCompV > RdetR

BPT : TopSubR > NCompV > TopSubR

RdetR > TopSubR

S16: IPT : FobSobR > VpvN

S17: IPT : FobSobR > VpvN

S18: BPT : RdetR > TopSubR

S19: IPT : LauxV > TopSubR

FobSobR > VpN

PPT : FobSobR > NpN

S20: IPT : LauxV > TopSubR > pNV

S21: IPT : FobSobR > VpN

BPT : RdetR > FobSobR

PPT : FobSobR > NpN

S22: IPT : FobSobR > VpN > VpN

BPT : RdetR > FobSobR

PPT : FobSobR > NpN

VpN > NpN

S23: BPT : NpN > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

NpN > pNV

S24: BPT : TopSubR > NCompV > TopSubR

NAdj > ClssR > FobSobR

S25: BPT : TopSubR > NCompV > ClssR > TopSubR

NAdj > ClssR > FobSobR

PPT : ClssR > TopSubR

S26: IPT : ClssR > NpN

BPT : NpN > TopSubR

LdetR > ClssR > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

S27: IPT : ClssR > NpN

BPT : NpN > TopSubR

LdetR > NumR > ClssR > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

S28: IPT : ClssR > NCompV

BPT : TopSubR > NCompV > TopSubR

NAdj > ClssR > FobSobR

LdetR > NumR > ClssR > TopSubR

PPT : ClssR > NCompV

S29: IPT : ClssR > NpN

BPT : NpN > TopSubR

NAdj > ClssR > FobSobR

LdetR > NumR > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

S30: IPT : ClssR > NpN

BPT : NpN > TopSubR

LdetR > ClssR > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

S31: IPT : ClssR > NpN

BPT : LdetR > ClssR > TopSubR

NpN > TopSubR

NAdj > ClssR > FobSobR

PPT : NpN > TopSubR

ClssR > NpN

S32: BPT : NAdj > ClssR > FobSobR

RdetR > ClssR > TopSubR

PPT : ClssR > TopSubR

S33: IPT : FobSobR > VpvN

BPT : NumR > ClssR > FobSobR

S34: IPT : FobSobR > VpvN

BPT : NumR > ClssR > FobSobR

S35: IPT : TopSubR > TopSubR

BPT : RdetR > ClssR > TopSubR

LdetR > NumR > ClssR > TopSubR

S36: IPT : LauxV > LauxV >TopSubR

BPT : NCompV > VpN

PPT : FobSobR > NpN

S37: IPT : FobSobR > VpN

BPT : VpN > NCompV > VpN

PPT : FobSobR > NpN

S38: IPT : LauxV > FobSobR

BPT : NCompV > TopSubR

RdetR > TopSubR

S39: IPT : LauxV > LauxV > TopSubR

BPT : AdjR > NCompV > FobSobR

NCompV > FobSobR

S40: BPT : RdetR > ClssR > VpvN > AdjR

RdetR > ClssR > TopSubR

PPT : ClssR > TopSubR

S41: IPT : VRaux > AdjR

BPT : VpvN > AdjR

S42: IPT : FobSobR > VpvN

BPT : FobSobR > VpvN

NpnN > FobSobR

S43: BPT : FobSobR > VpvN

PPT : NpnN > TopSubR

S44: IPT : LauxV > pNV

BPT : NpnN > FobSobR

FobSobR > pNV

PPT : FobSobR > TopSubR

S45: IPT : LauxV > TopSubR

BPT : FobSobR > TopSubR

NpnN > FobSobR

PPT : FobSobR > TopSubR

S46: IPT : TopSubR > pNV

BPT : NpnN > TopSubR

PPT : NpnN > TopSubR

NpnN > NAdj

S47: IPT : LauxV > LauxV > TopSubR

BPT : NCompV > FobSobR

NpnN > FobSobR

S48: IPT : LauxV > TopSubR

FobSobR > VRaux

S49: IPT : FobSobR > VRaux > VRaux

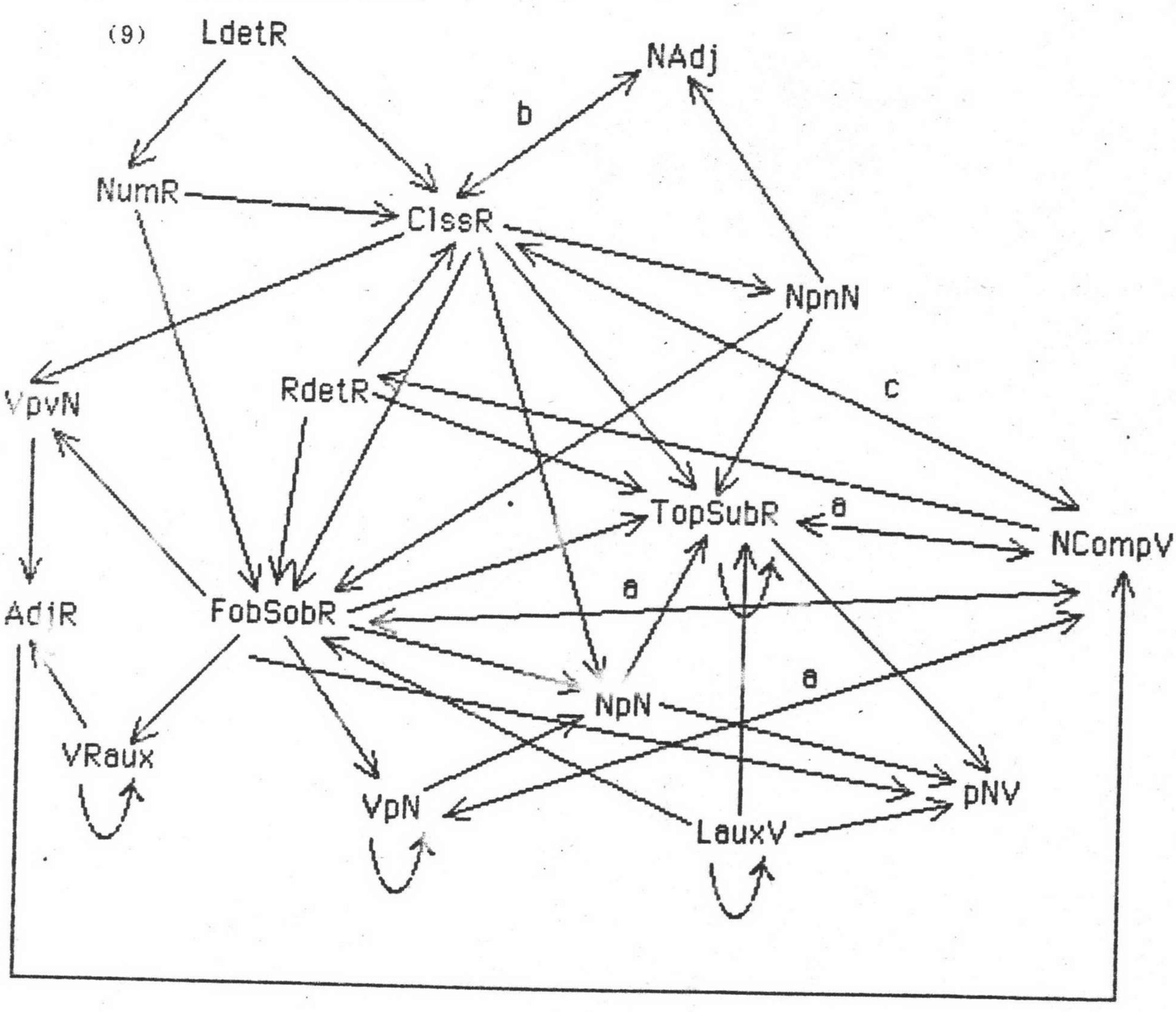
S50: IPT : VRaux > VRaux

BPT : RdetR > ClssR > TopSubR

PPT : ClssR > TopSubR

These priorities of rules can be represented as the priority graph in (9).

(9)



This priority graph shows three different conflicts, listed in (10). These conflicts are bidirectional.

(10) a) TopSubR > NCompV and NCompV > TopSubR

FobSobR > NCompV and NCompV > FobSobR

VpN > NCompV and NCompV > VpN

b) NAdj > ClssR and ClssR > NAdj

c) ClssR > NCompV and NCompV > ClssR

The conflict (10a) results from the relative clause. It is found as the junction between the relative clause and the main clause. This conflict can be resolved by separating the analysis process of the relative clause from the main clause. This solution denotes that the embedded clause should be constructed before the main clause; therefore, not only relative clauses but also nominalized clauses should be treated as separate processes.

The conflict (10b) results from PPT of sentence 10. It can be resolved by adding more conditions for the rule NAdj, as shown in (11), that the construction N->VADJ is done if and only if there still is at least one verb, which has no other category value, to the left or to the right of the head noun. This solution leads to the deletion of priority ClssR > NAdj in sentence 10, because the relation เครื่อง->แพง does not match this rule, since there is no verb left to the left or the right of เครื่อง.

S10: ราคา ของ เครื่องพิมพ์ เครื่อง นี้ แพง

(11) Rule: NAdj

```
if (*)=[N] (+)=[VADJ] and
    if LSEARCH for X that X equal to [V] or
       RSEARCH for X that X equal to [V]
     then construct N-COMPR->VADJ
endif
```

The conflict (10c) results from BPT of sentence 25 and PPT of sentence 28. This conflict can be resolved by placing rule ClssR in two positions, before and after rule NCompV and adding the condition to rule ClssR that the relation N->CLSS is constructed only when CLSS is a complete noun. The CLSS is a complete noun when it has some lexemes, such as CRDN, RDET, or a relative clause, as its modifier. The first rule of ClssR is used for the priority ClssR > NCompV, while the second is used for the priority NCompV > ClssR. As a consequence of this solution, rule ClssR applies before NCompV to sentence 28 because "เครื่อง" is modified by "ทั้ง 10". Rule NCompV applies before ClssR to sentence 25 because at the time the first ClssR is found, "เครื่อง" is not a complete noun; therefore, the first rule of ClssR does not apply. After rule NCompV applies, "เครื่อง" becomes a complete noun modified by a relative clause. The second rule of ClssR will then apply as shown in (12).

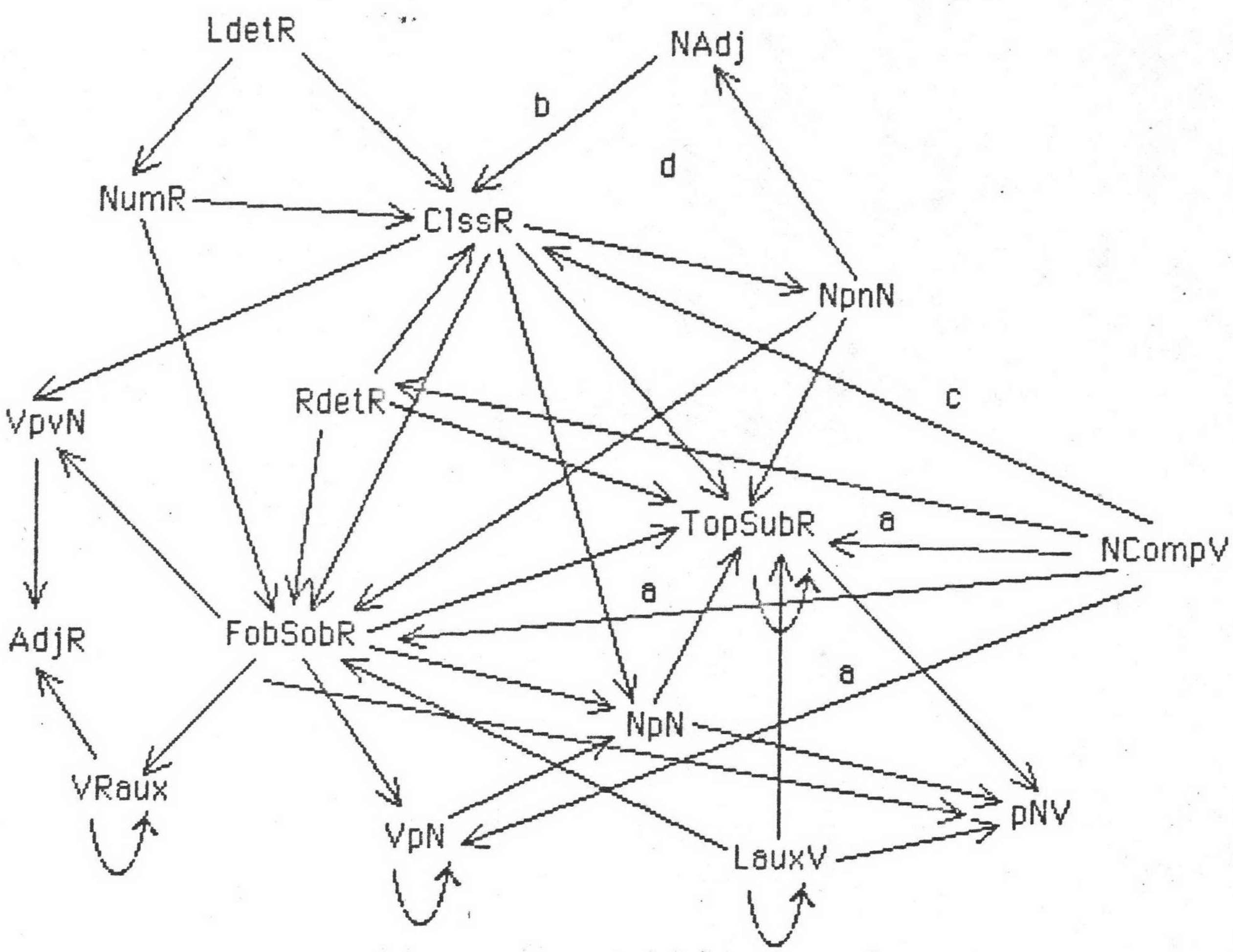
S25: คอมพิวเตอร์ เครื่อง ที่ เขา ใช้ เป็น คอมพิวเตอร์ รุ่น ใหม่

S28: คอมพิวเตอร์ ทั้ง 10 เครื่อง ที่ เขา ซื้อ เป็น คอมพิวเตอร์ รุ่น ใหม่

(12)	CLSSR	S28 applied	
	NCompV	S28 applied	S25 applied
	CLSSR		S25 applied

The new priority graph after all bidirectional conflicts are resolved is shown in (13). In this graph, the priority AdjR > NCompV is removed because it is the priority in a relative clause. This graph represents only the priorities for the main clause phase. However, from this graph, a new conflict, which is a circular conflict, is found. These are NAdj > ClssR, ClssR > NpnN, and NpnN > NAdj.

(13)

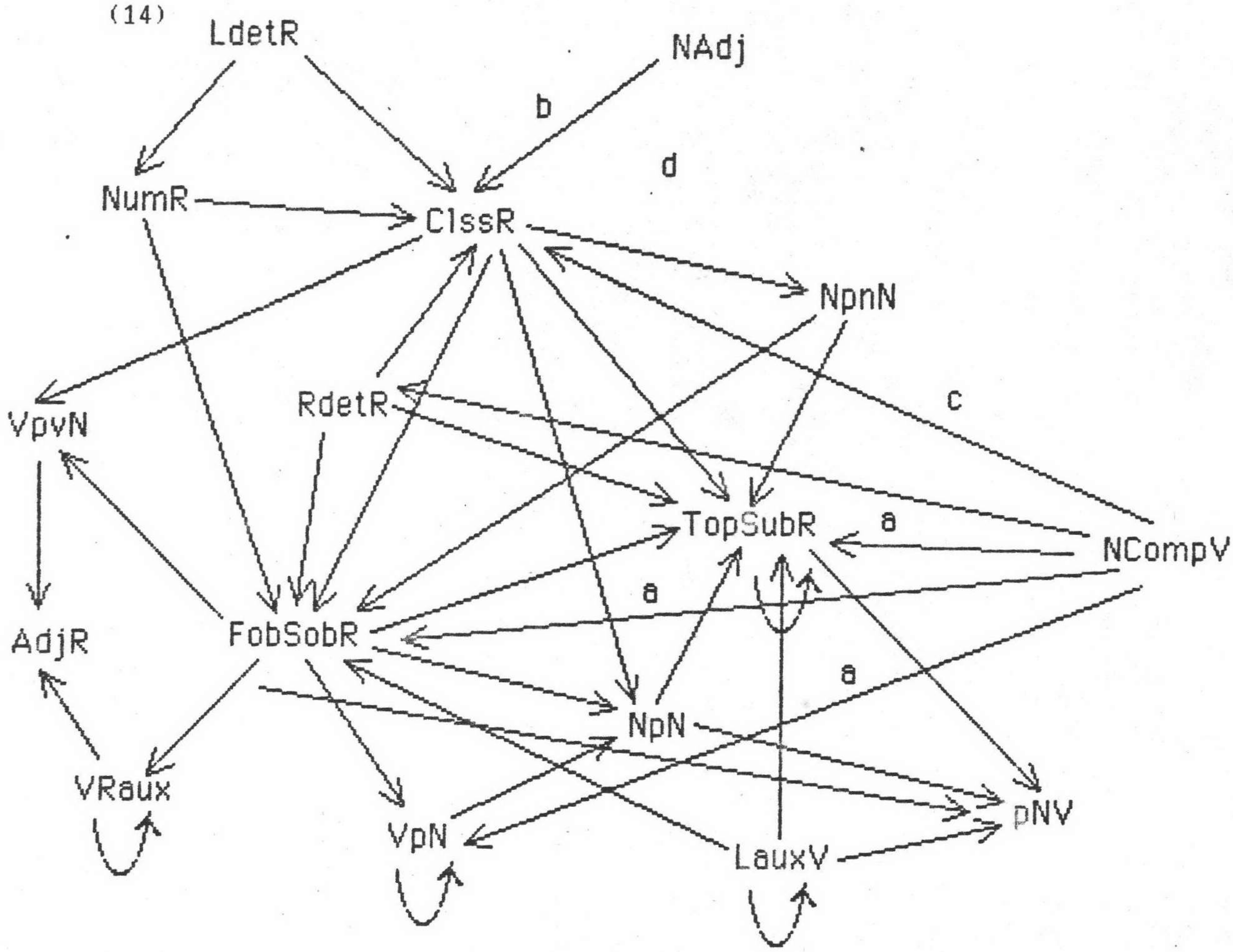


To solve conflict (13d), the circular link must be broken by removing one of the three possible priorities. In this case the priority NpnN > NAdj, which results from PPT of sentence 10 and 46, can be deleted, because rule NAdj does not apply when there is no verb left to the left and the right of noun (see (11)). The relations "เครื่องพิมพ์-COMPR->แพง" and "คอมพิวเตอร์-COMPR->ถูก" in sentences 10 and 46 respectively, cannot be a candidate of construction anymore because there is no verb left, and "ราคา" is decided to be a noun in these sentences.

S10: ราคา ของ เครื่องพิมพ์ เครื่อง นี้ แพง

S46: ใน ขณะนี้ ราคา ของ คอมพิวเตอร์ ถูก ลง

(14)



4.2.3 Links and phases

Rules are organized into links and phases. The former is a strategy to obtain efficiency of the system. The latter is linguistically motivated. Each phase analyzes a different linguistic phenomenon.

This section is divided into two parts. The concept of grouping rules into links is described first. The analysis phases are described later.

4.2.3.1 Grouping rules into links

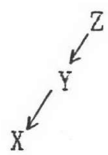
To obtain efficiency of the system, rules are organized into links. There are five criteria to determine which rules can be placed in the same link.

1. No priority link.

If no priority exists between rules A and B, they can be placed in the same link.

2. BPT with left depender.

If BPT exists between rules A and B, rules A and B can be placed in the same link if and only if rules A and B are used for relations X<-Y and Y<-Z respectively, and the subtree in (15a) is constructed from the lexeme sequence ...X...Y...Z. Since the window scope scans the lexeme sequence from left to right, rule A is applied before rule B as shown in (15c) because the windows bind lexemes X Y before Y Z.

(15) a) 

b) rule A : X<-Y

rule B : Y<-Z

c) .... X...Y...Z.... => rule A is applied

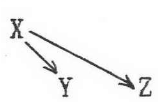
[l-*+r]

.........Y...Z.... => rule B is applied

[l-*+r]

3. IPT with right depender.

If IPT exists between rules A and B, rules A and B can be placed in the same link if and only if rules A and B are used for relations X->Y and X->Z respectively, and the subtree in (16a) is constructed from the lexeme sequence ...X...Y...Z. Since the window scope scans the lexeme sequence from left to right, rule A is applied before rule B as shown in (16c) because the windows bind lexemes X Y before Y Z.

(16) a) 

b) rule A : X->Y

rule B : X->Z

c) .....X..Y...Z..... => rule A is applied

[l-*+r]

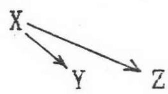
.....X....Z....... => rule B is applied

[l-*+r]

d. PPT in All Right Chain (ARC).

If PPT, which is obtained from ARC construction, exists between rules A and B, rule A and B can be placed in the same link if and only if rule A is used for relation X->Y and rule B is used for [Y<->Z], and the subtree in (17a) is constructed from the lexeme sequence ....X...Y...Z. This is because rule A is to be applied before rule B, as shown in (17c).

(17) a)



b) rule A : X->Y

rule B : [Y<->Z]

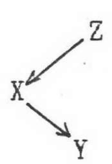
c) ....X...Y...Z => rule A is applied before rule B

[l-*+r]

e. PPT in Left-Right Chain (LRC).

If PPT, which is obtained from LRC construction, exists between rules A and B, rule A and B can be placed in the same link if and only if rule A is used for relation X->Y and rule B is used for [Y<->Z], and the subtree in (18a) is constructed from the lexeme sequence ...X...Y...Z. This is because rule A is to be applied before rule B, as shown in (17c).

(18) a)



b) rule A : X->Y

rule B : [Y<->Z]

c) ....X...Y...Z => rule A is applied before rule B

[l-*+r]

In addition to the above criteria, the grouping of rules into links may require an additional rule to facilitate the repetition of certain rules which construct relations between a head and its left dependers. These are rules LAuxV and TopSubR. The example (19a) shows that the rule LAUXV matches the second LAUX before the first LAUX. The first LAUX will lose an opportunity to match the rule LAUX unless the window scope is shifted to the left most lexeme as shown in (19b) and (19c); therefore, in the link which contains the rule having left repeating relations, an additional rule containing "shift(B)" command needs to be placed after those rules. The repetition also occurs in the right relations, such as VRaux, VpN; however, no extra rule is needed because the window scope already scans from left to right.

(19) a) .... LAUX LAUX V...

[l-*+r]

b) ....LAUX V......

[l-*+r]

c) ....LAUX V.......

[l-*+r]

4.2.3.2 Rule phases

As pointed out earlier, rule ordering can also be linguistically motivated. Embedding is a natural linguistic phenomenon. In the parsing process, embedded clauses need to be analyzed to yield embedded sub-structures of the D-tree before D-tree of the entire clause can be constructed (see conflict 10a in 4.2.2.2); therefore, rules are organized into two major phases: embedded phase and main clause phase. The embedding phase can be divided into two phases: relative clause phase and nominalized clause phase.

1. Relative clause phase

A relative Clause is an embedded clause that begins with the marker "ที่". From the 50 sentences, four syntactic cases, SUBR, FOBR, ADJR and LFRTPR, are found in relative clauses. These syntactic relations are constructed by rules TopSubR, FobSobR, AdjR and VpN respectively. No priority exists among these rules. They, therefore, can be placed in the same link which is the only one link contained in this phase. This phase begins when the prefix "ที่" is found, and ends when the right boundary of a relative clause, which is LAUX, RDET, VCMN, VEQU or #, is found. The link used for this phase is shown in (21).

```
(21)  Link : LRelC
      if [COMP] is found then
        do loop
          if (*)=[V] (+)=[N] then apply rule FobSobR
          if (*)=[N] (+)=[V] then apply rule TopSubR
          if (*)=[V] (+)=[VADJ] then apply rule AdjR
          if (*)=[V] (+)=[PREP] (r)=[N] then apply rule VpN
          if (+)=[LAUX] or [RDET] or [VCMN] or [VEQU] or # then
                                      apply rule NCompV and exit
        endloop
      endif
```

2. Nominalized clause phase

A nominalized Clause is an embedded clause that begins with the prefix "การ". In the corpus, only one syntactic case, FOBR, is found in all nominalized clauses. This syntactic case is constructed by rule FobSobR. This phase begins when the prefix "การ" is found, and ends when the right boundary of a nominalized clause, which is either LAUX or # in this study, is found. Only one link is needed for this phase. This link, LNomC, is shown in (20).

```
(20)  Link : LNomC
        if "การ" is found then
          do loop
            if (*)=[V] (+)=[N] then apply rule FobSobR
            if (+)=[LAUX] or (+)= # then apply rule PrefV and exit
          endloop
        endif
```

Note that the prefix "ความ" is not a nominalized clause marker. It is used as a nominal prefix for VADJ.

3. Main clause phase

Main clause phase is more complex than embedding phase discussed above, since it consists of many links, each of which contains a number of rules. According to the priority graph (14) discussed in 4.2.2.2, rules can be ordered as a linear sequence as shown in (22).

(22)	LauxV	=>	LdetR	=>	NumR	=>	RdetR	=>
	NAdj	=>	ClssR	=>	NpnN	=>	FobSobR	=>
	VpvN	=>	VpN	=>	NpN	=>	RAux	=>
	AdjR	=>	TopSubR	=>	pNV			

These rules are then organized into seven links, based on the five criteria discussed in 4.2.3.1, as shown in (23).

(23) Link LLmod : LauxV, LdetR, NumR

Link LRmod1 : NAdj, RdetR

Link LRmod2 : ClssR

Link LNpnN : NpnN

Link LObject: FobSobR

Link LPrep : VpvN, VpN, NpN

Link LMain : VRaux, AdjR, TopSubR, pNV

4.2.4 <u>Link order</u>

There are seven links in the main clause phase, one link in nominalized clause phase, and one link in relative clause phase. In addition, there is rule NClssDist which is used for adjusting the non-projective sentences as described in 4.2.2.1. This rule forms a separate link, LAdjust. Altogether, there are ten links in DTC. These links need to be ordered. The order used is as follows.

(24)	LLmod	=>	LAdjust	=>	LNomC	=>	LRmod1	=>
	LRmod2	=>	LRelC	=>	LRmod1	=>	LRmod2	=>
	LNpnN	=>	LObject	=>	LPrep	=>	Lmain	

Left modification is common in Thai and it can apply in all phases, so it is placed first in the link order. Link LAdjust needs to be applied before the links to analyze the nominalized clause, relative clause, and main clause phases; therefore, it is placed second. Links LNomC and LRelC analyze embedded clauses so they should immediately follow LAdjust. However, rule ClssR must be applied before and after rule NCompV (see conflict 10c in 4.2.2.2); therefore, link LRmod2 must be placed before and after LRelC. However, link LRmod1 must be applied before LRmod2; therefore, both LRmod1 and LRmod2 must be placed before and after link LRelC. All these factors leads to the link order for all phases as presented in (24) above.

4.3 CCA Analysis

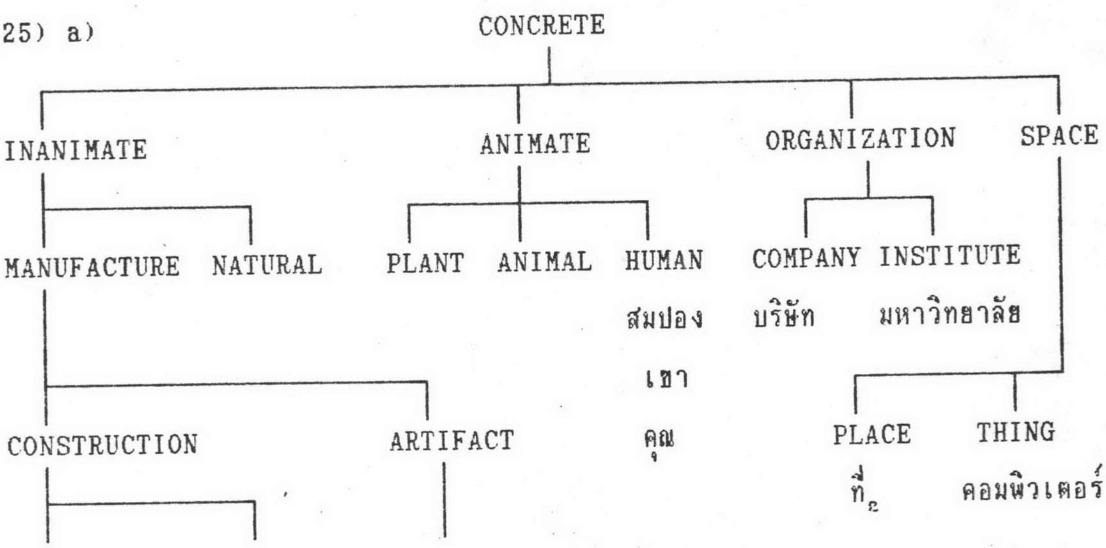
The main task of CCA is to convert syntactic cases in a D-tree into conceptual cases in a conceptual network. Following is presentation of the conceptual hierarchy, conceptual case constraints and case mapping proposed for the analysis of sentences in the corpus. These are grouped under the heading components of CCA analysis in 4.3.1. This is then followed by the analysis phase in CCA in 4.3.2.

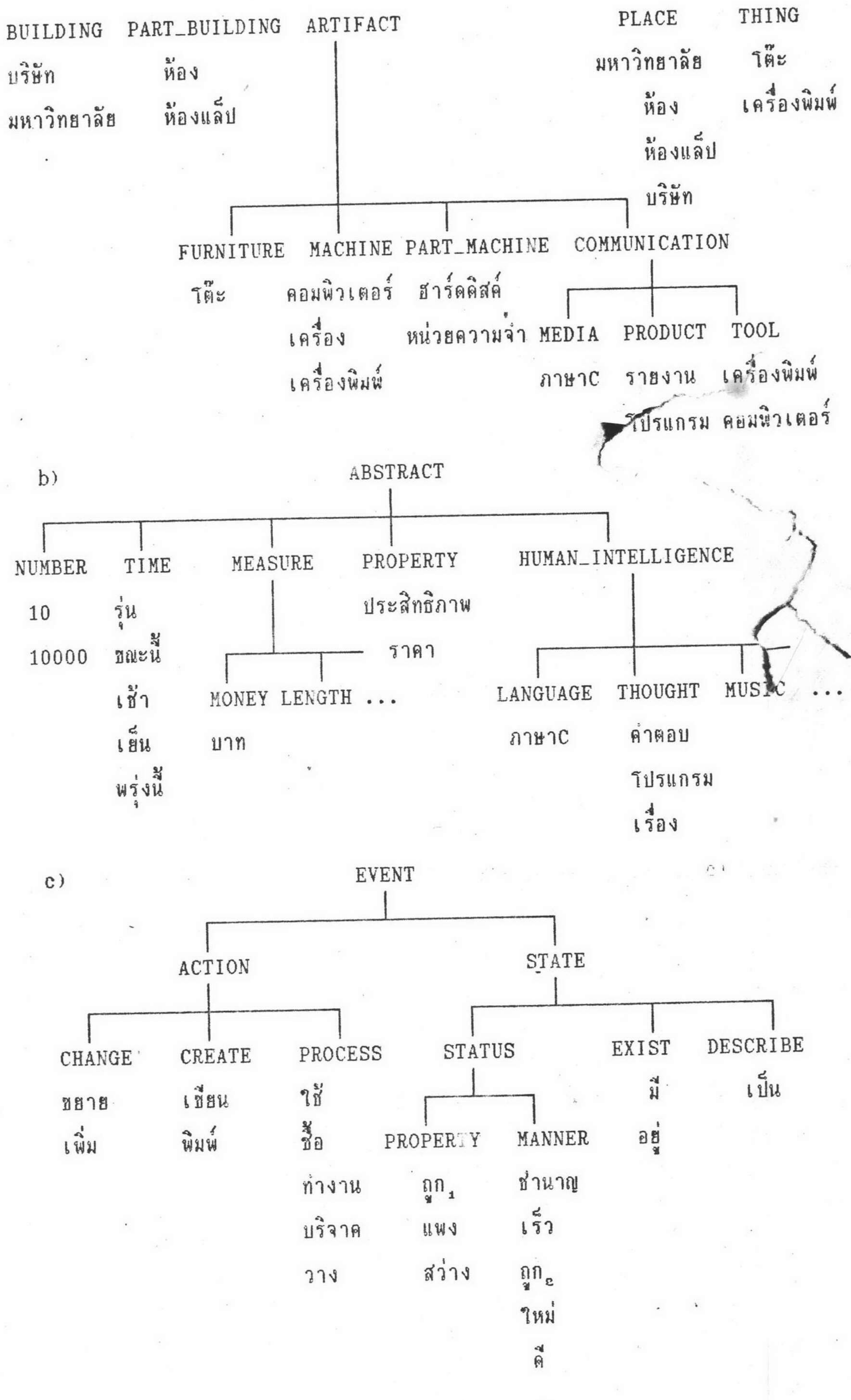
4.3.1 Components of CCA analysis

4.3.1.1 Conceptual hierarchy

There are 74 wordforms in our corpus. Of these, 10 are function words, 63 are content words, and one is both function word and content word. To represent meaning of these content words, 11 conceptual attributes and 53 word concepts have been proposed. These word concepts are organized into conceptual hierarchies as follows.

(25) a)




BUILDING
บริษัท
มหาวิทยาลัย
PART_BUILDING
ห้อง
ห้องแล็ป
ARTIFACT
PLACE
มหาวิทยาลัย
ห้อง
ห้องแล็ป
บริษัท
THING
โต๊ะ
เครื่องพิมพ์
FURNITURE
โต๊ะ
MACHINE
คอมพิวเตอร์
เครื่อง
เครื่องพิมพ์
PART_MACHINE
ฮาร์ดดิสค์
หน่วยความจำ
COMMUNICATION
MEDIA
ภาษาC
PRODUCT
รายงาน
โปรแกรม
TOOL
เครื่องพิมพ์
คอมพิวเตอร์
b)
ABSTRACT
NUMBER
10
10000
TIME
รุ่น
ขณะนี้
เช้า
เย็น
พรุ่งนี้
MEASURE
MONEY
บาท
LENGTH
...
PROPERTY
ประสิทธิภาพ
ราคา
HUMAN_INTELLIGENCE
LANGUAGE
ภาษาC
THOUGHT
คำตอบ
โปรแกรม
เรื่อง
MUSIC
...
c)
EVENT
ACTION
CHANGE
ขยาย
เพิ่ม
CREATE
เขียน
พิมพ์
PROCESS
ใช้
ซื้อ
ทำงาน
บริจาค
วาง
STATE
STATUS
PROPERTY
ถูก$_1$
แพง
สว่าง
MANNER
ชำนาญ
เร็ว
ถูก$_2$
ใหม่
ดี
EXIST
มี
อยู่
DESCRIBE
เป็น

4.3.1.2 Conceptual case constraints

As discussed earlier in 2.3.5, conceptual case constraints are represented as the properties of the head concept. Some constraints are regarded as general or default constraints, such as the concept "TIME", which is regarded as a default constraint on the conceptual case TIM. Others are considered individual constraints since they are properties or features of individual head concepts, such as constraints on conceptual cases OBJ and INS. Default conceptual case constraints needed for the analysis of our corpus are listed in (26) while individual constraints are listed in (27).

(26) CSTIM => "TIME"
CSLOC => "SPACE"
CSMAN => "STATUS"
CSMNS => "ABSTRACT"
CSINS => "TOOL"
CSTIM_B => "TIME"
CSTIM_E => "TIME"
CSNUM => "NUMBER"

(27)	ขยาย	: CSAGT => "HUMAN"
		: CSOBJ => "CONCRETE", "ABSTRACT"
	เขียน	: CSAGT => "HUMAN", "MACHINE"
		: CSOBJ => "PRODUCT"
		: CSINS => "MEDIA", "LANGUAGE"
	ใช้	: CSAGT => "HUMAN", "MACHINE"
		: CSOBJ => "MANUFACTURE"
	ซื้อ	: CSAGT => "HUMAN", "ORGANIZATION"
		: CSOBJ => "CONCRETE"
		: CSINS => "MONEY"
	ตี	: CSOBJ => "CONCRETE", "ABSTRACT"
	$ถูก_1$	: CSOBJ => "CONCRETE"
	$ถูก_2$	: CSOBJ => "THOUGHT"

ทำงาน : CSAGT => "HUMAN", "MACHINE"
บริจาค : CSAGT => "HUMAN", "ORGANIZATION"
: CSOBJ => "CONCRETE", "MONEY"
: CSAFF => "HUMAN", "ORGANIZATION"
ประสิทธิภาพ : CSPRPT => "HUMAN", "MACHINE", "ORGANIZATION"
เป็น : CSOBJ => "CONCRETE", "ABSTRACT"
: CSCMPL => "CONCRETE", "ABSTRACT"
พิมพ์ : CSAGT => "HUMAN"
: CSOBJ => "PRODUCT"
: CSINS => "TOOL"
เพิ่ม : CSAGT => "HUMAN", "MACHINE", "ORGANIZATION"
: CSOBJ => "CONCRETE", "ABSTRACT"
มี : CSAGT => "CONCRETE", "ABSTRACT"
: CSOBJ => "CONCRETE", "ABSTRACT"
ราคา : CSPRPT => "CONCRETE"
: CSCMPL => "MONEY"
เร็ว : CSOBJ => "CONCRETE"
วาง : CSAGT => "ANIMAL"
: CSOBJ => "CONCRETE"
สว่าง : CSOBJ => "CONCRETE"
ใหม่ : CSOBJ => "CONCRETE", "ABSTRACT"
อยู่ : CSOBJ => "CONCRETE", "ABSTRACT"

4.3.1.3 Case mapping

Mapping of syntactic cases onto conceptual cases are of two types: default mapping and individual mapping. Default mapping need not be specified in a lexeme since it is predictable which conceptual cases correspond to syntactic cases. For example, syntactic case POSSPR corresponds to conceptual case POSS. This mapping information is left unspecified. Default mapping in this study are listed in (29).

(29) RFROMPR => TIM_B

RTOPR => TIM_E

MWITHPR => MNS,INS

LATPR => LOC

LONPR => LOC

LFRTPR => LOC

LINPR => LOC,TIM

COMPPR => CMP

BENPR => AFF

ADJR => MAN

NUMR => NUM

CLSSR => CLSS,QUAT

ABOUTPR => ABOUT

POSSPR => POSS

Individual mapping of cases are defined as features, such as MSUBR, MFOBR, MCLSSR, on the head concept. Following is the list of individual mapping for head concepts used in this study.

(30) ขยาย : MSUBR => AGT, TOP

: MFOBR => OBJ

เขียน : MSUBR => AGT,OBJ,TIM

: MFOBR => OBJ

ใช้ : MSUBR => AGT,OBJ,TIM

: MFOBR => OBJ

ซื้อ : MSUBR => AGT,OBJ,INS,TIM

: MFOBR => OBJ

ดี : MSUBR => OBJ

ถูก$_1$: MSUBR => OBJ

ถูก$_2$: MSUBR => OBJ

ทำงาน : MSUBR => AGT,TIM

บริจาค : MSUBR => AGT,OBJ,TIM

: MFOBR => OBJ

	: MTOPR => OBJ
ประสิทธิภาพ	: MPOSSPR => PRPT
เป็น	: MSUBR => OBJ
	: MFOBR => CMPL
พิมพ์	: MSUBR => AGT,INS,OBJ,TIM
	: MFOBR => OBJ
เพิ่ม	: MSUBR => AGT,MNS,OBJ
	: MFOBR => OBJ
มี	: MSUBR => AGT
	: MFOBR => OBJ
ราคา	: MPOSSPR => PRPT
	: MSUBR => PRPT
	: MFOBR => CMPL
เร็ว	: MSUBR => OBJ
วาง	: MSUBR => AGT,OBJ
	: MFOBR => OBJ
สว่าง	: MSUBR => OBJ
ใหม่	: MSUBR => OBJ
อยู่	: MSUBR => OBJ,TIM

4.3.2 Analysis phases

The analysis process in CCA consists of two main phases: the assignment of all possible conceptual cases and the selection of the appropriate one.

A relative clause poses a unique phenomenon. Either subject or object of the verb is missing; therefore, another phase is added as a preliminary step. This is the supplying of the missing subject or object. This results in CCA having three phases altogether: subject-object resupplying; case assignment; and case selection.

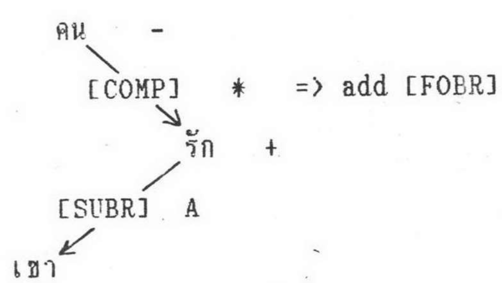
4.3.2.1 Subject-object resupplying

In a relative clause, either SUBR or FOBR case of relative verb is omitted. The syntactic case COMPR in the D-tree is used as a clue to trigger this preliminary phrase. Rules in (31a) apply to resupply the missing cases. These rules are organized into link LRelMissSubFob which is the only one link in this phase. Example in (31b) shows the status of windows when the object of a relative clause is omitted. Examples in (31c) and (31d) show the status of windows when the subject of a relative clause is omitted.

(31) a) Rules for finding the missing case.

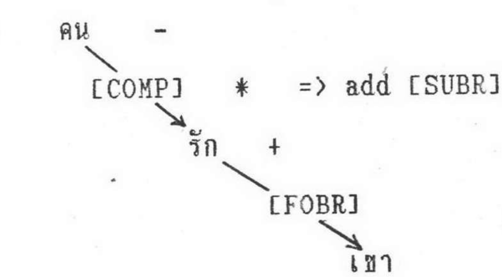
```
if (+)=[V] and (*)=[COMPR] and
     CSEARCH from (+) for A that A=[SUBR]
     then FOBR missing : assign [FOBR] to (+)
else if (+)=[V] and (*)=[COMPR] and
     CSEARCH from (+) for A that A=[SUBR] not found
     then SUBR missing : assign [SUBR] to (+)
endif
```

b)



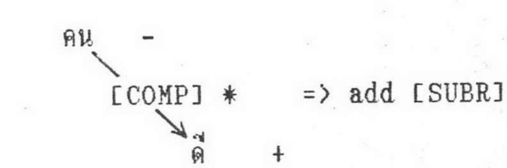
(คน ที่ เขา รัก)

c)



(คน ที่ รัก เขา)

d)



(คน ที่ ดี)

4.3.2.2 Case assignment

Case mapping and conceptual case constraints are information used for the assignment of conceptual cases. Case mapping is used to define possible conceptual cases. Each possible conceptual case will be checked against the constraints. For example, If the syntactic case SUBR relates the lexeme "ใช้" to the lexeme "เขา", and the environment of the lexemes "ใช้" and "เขา" are as (32b) and (32c) respectively while default constraints is (32a). Conceptual cases AGT, OBJ and TIM will be assigned to the syntactic relation between "เขา" and "ใช้". These cases are then checked against the constraints of "ใช้". The constraints on AGT, which are "HUMN" and "MACH", intersects with the feature UPCP of "เขา", which are "HUMN", "ANIM", and "CONC". On the other hand, the constraints on OBJ, which is "MANUF", does not intersect with the feature UPCP of "เขา", and the default constraint on TIM does not intersect with the feature UPCP of "เขา"; therefore, AGT is assigned as the conceptual case of "เขา" and "ใช้" as shown in (32d).

(32) a) default constraint: TIM => "TIME"

b) lexeme : "ใช้"

individual map => MSUBR : AGT, OBJ, TIM

individual constraints => CSAGT : "ANIM", "MACH"

=> CSOBJ : "MANUF"

c) lexeme : "เขา"

UPCP : "HUMN", "ANIM", "CONC"

d) "เขา<-SUBR-ใช้" => "เขา<-AGT-ใช้"

Rules used for assigning conceptual cases in example (32) are presented in (33).

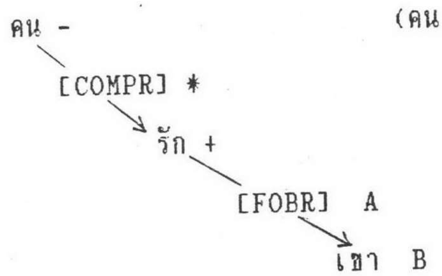
```
(33)  if -.MSUBR = [AGT] then
          if  -.CSAGT = +.UPCP then
                assign [AGT] to (*)  endif
      endif
```

```
if -.MSUBR = [OBJ] then
     if -.CSOBJ = +.UPCP then
          assign [OBJ] to (*)  endif
endif
if -.MSUBR = [TIM] then
     if -.CSTIM = +.UPCP or +.UPCP = "TIME" then
          assign [TIM] to (*)  endif
endif
```

The verb in a relative clause is different from that of the main clause because one of its argument nouns is its head and one is its depender. Rules for case assignment in the relative clause must consider both arguments at the same time. (34a) contains rules used when syntactic case FOBR relates the depender but the SUBR case is missing. Example (34b) shows the status of windows when these rules apply.

(34) a) Rules used for relative clause with object depender.

```
if +.RELMS = [SUBR] and (*)=[COMPR] and
     CSEARCH from (+) for A that A=[FOBR]
     if +.MSUBR = [AGT] then
          if +.CSAGT=-.UPCP then assign [AGT] to (*) endif
     endif
     if +.MSUBR = [OBJ] then
          if +.CSOBJ=-.UPCP then assign [OBJ] to (*) endif
     endif
     if +.MFOBR = [OBJ] and B is child of A then
          if +.CSOBJ=B.UPCP then assign [OBJ] to A endif
     endif
endif
```

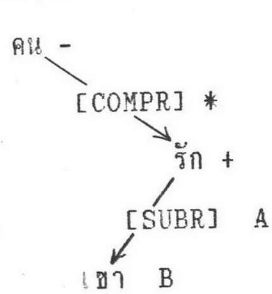
b) 

(คน ที่ รัก เขา)

Rules used when the depender in the relative verb has the SUBR case and the FOBR case is missing are given in (35a). Example (35b) shows the status of windows when these rules apply.

(35) a) Rules used for relative clause with subject depender.

```
if +.RELMS = [FOBR] and (*)=[COMPR] and
    CSEARCH from (+) for A that A=[SUBR] and B is child of A then
    if +.MSUBR = [AGT] then
        if +.CSAGT=B.UPCP then assign [AGT] to A endif
    endif
    if +.MSUBR = [OBJ] then
        if +.CSOBJ=B.UPCP then assign [OBJ] to A endif
    endif
    if +.MFOBR = [OBJ] then
        if +.CSOBJ=-.UPCP then assign [OBJ] to (*) endif
    endif
endif
```

b) 

(คน ที่ เขา รัก)

An exception to this is the relative clause in which there is no depender argument noun. (36a) gives rules used for this type of relative clause. Example (36b) shows the status of windows when these rules apply.

(36) a) Rules used for relative clause with no depender.

```
if +.RELMS = [SUBR] and (*)=[COMPR] and
     CSEARCH from (+) for X that X=[FOBR] not found
     if +.MSUBR = [AGT] then
          if +.CSAGT = -.UPCP then assign [AGT] to (*) endif
     endif
     if +.MSUBR = [OBJ] then
          if +.CSOBJ=-.UPCP then assign [OBJ] to (*) endif
     endif
endif
```

b)

```
คน -                          (คน ที่ ดี)
  \
   [COMPR] *
       ↘
         ดี  +
```

All rules used in this phase are organized into two links, LCaseAssign and LCaseAssign1, since there are many rules in this phase.

4.3.2.3 <u>Case selection</u>

The mapping between syntactic and conceptual cases can be one to one or one to many. In the latter case, the process to select the appropriate one is needed. Example (37) shows that SUBR can map onto both AGT and OBJ depending on whether the feature UPCP of "เขา" corresponds to either the animate concept or the concrete concept. In the sentence "เขา วาง คอมพิวเตอร์ ใน ที่ ที่ สว่าง", two conceptual cases can be assigned between "เขา" and "วาง" as in (37d). If the conceptual case derived from FOBR is OBJ as shown in (37e), the conceptual case OBJ derived from SUBR is canceled and thus AGT is selected.

(37) a) The lexeme : วาง

individual map => MSUBR : AGT, OBJ

MFOBR : OBJ

individual constraints => CSAGT : "ANIM"

CSOBJ : "CONC"

b) lexeme : เขา

UPCP : "HUMAN.ANIM.CONC"

c) lexeme : คอมพิวเตอร์

UPCP : "THING.SPACE.CONC.MACH.ARTIF.MANUF.INANM.TOOL.COMM".

d) เขา<-SUBR-วาง => เขา<-AGT-วาง

=> เขา<-OBJ-วาง

e) วาง-FOBR->คอมพิวเตอร์ => วาง-OBJ->คอมพิวเตอร์

The ambiguity of conceptual case is often found in the mapping of syntactic case TOPR, SUBR, FOBR and SOBR. This is solved by the use of case frame, which is the feature of verbs. The case frame in this study is considered only from the perspectives of these four syntactic cases. The conceptual cases that correspond to the case frame specified as feature of the verb will be selected. For example, the case frame of the lexeme "วาง" are [AGT,OBJ] and [OBJ]. If SUBR can map onto AGT and OBJ, and FOBR map onto OBJ, then AGT is preferred for SUBR and OBJ for FOBR because it corresponds to the case frame [AGT,OBJ] or [AO] of the verb "วาง". An example of case selection rules used for the verb with case frame AO is in (38a). The status of windows when these rules apply is shown in (38b)

(38) a) Rules used for case frame AO

```
if -.CFRM = [AO] then
    if (*)=[AGT] and CSEARCH from (-) for A that A=[OBJ]
        then select [AGT] for (*) and [OBJ] for A endif
endif
```

b)

```
            วาง -
          /      \
   [AGT] *    [OBJ] A
    /              \
เขา +              คอมพิวเตอร์
```

Case selection rules used for relative clause are different from those of the main clause. An example of rules used for the verb with case frame AO in a relative clause is in (39a). The status of windows when these rules apply is as (39b) or (39c).

(39) a) Rules used for case frame AO in relative clause

```
if +.CFRM = [AO] then
   if CSEARCH from (+) for A that A=[SUBR] and
   (*)=[OBJ] and A=[AGT] then
   select [AGT] for A and [OBJ] for (*) endif
   if CSEARCH from (+) for A that A=[FOBR] and
   (*)=[AGT] and A=[OBJ] then
   select [AGT] for (*) and [OBJ] for A endif
endif
```

```
b)      คน -                          (คน ที่ รัก เขา)
          \
           [AGT] *
              \
               รัก +
                  \
                   [OBJ] A
                      \
                       เขา

c)      คน -                          (คน ที่ เขา รัก)
          \
           [OBJ] *
              \
               รัก +
              /
           [AGT] A
          /
        เขา
```

Rules used for case selection are organized into Link LCaseFrame, which is only one link used in this phase. The order of links in CCA used for the corpus is list as follows: LRelMissSubFob, LCaseAssign, LCaseAssign1, LCaseFrame.

4.4 Output of analysis

CUPARSE analysis yields two levels of output for every sentence: a D-tree at the syntactic level and a conceptual network at the conceptual level. The symbol "[]" encloses lexical nodes in a D-tree and concept nodes in a conceptual network, while the symbol "()" encloses case nodes. The symbol "¦" is used to represent dependency relation. In a conceptual network, this symbol ends or begins with either "V" or "A" to denote a depender. The symbol "V" is used when the depender is located in the following line and the symbol "A" is used when the depender is on the preceding line. However, since all dependency relations in a D-tree are downward, no symbol is needed. The head in a D-tree is placed in the preceding line. For example, the lexeme "มี" is the depender of "คอมพิวเตอร์" in a D-tree (40a). The concept "HAVE" is the head of the concepts "COMPUTER" and "HARDDISK" in a conceptual network (40b).

(40) S2: เขา ใช้ คอมพิวเตอร์ ที่ มี ฮาร์ดดิสค์

a)
```
[        ใช้         ]
  ¦          ¦
(SUBR) (   FOBR    )
  ¦          ¦
[เขา ] [ คอมพิวเตอร์ ]
             ¦
       (  COMPR   )
             ¦
       [     มี     ]
             ¦
       (   FOBR   )
             ¦
       [ ฮาร์ดดิสค์  ]
```

b)
```
[      USE       ]
   ¦        ¦
   V        V
 (AGT)  (  OBJ   )
   ¦        ¦
   V        V
 [HE ]  [COMPUTER]
            A
            ¦
        ( AGT )
            A
            ¦
       [ HAVE ]
            ¦
            V
        ( OBJ )
            ¦
            V
       [HARDDISK]
```

Output of all the 50 sentences in the corpus analyzed by CUPARSE are listed in Appendix B.

During the analysis of these sentences, there are interesting instances of ambiguity, the handling of prefixes and case mapping, which should be pointed out and discussed.

4.4.1 Lexical ambiguity

Lexical ambiguity results from homonyms of one wordform having more than one meaning. There are two lexical ambiguities in the corpus. The first one is the ambiguity between "ที่" which represents the concept "LAND" and "ที่" which is used as a relator. The second is the ambiguity between "ถูก" which represents the concept "CHEAP" and "ถูก" which represents the concept "CORRECT".

The "ที่" ambiguity is found in many sentences. Disambiguation of this ambiguity can be done by the use of LD rule NCompV. The sentences which can be successfully disambiguated by this rule are the sentences 2,7,36,37,38,39, and 47 (see Appendix A). Only one possible path is, therefore, generated for these sentences. However, there are sentences which cannot be disambiguated by this rule. These are sentences 5,6,11,15,24,25, and 28. Two possible paths are generated for these sentences, but only one path succeeds during the syntactic analysis phase.

For sentences having the "ถูก" ambiguity, which are sentences 46 and 47, on the other hand, two possible paths are generated and both paths are syntactically successful because the lexemes "CHEAP" and "CORRECT" have the same category value. Disambiguation in these two sentences is done by conceptual case constraints. The following examples illustrate the possible paths for "ถูก" and the dictionary information used for disambiguation.

(41) a) S46: ใน ขณะนี้ ราคา ของ คอมพิวเตอร์ ถูก ลง

S47: คำตอบ ที่ ถูก จะ ต้อง เป็น เรื่อง เกี่ยวกับ คอมพิวเตอร์

b) Path 1: "ถูก" represents the concept "CHEAP".

Path 2: "ถูก" represents the concept "CORRECT".

c) PRICE : CSMAN = "PRPT_STS"

UPCP = "ABST,PRPT"

ANSWER : UPCP = "THOUGHT,HM_INTLL,ABST"

CHEAP : CSOBJ = "CONC"

UPCP = "PRPT_STS,STATUS,STATE,EVENT"

CORRECT : CSOBJ = "THOUGHT"

UPCP = "MAN_STS,STATUS,STATE,EVENT"

In sentence 46, path 1 is semantically correct because CSMAN of the concept "PRICE" matches UPCP of "CHEAP", while path 2 is semantically incorrect because CSMAN of the "PRICE" does not match UPCP of "CORRECT". In sentence 47, on the other hand, path 2 is semantically correct because CSOBJ of "CORRECT" matches UPCP of "ANSWER", while path 1 is semantically incorrect because CSOBJ of "CHEAP" does not match UPCP of "ANSWER" as shown in (41).

4.4.2 Category ambiguity

The lexeme with category ambiguity is a lexeme which has more than one category value for MAJCAT or MINCAT. Each value indicates the syntactic potential of the lexeme in terms of the syntactic processes which it can undergo. In the corpus, category ambiguities are found in the lexemes as listed in (42).

(42) ที่ MINCAT: PREP, COMP

ราคา MINCAT: CMNN, VEQU

ได้ MINCAT: RAATT, LAATT

เกี่ยวกับ MINCAT: PREPN, PREP

ต้อง MINCAT: LAAMD, LAATT

ห้อง MINCAT: CMNN, CLSS

อยู่ MINCAT: VCMN, RAASP

Disambiguation of categories is done by the first applicable syntactic process which the ambiguity features undergo. The category value which matches that of the syntactic rule will be selected. For example, the category value of "ที่" in sentence 6 is PREP because rule VpN matches the lexeme sequence "ใช้ ที่ ห้องแล็บ". On the other hand, the category value of "ที่" in sentence 7 is COMP because rule NCompV matches the lexeme sequence "คอมพิวเตอร์ ที่ ทำงาน" as shown in (43).

(43) S6: เขา ใช้ คอมพิวเตอร์ ที่ ห้องแล็บ

V PREP N => rule NpN apply

S7: เขา ใช้ คอมพิวเตอร์ ที่ ทำงาน เร็ว

N COMP V => rule NCompV apply

Another example showing category disambiguation of "ราคา" by syntactic rules is given in (44).

(44) S9: เครื่องพิมพ์ เครื่อง นี้ ราคา แพง

V Adj => rule AdjR apply

S10: ราคา ของ เครื่องพิมพ์ เครื่อง นี้ แพง

N PREPN N => rule NpnN apply

4.4.3 <u>Nominal prefix</u>

A nominal prefix is the lexeme used as the marker for a nominalized clause or a nominal noun. There are two nominal prefixes in this study: "การ" and "ความ". "การ" is the prefix used for a nominalized clause. "ความ" is the prefix used for a nominal noun and it is found immediately to the left of VADJ lexeme. It changes the category of that lexeme into noun as shown in (45). After this process, the lexeme "ชำนาญ" will have the syntactic property of a noun.

(45) S17: สมปอง เขียน โปรแกรม ด้วย ความ ชำนาญ

PREF VADJ

=> สมปอง เขียน โปรแกรม ด้วย ชำนาญ

NOM

The prefix "การ" is found to the left of VCMN lexeme. It changes the category value of that lexeme into noun. However, before the category changing rule applies, arguments of the VCMN lexeme should be attached to the VCMN first as shown in (46). After that, the lexeme "ขยาย" will have the syntactic property of a noun.

(46) S43: ประสิทธิภาพ ของ คอมพิวเตอร์ เพิ่ม ด้วย การ (ขยาย->หน่วยความจำ)

PREF VCMN

=> ประสิทธิภาพ ของ คอมพิวเตอร์ เพิ่ม ด้วย (ขยาย->หน่วยความจำ)

NOM

4.4.4 Case mapping

A comparison of D-trees and conceptual networks yielded by CUPARSE as output of the analysis reveals various forms of correspondence between syntactic cases and conceptual cases. First, there are sentences which have identical syntactic cases but different conceptual cases. Second, there are sentences which have identical conceptual cases but different syntactic cases.

4.4.4.1 One to many mapping

Sentences with identical syntactic cases but different conceptual cases are sentences 16, 17. In these two sentences, the syntactic cases are the same but the difference is found in the mapping of MWITHPR. In sentence 16, MWITHPR maps onto conceptual case INS while in sentence 17, it maps onto MAN.

(47) S16: สมปอง เขียน โปรแกรม ด้วย ภาษาC

S17: สมปอง เขียน โปรแกรม ด้วย ความ ชำนาญ

Selecting conceptual cases depends on the constraints on INS and MAN in lexeme "เขียน", as shown in (48). The UPCP "ABST" in (48c) is assigned from the prefix "ความ". In (48), CSINS of เขียน matches UPCP of "ภาษาC"; therefore, INS is selected for sentence 16. On the other hand, MAN is selected for sentence 17 because the default CSMAN matches UPCP of "ชำนาญ".

(48) a) The lexeme: เขียน

CSINS = "MEDIA,LANG"

CSMAN default = "ABST"

b) The lexeme: ภาษาC

UPCP:"LANG,HM_INTLL,ABST,MEDIA,COMM,ARTIF,MANUF,INANM,CONC"

c) The lexeme: (ความ) ชำนาญ

UPCP:"MAN_STS,STATUS,STATE,EVENT,ABST"

4.4.4.2 Many to one mapping

Many sentences in the corpus have the same conceptual case which are derived from different syntactic cases, such as sentences 16 and 18, sentences 12 and 13, sentences 42 and 44, etc. This phenomenon results from the noun phrase movement in a sentence. Examples to illustrate this many to one mapping are sentences 12 and 13.

(49) S12: รายงาน นี้ พิมพ์ ด้วย คอมพิวเตอร์

S13: คอมพิวเตอร์ พิมพ์ รายงาน นี้ ได้

The lexeme "คอมพิวเตอร์" in sentences 12 and 13 has the same conceptual case INS to "พิมพ์" whether it has syntactic case, MWITHPR as in sentence 12, or SUBR, as in sentence 13. INS is assigned from the correspondence between CSINS of "พิมพ์" and UPCP of "คอมพิวเตอร์" as shown in (50).

(50) The lexeme: พิมพ์

CSINS = "TOOL"

The lexeme: คอมพิวเตอร์

UPCP : "THING,SPACE,CONC,MACH,ARTIF,MANUF,INANM,TOOL,COMM"

It is true that there is a subtle difference in meaning between sentences 12 and 13, which is due to the difference in sentence perspective or informational structure. However, this subtle meaning difference is not included in our present study.

An interesting case of many to one mapping is what are called syntactic paraphrases. These paraphrasing sentences have the same conceptual network which are derived from different D-trees. Syntactic paraphrases found in this study are sentences 9 and 10, 19 and 20, 26 and 30, 27 and 29, 33 and 34, 44 and 45, as shown in (51).

(51) a) S9: เครื่องพิมพ์ เครื่อง นี้ ราคา แพง

S10: ราคา ของ เครื่องพิมพ์ เครื่อง นี้ แพง

b) S19: สมปอง จะ เขียน โปรแกรม ใน วันพรุ่งนี้

S20: ใน วันพรุ่งนี้ สมปอง จะ เขียน โปรแกรม

c) S26: คอมพิวเตอร์ ทุก เครื่อง ใน ห้องแล็บ เป็น คอมพิวเตอร์ รุ่น ใหม่

S30: คอมพิวเตอร์ ใน ห้องแล็บ ทุก เครื่อง เป็น คอมพิวเตอร์ รุ่น ใหม่

d) S27: คอมพิวเตอร์ ทั้ง 10 เครื่อง ใน ห้องแล็บ เป็น คอมพิวเตอร์ รุ่น ใหม่

S29: คอมพิวเตอร์ ใน ห้องแล็บ ทั้ง 10 เครื่อง เป็น คอมพิวเตอร์ รุ่น ใหม่

e) S33: บริษัท บริจาค คอมพิวเตอร์ แก่ มหาวิทยาลัย 10 เครื่อง

S34: บริษัท บริจาค คอมพิวเตอร์ 10 เครื่อง แก่ มหาวิทยาลัย

f) S44: ด้วย การ ขยาย หน่วยความจำ จะ เพิ่ม ประสิทธิภาพ ของ คอมพิวเตอร์

S45: การ ขยาย หน่วยความจำ จะ เพิ่ม ประสิทธิภาพ ของ คอมพิวเตอร์