

## Chapter 6

# Implementation of the Information-Structure Analyzer

In this chapter we demonstrate that the formalized theory can be implemented for practical applications and evaluation. In particular, we show that (1) the backbone of the system, CCG parser, is practical despite some previously-addressed concerns about spurious ambiguity and (2) the specifications of contextual link and information structure are implementable with some additional procedural aspects, which are modularly upgradable.

The chapter starts with an introduction of the overall architecture in the first section. The following two sections focus on the CCG parser and information-structure analyzer. In the latter section, we also discuss an implementation of particle prediction in Japanese based on the analysis in Chapter 5.

### 6.1 Introduction

The current system accepts text as input, analyzes its information structure, and predicts particle choice in Japanese as shown in Fig. 6.1. It has two main modules: the parser and the information-structure analyzer. Since our grammar, CCG, can recognize non-traditional constituency in accordance with divisions of information structure, analysis of information structure can proceed in

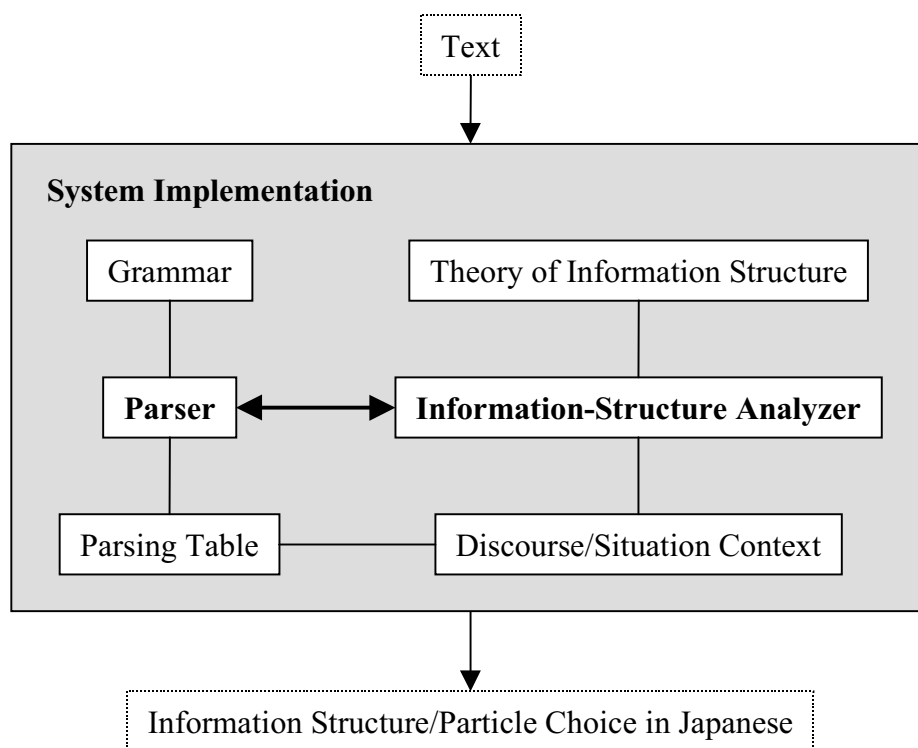


Figure 6.1: System Architecture

parallel to parsing.<sup>1</sup> This situation is represented by the bidirectional arrow ‘↔’ between the parser and the information-structure analyzer in the figure. Also in the system, the parsing table is used to derive the results in an efficient way, avoiding redundancy. The information obtained through parsing is stored as a part of the context, and later used for identifying discourse status. The parser has evolved from an earlier implementation [Komagata, 1997a (for Japanese)]. Another previous implementation [Komagata, 1998a (for English)] included a module for identifying information structure with limited analysis of linguistic marking and no structured-meaning component.

The system is implemented on a Sun Ultra E4000 2×250MHz Ultrasparcs with 320MB memory running SunOS 5.5.1. The code is written entirely in Sicstus Prolog Ver. 3. The program source files are approximately 100KB in size and the data/grammar files are about 200KB (including both training and test data and also lexicons).<sup>2</sup>

<sup>1</sup>This also makes it possible to control parsing, e.g., disambiguation, by the result of discourse processing. This possibility is left for future work.

<sup>2</sup>The source code and data files are available through the author’s thesis web page at

## 6.2 Practical CCG Parser

The practicality of our CCG parser depends primarily on the elimination of spurious ambiguity (i.e., multiple derivation of semantically-equivalent categories as introduced in Subsection 4.1.5) and some other engineering solutions such as preprocessing and the use of features.

We start this section with requirements for the parser. Then, we discuss the elimination of spurious ambiguity, processing of linguistic specifications, and the performance of the parser.

### 6.2.1 Requirements for the Parser

In order to process information structure as described in the previous chapters, we need a parser to derive semantic representations (to be precise, structured meanings) from input strings. In order to deal with this process, the parser needs to satisfy the following requirements:

1. Capable of processing referents (in our case semantic representations) across utterance boundaries for discourse-status analysis
2. Capable of parsing the complexity of real data, involving the following:
  - (a) Spurious ambiguity
  - (b) Genuine ambiguities (e.g., modification and coordination)
  - (c) Factors beyond ‘toy’ grammars: including inflection, punctuation, and lexical specification
3. Scalable to larger data (no pre-set limitation associated with the initial data and scale)
4. Applicable to multiple languages (at least English and Japanese)
5. Efficient enough for interactive use (response in the order of *seconds*)

Some of these, but not all, have been addressed in previous work with respect to CCG and similar formalisms. The CCG parsers have been built for several languages: English [Wittenburg, 1986; Komagata, 1998a], Turkish [Hoffman, 1995], and Japanese [Whitelock, 1988 (focus on morphology); Komagata, 1997a]. Applicability to fairly large data has also been shown by Wittenburg

---

[“http://www.cis.upenn.edu/~komagata/thesis.html”](http://www.cis.upenn.edu/~komagata/thesis.html).

[1986]. Application to long, complex sentences is shown to be practical [Komagata, 1997a] with a CKY-style parsing algorithm from [Aho and Ullman, 1972], cf. use of shift-reduce algorithm [Prevost, 1995; Hoffman, 1995]. Before proceeding, we need to distinguish **parsing** and **recognition**: the former derives semantic representation of a parse, and the latter only decides on grammaticality.

Since the problem with spurious ambiguity for practical parsing is only recently addressed, we include the discussion from Komagata [1997a] in the next subsection. The other issues are discussed in Subsection 6.2.3.

## 6.2.2 Elimination of Spurious Ambiguity

Let us first define several types of ambiguities involved in the parsing process:

- (193) *a. **Categorial ambiguity***: Availability of multiple categories (lexical/derivational), e.g., noun-verb ambiguity for *rose*
- b. **Spurious ambiguity***: Multiple derivations of *semantically-equivalent* categories, e.g., “John visited Bill.” has two derivations (left and right branching) in CCG with the identical semantic representation “*visited'* (*bill'*) (*john'*)”
- c. **Genuine ambiguity***:
- (i) Lexico-semantic ambiguity: Multiple semantic assignments to a single lexical category, e.g., financial *bank* vs. river *bank*
  - (ii) Attachment ambiguity: Multiple derivations of the same category with *distinct* semantics, e.g., PP attachment

Since spurious ambiguity is unnecessary and can result in an exponential explosion (see Section A.3), CCG parsers must implement some means of eliminating this type of ambiguity. We review three classes of approaches: (i) syntactic, (ii) semantic, and (iii) those which do not belong to the previous two.

First, syntactic approaches eliminate ‘spurious derivations’, which are not ‘the normal form’. Each proposal defines its own ‘normal form’, but a simplistic example is to choose, e.g., left-branching as the normal form. Then, if there are multiple derivations, only the left-branching is chosen. This does not necessarily suffer from incompleteness because if the left-branching is

unavailable, the right branching can be chosen without conflict. The syntactic approach blends naturally with a theoretical polynomial parsing algorithm for CCG [Vijay-Shanker and Weir, 1990]. Vijay-Shanker and Weir also include a mechanism of eliminating spurious ambiguity during a stage after recognition. Among several proposals, Hendriks [1993] and König [1994] work on Lambek calculus. But Lambek calculus does not include functional composition as a primitive rule. Thus, their proposal does not immediately apply to CCG. Hepple and Morrill [1989] cover a subset of the current formalism but do not have crossing instances of function composition nor type raising. Eisner [1996] covers an even wider range of CCGs but the case including type raising remains to be shown correct. By definition, the syntactic approach does not take semantics into consideration. But our definition of spurious ambiguity refers to semantics. Therefore, normal form parsing does not necessarily match our definition of spurious ambiguity elimination. There is an approach called labeled deduction, which includes semantics within syntactic types [Morrill, 1994]. But the above-mentioned syntactic approaches are not automatically applicable to labeled deduction.

Karttunen [1986] proposes the following semantic method. A new derivation is discarded if its semantic representation is *equivalent to* (or mutually subsumes) that of some entry with the same category already derived and stored.<sup>3</sup> This directly enforces the definition of spurious ambiguity and does not depend on the syntax. Note that ‘equivalence’ depends on the form of semantic representation [for general discussion Thompson, 1991]. For the case where the semantics is represented in  $\lambda$ -calculus, equivalence is not generally computable [Paulson, 1991]. For the case of feature structure, equivalence is defined as alphabetic variants and characterized by the isomorphism between the structures [Carpenter, 1992]. Our case corresponds to the latter.

Pareschi and Steedman [1987] present a method that belongs to the third type. The approach integrates Karttunen’s equivalence check in a CKY-style parsing algorithm, but invokes the mechanism for certain cases of category combination (i.e., a syntactic component). But the published algorithm is shown to be incomplete [Hepple, 1987]. Another approach by Wittenburg and Wall [1991] compiles the grammar so that only normal form derivation is possible. But this compilation replaces the original functional composition schemata with a ‘predictive’ version of composition schemata. As a consequence, certain non-traditional constituents such as subject-verb sequence

---

<sup>3</sup>For a detailed discussion of subsumption, see Shieber [1986].

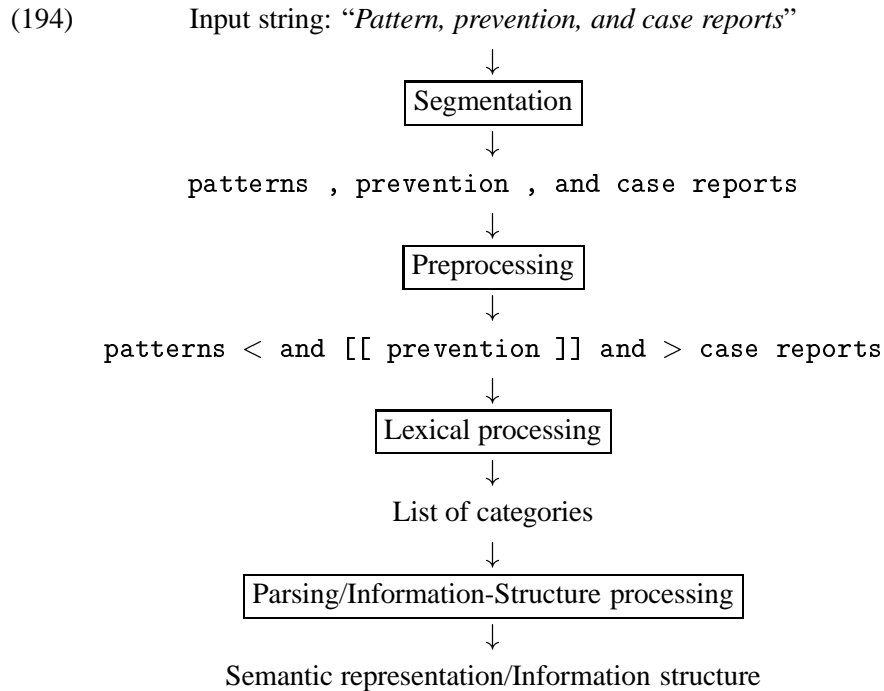
that depend on the original functional composition are no longer available for coordination. Thus, a crucial property of CCG is compromised.

Among the methods discussed above, we adopt Karttunen’s semantic equivalence check for its direct connection to the definition of spurious ambiguity and also for its conceptual simplicity. In support of this position, let us review some arguments against this approach. Eisner [1996] argues that a sequence of categories exemplified by “ $X/X \dots X \dots X \backslash X$ ” can slow down a parser exponentially. Here we assume that  $X/X$  and  $X \backslash X$  are ‘modifiers’ of  $X$  with *distinct* semantics, e.g., sentential adverbs. Then, this is an instance of genuine ambiguity because the result of “[ $X/X + X$ ] +  $X \backslash X$ ” and “ $X/X + [X + X \backslash X]$ ” have distinct semantics. Syntactic approaches would consider them as spurious ambiguity. But, then, derivation of semantic representation would face incompleteness. Wittenburg [1987] objects to the cost of an equivalence check. But an equivalence check (for our semantics) is inherently easier than the general case of subsumption check. The latter requires the costly *occurs check* for soundness [Pereira and Shieber, 1987]. Hepple and Morrill [1989] raise another objection. While syntactic methods detect spurious ambiguities before deriving a result, an equivalence check needs to compare the derived result with every entry in the current table cell. However, the cost associated with the semantic method depends on how many genuinely ambiguous entries are in the cell but not on the number of spuriously-ambiguous entries (they are eliminated as soon as they are derived and do not accumulate). This does not introduce additional complexity that is specific to the spurious ambiguity check. Further, once semantics is involved, it is not possible to distinguish between spurious and genuine ambiguities unless we actually check the involved semantics.

The effect of spurious ambiguity for a practical parser is enormous. In the implementation of [Komagata, 1997a] (with a CKY-style parser), a sentence with more than 10 words mostly resulted in an out-of-space error. This result applies to both parsing and recognition because spurious ambiguity can derive syntactic types in an exponential manner. By eliminating spurious ambiguities with a mutual subsumption method, the performance of a CCG parser can be brought to a level comparable to other grammar-based parsers.

### **6.2.3 Linguistic Specification and Processing**

In this subsection, we describe components of the parser in the order of processing as shown below.



### Segmentation

Segmentation is a simple finite-stage process that converts an input string of characters to a list of strings. Each string roughly corresponds to a word and many punctuation symbols. A sample specification (for English) to separate comma ‘,’ from the attached word is shown below.

(195) `segmentation(e, ", ", [break_before=yes, break_after=yes, delete=no])`.

### Preprocessing

The preprocessor is a finite-state string processor, and is an effective engineering solution to various problems. For example, frozen expressions, hyphens, numerals, and some punctuation are handled by the parser this way. The most significant effects can be seen in handling coordination. The preprocessor detects coordination patterns and replaces them in the following way (for 4-way coordination):

(196)                   A, B, C, Coord D

↓

A < Coord [[ B Coord C ]] Coord > D

This allows the parser to apply the same coordinator for multiple-conjunct coordination. Replacement is done repeatedly for the preceding and succeeding parts, but not recursively (thus it is still finite-state). The processor tries to match patterns starting from 3-way coordination, up to the 5-way case. Currently, the preprocessor stops searching for alternative patterns once a solution is found. In this respect, the parser is not complete, but this has not been a problem in our case.

The double square brackets ‘[[’ and ‘]]’ force that the combination between them must be complete before combination with outside categories. This fixes the domain of, e.g., “B Coord C” and is found extremely effective, especially when B or C includes an embedded coordination (without comma) within a conjunct. The underlined phrase in the following example is fixed in this way.

(197) Laboratory work includes blood tests, liver and renal function studies, analysis of aspirated fluids, and sputum cultures.

Multiple instances of such a case are observed in the data.

There is a parasitic effect associated with comma replacement. Simply replacing a comma with a coordinator may destroy the original span of the conjuncts. The following example involves an instance of comma replacement (underlined as and) which may be analyzed incorrectly.

(198) Original: Treatment generally consists of daily doses of isoniazid, rifampin, and ethambutol.

- a. \* Treatment generally consists of {{daily doses of isoniazid and pyrazinamide} and ethambutol}.
- b. Treatment generally consists of daily doses of {isoniazid and pyrazinamide and ethambutol}.

In order to avoid this problem, an additional pair of symbols ‘<’ and ‘>’ are used. They glue the entire span of the original coordination.

Preprocessing of frozen expressions is slightly different from the above case in that the specification is found in the lexical entries such as follows:

(199) ["x", "-", "ray"] := [lang=e, head="ray", class=n(c), infl=reg].

For this reason, the preprocessor needs to maintain a set of frozen expressions and check the segmented list of strings against them. By adopting ‘longest-match’, the process prioritizes matching

frozen expressions over single-word entries.

Finally, the current process ignores discourse markers. Discourse markers indicate a relation between utterances and are not used in our analysis of information structure. This situation is currently handled by the preprocessor, which eliminates the following discourse markers: *and*, *but*, *so*, “*in addition*”, *however*, and *therefore* in the environments shown below (with or without comma).

- (200) a. `However(,) <utterance>`  
b. `<subject>(,) however(,) <rest of utterance>`

The second case is applied only to discourse markers without other functions, e.g., *however*, but not *and*, which is also a coordinator.

## Lexical Processing

The lexical processing consists of identifying the matching lexical entry and assigning the corresponding categories. Some examples of lexical entries are shown below, but the details do not matter here.

- (201) a. `"medicine" := [lang=e,class=n(u),infl=reg,pre_np=yes,arg=[pp(for)]] .`  
b. `"his" := [lang=e,class=det(his),num_pers=[-,-,s3,-,-,p3],def=yes] .`  
c. `"require" := [lang=e,class=v(reg),infl=reg-d,arg=[np,[np,pp(for)]]] .`

These are all English entries for the specified word classes. Information such as inflection, agreement, and subcategorization is also included. For our training data set (more detail on the data is in the next chapter) including 16 texts or 2300 words, there are about 900 such lexical entries including punctuation.

The following is an example of a singular noun inflection macro also including some other common properties. A macro is later used as a part of lexical assignment.

- (202) `macro(e,noun_infl_sg,  
    [if(class=n(_)),  
    ifnot(pl_only=yes),  
    lab=np(com),  
    (if(human=H) -> [] ; [call(H=no)]),  
    (if(sit=Sit) -> [] ; [call(Sit=no)]),`

```

(if(class=n(c)) -> [call(UC=c),call(NP='')] ;
[call(NP=np),(if(class=n(u)) -> [call(UC=u)] ; [call(UC=_)])]),
{if(gend=G)},
(if(implicit_arg=req) -> [call(Inf=yes)] ; []),
features=[agr=[-, -, s3, -, -, -], G, _], n_np=[n(UC), NP], human=H, def=_,
sit=Sit, inferrable=Inf],
locase_pf(Int),
int=Int,
cont=(cont, n: Int)).

```

This macro only applies to the noun class *n*, specifies a syntactic type *np(com)*, sets features including agreement and human, and specifies the semantic representation as the lower case of the string. The above specification is written in a form of a simple procedural description language. There are assignment and conditional statements. These statements are interpreted by the system at the time of lexical look up.

The standard Montagovian analysis of NP is that there is a (common) noun category *N* (or *CN*) and determiner category *NP/N*. The current implementation deviates from this by assuming a single category *NP* for both of these. The distinction between *N* and *NP* is still maintained by the use of features. This approach has an advantage of reducing categorial ambiguity for, e.g., plural nouns. While *N* and *NP* are specified with features *n\_np=[n(c), -]* and *n\_np=[-, np]*, respectively, an ambiguous case is simply *n\_np=[n(c), np]* (all for the countable case).

The above macro is used as a template for several lexical assignment for nouns including the following that subcategorizes a PP.

```

(203) lex_assign(e, n(_),
[ifnot(num_req=yes),
incl(arg, pp(Prep)),
(macro(noun_infl_sg); macro(noun_infl_pl)),
lab=(Lab=>Lab/pp(Prep)),
(if(implicit_arg=req) -> [call(Src=arg)] ; [call(Src=self)]),
features=(F=>[npostmod=yes, composition=no]
/[composition=no, colon=no, context_link=proj(Src)]),
int=(Int=>PP^(Int-PP))).

```

It calls the macro (both singular and plural cases), adds the PP argument, adds the features corresponding to the PP argument, and also adjusts the semantic representation to reflect this change.

For the training data set, there are about 200 lexical assignments (but not including macros) including different subcategorizations for verbs.

In the above, we have seen the specification of a noun class that takes a PP as an argument, rather than as a modifier. We take this position for most post-nominal PP's including situational ones. The motivation for the current position comes from difficulty with explosive ambiguity for considering PP's as post-nominal *modifiers*. Although this move might sound too restrictive, it actually corresponds to the difficulty in choosing the right preposition for a post-nominal PP modification, often experienced by non-native speakers. Thus, it seems justifiable that most noun-preposition relations must be specified.

The lexical processor reads the output of the preprocessor and assigns a set of categories to each string. For some string (e.g., *in*), over a dozen categories are assigned. In principle, looking up inflected forms takes a simple approach of generate and test. But to avoid the inefficiency of looking up unnecessary forms, the current implementation skips the cases where the stem is different from the target word. This technique has improved the performance of the current implementation over that of [Komagata, 1997a].

The system is capable of parsing both English and Japanese provided that the corresponding sets of lexical entries are prepared. But the current implementation only contains the English lexicon reflecting the scope of work. Although the implementation is also capable of dealing with generalized type-raised categories (GTRC) [Komagata, 1997a], which can simplify the grammar for Japanese, the capability is not activated because it is not necessary for English.

### **Use of Features**

The focus of Komagata [1997a] was elimination of spurious ambiguity. The paper avoided the issue of genuine ambiguity by working mainly on recognition. In the previous implementation [Komagata, 1998a], I tackled a small, but noticeable part of genuine ambiguity as well. It is a subclass of genuine ambiguity that can be resolved by use of features. Let us call it 'absurd' ambiguity. This subclass must be distinguished from the main, and more difficult type of genuine ambiguity that requires domain-specific or more general world knowledge.

Absurd ambiguities are eliminated by using both syntactic and semantic features in the grammar. Some of these features are to (i) limit modification structures in and around NPs, (ii) restrict

coordination patterns, (iii) condition on the modification of adjectives by *more/most*, and (iv) apply the human/non-human distinction.

The following examples show an absurd modification with respect to syntax possibly allowed by a coarse grammar.

(204) a. \* [minor skin] complication, cf. minor [skin complication]

b. \*[tuberculosis in a young baseball] player, cf. tuberculosis [in a young baseball player]

A coarse grammar that allows noun-noun compounds in a reasonably general way may face absurd ambiguities like this. As a first technique to reduce these absurd ambiguities, the current grammar assumes and imposes the following structure in/around NP, mainly adopting the analysis in [Quirk et al., 1985].

(205)

$$\left[ \left[ \left[ \text{Predet} \left[ \text{Det} \left[ \text{Pre-mod} \left[ \left[ \text{Pre-N Noun} \right] \text{NPost-mod} \right] \right] \right] \right] \right] \text{NPPost-mod} \right]$$

a. Predet: predeterminers such as *such* and *half*

b. Det: determiners such as (in)definite article

c. Pre-mod: premodifier such as adjective

d. Pre-N: noun to form a noun-noun compound with the head noun

e. NPost-mod: post-nominal modifier such as PP, restrictive relative

f. NPPost-mod: post-NP modifier such as appositive, non-restrictive relative

This restriction is achieved by the use of features such as *premod=yes* or *npostmod=yes* for results of pre- and post-nominal modification. The head noun that allows noun-noun compounds has features [*premod=no, npostmod=no*] to avoid these ‘heavy’ words.

The distinction between nominal and NP modification is crucial. For example, “*acute injuries typical of the sport*” must be analyzed as “[*acute injuries*] *typical of the sport*”, not as “*acute* [*injuries typical of the sport*]”. The modifier “*typical of*” is assigned a feature *npostmod=no* and is prevented from modifying the noun *injuries*. If there is no adjective *acute*, the noun *injuries* can be successfully modified by a post-NP modifier because it is underspecified between a noun and an NP.

There is another case involving nouns that can form a noun-noun compound. For this case, coordination is the primary factor. A phrase “*exercise modifications or medications*” should be analyzed as “[*exercise modifications*] *or medications*” but not as “\**exercise [modifications or medications]*”. Now, both *modification* and *medication* are allowed to form a noun-noun compound, e.g., “*exercise modifications*” and “*antihypertensive medications*”. Thus, a general form of coordination allows the unintended analysis. To avoid this, the current implementation adds a procedural constraint to exclude noun-noun compounds where the second noun is a result of coordination.

Absurd modification may also involve lexico-semantic aspects as can be seen in the following example.

- (206) *a.* \*[most lateral] ankle sprains
- b.* most [lateral ankle sprains]
- c.* cf. [most unusual] ankle sprains

The lexical specification of adjectives includes whether it can be modified by *more* and *most* (this information is shared as an inflectional feature whether the adjective can have suffix forms of comparative/superative).

The parser also uses the feature ‘human’ for various purposes including subject-verb agreement, modification, and coordination. Without this feature, the expression “refining rehabilitation” can be ambiguous between “an act of (someone’s) refining rehabilitation” or “rehabilitation that refines something”, as in the well-known “flying planes” example. In our case, the verb entry for *refine* specifies that the subject be ‘human’, eliminating the latter possibility.

Finally, the current grammar specifies agreement and subcategorization fairly accurately. For example, in addition to subject-verb agreement, the grammar specifies agreement for relative clauses including the possessive form and various coordination patterns. This helps the disambiguation process substantially.

## **Parsing**

The list of sets of categories obtained through the lexical processing is now fed to the CKY-style parser based on Aho and Ullman [1972]. Informally, a CKY-style algorithm parses “*Felix praised Donald*” using a chart, as shown in Fig. 6.2.

	Column $a$	Column $b$	Column $c$
Row 1	Felix	praised	Donald
Row 2	[Felix praised] <sub>1a+1b</sub>	[praised Donald] <sub>1b+1c</sub>	
Row 3	[Felix praised Donald] <sub>2a+1c</sub> [Felix praised Donald] <sub>1a+2b</sub>		

Figure 6.2: CKY-Style Parsing Table

Starting from the lexical categories for the entries in row 1, the parser proceeds to a lower row by combining the component categories specified in the subscript. In Row 3, multiple entries with exactly the same results are obtained. This is an example of spurious ambiguity. As we have discussed, we adopt a mutual subsumption check to eliminate spurious ambiguity. For the above case, since the two entries in 3a have equivalent semantics, they are reduced to a single entry. This process takes place whenever a new entry is entered into a cell. The situation gets more complicated once structured meaning is introduced. We will come back to this topic in the next section.

The linguistic specification file contains the following CCG rules to combine categories.

- (207)  $\text{c cg\_rule}(e, [x/y, y] \Rightarrow x, [])$ . ( $>$ )  
 $\text{c cg\_rule}(e, [y, x \backslash y] \Rightarrow x, [])$ . ( $<$ )  
 $\text{c cg\_rule}(e, [x/y, y/z] \Rightarrow x/z, [])$ . ( $>B$ )  
 $\text{c cg\_rule}(e, [x/y, y/z/u] \Rightarrow x/z/u, [])$ . ( $>B^2$ )  
 $\text{c cg\_rule}(e, [y?z, x \backslash y] \Rightarrow x?z, [])$ . ( $<B_{(\times)}$ )  
 $\text{c cg\_rule}(e, [x, \&, x] \Rightarrow x, [])$ . ( $<\&>$ )

The question mark ‘?’ is used for underspecifying the slash directionality (but the two instances of ‘?’ must agree). This rule specification is interpreted by the program for the corresponding operation. In the present implementation, type raising is considered a unary rule, and is activated dynamically when categories  $NP$  or  $PP$  are inserted into the CKY table.  $NP$  is type raised to  $S/(S \backslash NP)$  and  $S \backslash (S/NP)$ , and  $PP$  is type raised to  $S \backslash (S/PP)$ .

## 6.2.4 Performance

In this subsection, we discuss the performance of the parser for the training data (i.e., before extension to the test data), and show that it provides a reasonable backbone for analyzing information structure.

The system parses 16 introduction sections of medical case reports including 131 utterances. The average word length for an utterance (after preprocessing and including punctuation symbols) is 20, and the maximum, 42. There are four utterances beyond this level. Since they slow down the process so much, they are divided into two segments. Unfortunately, the utterances of 40 or more words seem to be beyond the capacity of the system. After this preparation, the measured CPU time is on average 16 seconds per utterance.<sup>4</sup> The average number of parses per utterance is 16. While there are a number of utterances that take too long for an interactive response, many utterances can be parsed in the order of *seconds*.

The above performance does not appear as good as the previous version [Komagata, 1998a] implemented for the abstracts of the same journal. The average parse time was about 2 seconds per utterance. But there are several factors involved in this difference. The average utterance length increased from 17 to 20. If we assume cubic parsing complexity in practice, this translates to 60% increase in parse time. The total size of the lexicon has increased about 50%. This proportionally slows down the lexical look up time. The parser now processes structured meaning. As we discuss at end of Subsection 6.3.3, structured meaning can introduce additional complexity. This seems to be reflected in the increase in average number of parse from 2 to 16. Considering all these, the performance of the present parser seems to scale reasonably from the previous implementation.

Since the goal of this implementation is to provide an adequate platform for analyzing information structure, no comprehensive comparison with other parsing systems is made. Informal side comments are that those long sentences are very difficult for a large-scale grammar-based parsers. For example, the XTAG parser [Doran et al., 1994] would have difficulty parsing many of the long sentences in our texts. Since the XTAG system has hundreds of thousands of lexical entries and up to dozens of trees for each lexical entry, this is only a confirmation that parsing real data is still challenging.

We thus conclude that the parser can be a reasonable platform for analyzing information structure. Two major factors are the use of CKY-style parsing algorithm and the elimination of spurious ambiguity, in comparison to earlier experimental parsers [Prevost, 1995; Hoffman, 1995]. The

---

<sup>4</sup>Time measurement is done by `Sicstus` built-in predicate `statistics`. The time measurement includes most of the stages: segmentation, preprocessing, lexical processing, and CKY parsing with derivation of semantic representations (structured meanings). There are a few off-line processes such as asserting (i) relations between a word-form and the canonical form and (ii) a set of frozen expressions. These can be done in a negligible time.

parser also demonstrates improvements over the version in Komagata [1997a] due to preprocessing, more efficient lexical processing, and use of features.

## 6.3 Processing Information Structure

This section presents the key element of the system, the information-structure analyzer. This module is a straightforward implementation of the theory developed in Chapter 3 and formalized in Chapter 4. It includes a small number of procedural aspects, but they are modularly specified and can be upgraded when necessary.

Each step of processing information structure is associated with a step of parsing. Parsing steps consist of lexical processing and combination of two (non-coordination) or three categories (coordination). Thus, this section only describes *local* processes applicable to either lexical or combinatory process.

In this section, we discuss the three properties for contextual links, composition of structured meaning, identification of information structure, and prediction of particle choices in Japanese.

### 6.3.1 Discourse Status and Domain-Specific Knowledge

#### Discourse Status

As a consequence of adopting a CKY-style algorithm for parsing CCG, semantic representations corresponding to information structure units are available in CKY table cells. In order to analyze discourse status, we modify the CKY table so that table cells only contain *pointers* to categories, not categories themselves. Categories are stored in the discourse context by the `assert` predicate of Prolog. Then, we can easily decide whether the category should remain in the context. Basically, we keep all the categories that are used in a successful parse.<sup>5</sup> Then, we can define the notion of discourse-oldness as presence of an equivalent category in the discourse context. The process of identifying discourse-old referents utilizes Prolog's unification mechanism. In order to correctly identify the existence of an equivalent semantic representation, we use mutual subsumption, not

---

<sup>5</sup>The actual process of asserting categories is slightly more complicated. Categories are initially assigned a temporary status until the parsing process completes. After completion, a top-down process traces down the successful parses and changes the temporary status to a permanent one. The unsuccessful categories are then eliminated.

simple unification.<sup>6</sup>

When there are multiple occurrences of identical semantic representations in a single utterance, only one instance is asserted and pointed to from multiple CKY-table cells. At this point, analysis of discourse-oldness is applied only across utterances. Thus, intra-utterance reference cannot be made. This is not a problem for the analysis of information structure, as will be seen in the next chapter.

### **Situationally-Available Referents and Domain-Specific Knowledge**

The above discourse-status processing can be applied to the analysis of situationally-EVOKED referents as well. For example, pronouns such as *we* and *they* are asserted at the beginning of an analysis under the assumption that these are situationally available.

The present proposal also assumes domain-specific knowledge that referents such as *physician*, *clinician*, and *patient* are available in the domain. This assumption can be implemented exactly the same way as for the above case of pronouns. That is, common nouns *physician*, *clinician*, and *patient* are asserted at the beginning of an analysis. Thus, this case too can be handled by the same mechanism as that for discourse status.

### **Use of Morphological Forms**

There is one procedural aspect added to the lexical process. In identifying discourse status, we also use morphological forms as a cue [see Dahl et al., 1987 and Palmer et al., 1993 for an analysis of derivational forms]. For example, the use of a verb *damage* is assumed to imply that there is a damaging event. Then, a NP “*the damage*” may be considered to refer to that event. This is in a sense a combination of linguistic marking of contextual linking and discourse status because we identify the contextual-link status of a word only if a morphologically-related referent is discourse-old.

Currently, the system deals with the following cases:

(208) *a.* Nouns: between singular and plural forms

*b.* Adjectives: between base, comparative, and superlative forms

---

<sup>6</sup>Sicstus Prolog has a built-in predicate called `variant` which does exactly the mutual subsumption, i.e., identity except for variable names.

- c. Verbs: between inflected forms, e.g. *damages* and *damaged*
- d. Derivation: between a noun and a verb with a shared sense

The system realizes the above condition by keeping content information (usually a dictionary form) as in the underlined portion below.

```
(209) macro (e, noun_infl_sg,
          [if(class=n(_)),
           .
           .
          lowercase(Int),
          int=Int,
          cont=(cont,n:Int)]).
```

Here, the attribute `cont` (for content information) has a pair of values. The first component `cont` indicates that the entry is a content word and not a function word. The second component `n: Int` indicates that the entry is a noun with a key value shared among different word classes.<sup>7</sup> For example, a noun *damage* contains a feature `cont=(cont,n: damage)` and the verb *damage* contains `cont=(cont,v: damage)`. The system checks the noun-verb relation by comparing the specification but ignoring the difference between the word classes `n` and `v`. For inflection, the entire content specifications are compared. We need to use this feature rather than semantic representations because the latter naturally differ between the cases mentioned above.

The above process for morphologically-related forms is only available at the lexical level. But its effect may project to a more complex structure exactly like other contextual links.

### 6.3.2 Linguistic Marking of Contextual Links

This subsection describes how the system processes linguistic marking of contextual links. The discussion covers the following topics: lexical assignment of categories, composition of two categories, a special case involving utterance-initial modifiers, and coordination.

#### Lexical Processing

There are a few cases where linguistic marking of contextual links needs to be processed at the time of lexical processing. First, function words are assigned contextual-link status. This

---

<sup>7</sup>In this example, `Int` is unified with the phonological form of the entry.

class includes: auxiliary verbs, modals, prepositions, and subordinators. They have the feature  $\text{cont}=(\text{func}, \text{FuncWordType})$  where *FuncWordType* specifies a type of function word. Since function words are available in the grammar, we can associate them with zero-inference. Thus, it seems natural to assume a contextual-link status for them.

Another case is two-place nouns such as *page* (see discussion on p. 68 in 3.3.1). This type of nouns can be considered to have an implicit argument without a PP argument, and thus is assigned a contextual-link status. The process needs to check if the category is *NP* without arguments and the feature  $\text{implicit\_arg=req}$  is specified.

Finally, numerals with the category *num* are assigned a *non-contextual-link* status.

### Composition of Two Categories

In Chapter 4, we presented a specification of linguistic marking (of contextual links) in terms of feature unification associated with categories. The system still uses features for this purpose, but implements them in a slightly different way. To avoid cluttering the feature area and to consolidate specifications shared in different categories, the system includes a special module to deal with assignment and projection of contextual-link statuses.

For example, a contextual-link projection from the argument is specified as a feature “ $cl = \text{proj}(arg)$ ” (*cl* for contextual link) on the argument of the functor category, and the special module processes structured meaning according to the specification shown below.

(210)	Determiner	Noun
Example:	<i>many</i>	<i>researchers</i>
Syntactic type:	$NP / \begin{matrix} N \\ cl = \text{proj}(arg) \\ \langle C_1, - \rangle \end{matrix}$	$N$ $\langle -, N_2 \rangle$
Syntactic type:	$NP$ $\langle -, N' \rangle$	

There are three more features corresponding to the specification of contextual-link assignment/projection: “ $cl = \text{set}$ ”, “ $cl = \text{reset}$ ”, and “ $cl = \text{proj}(self)$ ”.

We now move to specific cases. First, let us discuss some special cases: composition with dummy categories (e.g., punctuation) and function words as an argument (e.g., particles), and composition of two function words. In these cases, function words are handled transparently. That

is, the result is a projection of the contextual-link status of the other component. Composition of two function words is treated as a new function word.

Next, the process sets the contextual-link status to the following: definite determiner, indefinite generic, and utterance-initial modifier. The type of compositions involving the definite determiner can be represented as “ $X_{[def=yes]}/Y + Y$ ”. That is, only the feature `def=yes` on the result category specifies the process. This specification is more general than explicitly specifying “ $NP_{[def=yes]}/NP + NP$ ”. Thus, the notion of ‘definiteness’ (for the purpose of setting a contextual link) can be extended to other categories as well. For an indefinite generic, the composition can be represented as: “ $X_{[def=no]}/Y + Y$ ” with the additional condition that  $Y$  is a contextual link. The contextual-link status is set only if the right category,  $Y$ , is a contextual link. Utterance-initial modifiers receive a contextual-link status if the result of composition is  $S/S$ . Inverted phrases also assign a contextual-link status.

The case that assigns a non-contextual-link status is analogous. For an indefinite article, “ $X_{[def=no]}/Y + Y$ ” is specified. For numerals of the modifier type, the following pattern is detected and processed “ $X/Y_{[cl=reset]} + Y$ ”.

The system may also project the contextual-link status from an argument to the result. This takes place for the pattern “ $X.../Y_{[cl=proj(arg)]} + Y$ ”, its directional variant “ $Y + X... \setminus Y_{[cl=proj(arg)]}$ ”, and for a complex argument “ $X.../ (Y_{[cl=proj(arg)]}/Z) + (Y/Z)$ ”.

Projection of the contextual-link status from itself is similar. The same set of patterns are currently implemented: “ $X.../Y_{[cl=proj(self)]} + Y$ ”, “ $Y + X... \setminus Y_{[cl=proj(self)]}$ ”, and “ $X.../ (Y_{[cl=proj(self)]}/Z) + (Y/Z)$ ”.

### Composition of an Utterance-Initial Modifier and the Subject

There is a case where linguistic marking functions slightly differently from the previous cases. It involves an utterance-initial modifier, analyzed as  $\langle C_1, - \rangle$ , composing with the main clause with a structured meaning  $\langle \begin{matrix} C_2 \\ \text{contextual link} \end{matrix}, \begin{matrix} N_2 \\ \text{non-link} \end{matrix} \rangle$  where  $C_2$  is the subject. If the combination of  $C_1 + C_2$  is a contextual link, the resulting structured meaning is  $\langle C', N_1 \rangle$  where  $C' = C_1 + C_2$ . This is a kind of discontinuous information structure and possible even though the combination of the utterance-initial modifier and the subject cannot form a constituent in English.<sup>8</sup> If  $C'$  is not a contextual link

<sup>8</sup>For example, such a phrase cannot form a conjunct in English.

on its own, the resulting structured meaning would be  $\langle C_1, C_2 + N_2 \rangle$ , i.e., the entire main clause becomes a rheme. But an observation of the experiment data suggests that in many cases, the informational partition in the main clause seems as strong as the case without an utterance-initial modifier. To accommodate this situation, we assume the following hypothesis:

- (211) (Operational hypothesis) The utterance-initial modifier is not only a contextual-link marker of the modifier phrase itself but also a marker of the discontinuous theme including the subject where the subject is a contextual link.

Such a discontinuous theme can satisfy the condition of a discontinuous structured-meaning component: the semantic representation of the utterance-initial modifier and the subject can compose to derive a sound semantic representation. For example, an adverb *yesterday* with “ $\lambda X.S//yesterday'$ ” and the subject *John* with  $\lambda P.P(john')$  can derive “ $\lambda P.[P(john')]/yesterday'$ ”. Note that the notation  $X//Y$  is used for a modification (or adjunct) structure, which is distinguished from the functor-argument structure. In terms of information structure, there is no reason such a semantic representation cannot be a (discontinuous) theme. In fact, Japanese allows coordination of a phrase corresponding to “*yesterday–John*” with another phrase, say, “*today–Mary*”. In this case, the subject must be compatible with the type-raised form  $S/(S\backslash NP)$ . Then, a modifier-subject composition can be recognized as “ $S/S + S/(S\backslash NP) \implies S/(S\backslash NP)$ ” with the intended semantics.

In terms of assignment/projection of contextual links, we can consider that utterance-initial modifiers either (i) project the contextual-link status of the subject or (ii) project the status of itself. In the system, the same function is performed by the above-mentioned module that deals with assignment/projection of contextual links.

## Summary

The process of identifying contextual links is summarized in Fig. 6.3 on page 180.

### 6.3.3 Composition of Structured Meaning

Perhaps, the most innovative feature of the current system is implementation of structured meaning in a fairly general sense. This subsection describes the implementation of the ideas formalized in

Section 4.3. At the end, we also describe the way we deal with spurious ambiguity in relation to structured meaning.

### Data Types for Structured Meaning

Let us represent a structured meaning in the following form:

$\langle \begin{matrix} C \\ \text{contextual link} \end{matrix}, \begin{matrix} N \\ \text{non-link} \end{matrix} \rangle_{\text{LeftBoundary-RightBoundary}}$ . Although we allow arbitrary discontinuous construction of  $C$  and  $N$ , we distinguish instances of structured meaning only by the boundary categories.

Then, we have the following six possible types of structured meanings for inputs and results:  $\langle C, N \rangle_{C-C}$ ,  $\langle C, N \rangle_{C-N}$ ,  $\langle C, N \rangle_{N-C}$ ,  $\langle C, N \rangle_{N-N}$ ,  $\langle -, N \rangle$ ,  $\langle C, - \rangle$ .<sup>9</sup> As a consequence, the number of composition rules is bounded. The recursive process of dealing with structured meanings is defined for the lexical and the derivation steps (Subsection 4.3.1). The existence of the bound on the derivational process thus guarantees a closed operation.

To be complete, we have to discuss all the possible combinations, i.e., 216 (see p. 115). But, since it is tedious and not all the cases are equally common, the system only implements about 20 possibilities. In the following, we look at a few common cases among those discussed in Subsection 4.3.1. Note that we use the notations  $Type_C$  and  $Type_N$ , representing the syntactic type corresponding to  $C$  and  $N$ , respectively.

**Composition Type:**  $\langle C_1, N_1 \rangle_{N-C} + \langle -, N_2 \rangle$

Let us first recall the case where this type of composition is needed. In Subsection 4.3.1, we observed the non-traditional derivation of “[**Fred** praised] [**Donald**]”, e.g., as a response to “Who praised who?”. The component “**Fred** praised” is analyzed as  $\langle \text{praised}', \text{fred}' \rangle_{\text{fred}' - \text{praised}'}$  where the contextual-link and non-link components of the structured meaning are  $\text{praised}'$  and  $\text{fred}'$ , respectively, and the left and the right boundaries are  $\text{fred}'$  and  $\text{praised}'$ , respectively.

Now, the composition in question,  $\langle C_1, N_1 \rangle_{N-C} + \langle -, N_2 \rangle$ , would result in another structured meaning,  $\langle C_1, N' \rangle_{N-N}$  where,  $N'$  is a semantic composition of  $N_1$  and  $N_2$  and the boundary  $N - N$  indicates that  $C_1$  is not at the boundaries. But, as we have discussed in Subsection 4.3.1, this  $N'$  must satisfy certain conditions so that the composition of  $C_1$  and  $N'$  can result in the correct

<sup>9</sup>Note that  $\langle C, - \rangle$  and  $\langle -, N \rangle$  are the cases where the entire phrase is a contextual link and a non-contextual link, respectively.

semantic representation corresponding to the entire phrase. For the current example, it must be  $\lambda P.P(\text{donald}^l)(\text{fred}^l)$ . We say that this semantic representation is ‘correct’ reflecting that it can combine with the verb  $\lambda X.\lambda Y.\text{praised}^l(X)(Y)$  with the correct result. We also require that this be guided by an appropriate syntactic process, i.e., functional composition of two type-raised categories  $S/(S\backslash NP)$  and  $(S\backslash NP)\backslash(S\backslash NP/NP)$  with the result,  $S\backslash(S\backslash NP/NP)$ .<sup>10</sup>

The conditions described above can be stated in the following way (the notation  $Type_{fred^l}$  denote the syntactic type corresponding to  $fred^l$ , i.e.,  $S/(S\backslash NP)$  in the above example):

(212) a. There is some syntactic type  $Type_{N'}$  such that  $Type_{N_1} + Type_{N_2} = Type_{N'}$

There is some semantic representation  $N'$  such that  $N_1 + N_2 = N'$

b. Either of the following holds:

(i) “ $Type_{C_1} + Type_{N'}$ ” results in the correct syntactic type of the entire phrase *and*  $C_1 + N'$  results in the correct semantic representation of the entire phrase

(ii) “ $Type_{N'} + Type_{C_1}$ ” results in the correct syntactic type of the entire phrase *and*  $N' + C_1$  results in the correct semantic representation of the entire phrase

The condition (b) allows either direction because the position of  $C_1$  relative to the composition of  $N_1$  and  $N_2$  no longer corresponds to the surface order, and becomes ‘virtual’.

If the above conditions are not satisfied, this derivation is not available. Another possibility for the above example is the traditional derivation, “[**Fred**] [praised **Donald**]”. The conditions for this case is analogous, but the current implementation has a fail-safe case, which allows the result of the form  $\langle -, N'' \rangle$  where  $N''$  is the semantic representation of the entire phrase. That is, no contextual link survives the composition.

**Composition Type:**  $\langle C_1, N_1 \rangle_{C-N} + \langle -, N_2 \rangle$

This case corresponds to the example “[**Felix praised**] [**Donald**]”, e.g., in response to “What about Felix?”. The non-traditional constituent in this case is  $\langle \text{felix}^l, \text{praised}^l \rangle_{\text{felix-praised}}$  where  $\text{felix}^l$  and  $\text{praised}^l$  are contextual-link and non-contextual link, respectively, in that order at the surface. The condition is similar to the previous case except that in this case, the surface order between  $C_1$  and  $N'$  ( $N_1 + N_2$ ) is fixed.

<sup>10</sup>The detail of this composition is described in Subsection 4.3.3.

The conditions are thus specified as follows:

- (213) *a.* There is some syntactic type  $Type_{N'}$  such that  $Type_{N_1} + Type_{N_2} = Type_{N'}$   
 There is some semantic representation  $N'$  such that  $N_1 + N_2 = N'$
- b.* “ $Type_{C_1} + Type_{N'}$ ” results in the correct syntactic type of the entire phrase *and*  
 “ $C_1 + N'$ ” results in the correct semantic representation of the entire phrase

For this case, the traditional derivation “[Felix] [praised Donald]” is also possible. And, it is probably more natural in general. Thus, the analysis may end up with a spurious ambiguity. Elimination of spurious ambiguity involving structured meanings will be discussed at the end of this subsection.

**Composition Type:**  $\langle C_1, - \rangle + \langle C_2, - \rangle$

The last case examined here is a composition of two contextual links. For example, consider an example  $\langle praised', - \rangle + \langle donald', - \rangle$ . This would result in  $\langle \lambda Y.praised' (donald') (Y), - \rangle$  if  $\lambda Y.praised' (donald') (Y)$  is indeed a contextual link. We enforce this requirement (146*b*) as follows:

- (214) *a.* “ $Type_{C_1} + Type_{C_2}$ ” results in the correct syntactic type  
 “ $C_1 + C_2$ ” results in the correct semantic representation  $C'$
- b.*  $C'$  is a contextual link.

Then, the resulting structured meaning is  $\langle C', - \rangle$ . Otherwise, the current implementation assumes  $\langle C_1, C_2 \rangle_{C-N}$ , but not  $\langle C_2, C_1 \rangle_{N-C}$ . This is a disambiguation heuristic and a weak form of ‘theme-first’ principle. In practice, when both the subject and the predicate are contextual links (and thus either can be a theme), this heuristics appears as choosing the subject as a theme.

Other cases discussed in Section 4.3.1 are analogous.

### Structured Meaning and Spurious Ambiguity

Integration of structured meaning with our CCG parser complicates the situation involving spurious ambiguity. The elimination method based on mutual subsumption needs to be redefined because the comparison between the result semantic representation does not reflect potential difference in structured meaning. The adopted solution is to apply mutual subsumption check to each

component of structured meaning. For example, to compare  $\langle C_1, N_1 \rangle$  and  $\langle C_2, N_2 \rangle$ , mutual the subsumption of  $C_1$  and  $C_2$  and that of  $N_1$  and  $N_2$  are checked.

When both of the involved structured meanings are the type of  $\langle C, - \rangle$  or  $\langle -, N \rangle$ , the case reduces to the original mutual subsumption check. Although components  $C$  and  $N$  in  $\langle C, N \rangle$  may consist of discontinuous elements, the proposed method is along the same line with the original mutual subsumption check, which ignores the syntax.

Since we have estimated that the practical maximum of distinct structured meanings for a category is 16 (p. 114), we also expect that each category may correspond to up to 16 structured meanings. But integration of structured meaning even with the presence of spurious ambiguity does not in practice introduce an exponential explosion.

### 6.3.4 Identification of Information Structure

As discussed in Subsection 4.3.2, once we adopt the structured meaning approach, its identification of information structure is almost trivial. At the last semantic composition, simply retrieve  $C$  and  $N$  from  $\langle \begin{matrix} C \\ \text{contextual link} \end{matrix}, \begin{matrix} N \\ \text{non-link} \end{matrix} \rangle$ . But there is one procedural aspect we should consider here.

In Subsection 2.3.4 (p. 42), we have mentioned the possibility of accommodated theme. The following is the example used there.

(215) *Q*: Who did Felix praise?

*A*<sub>1</sub>: [Felix praised]<sub>Theme</sub> [Donald]<sub>Rheme</sub>. (direct response)

*A*<sub>2</sub>: [Felix]<sub>Theme</sub> [praised Donald]<sub>Rheme</sub>.

*A*<sub>3</sub>: [Felix praised Donald]<sub>Rheme</sub>.

The current implementation adopts a heuristic to pick up only the possibility (*A*<sub>1</sub>), corresponding to the maximal theme. This is achieved by checking the dominance relation between categories. This way, only the themes that are not dominated by another survive. This process does not guarantee a unique theme, though. There may be incomparable maximal themes, e.g., the subject and the object where the verb is a non-contextual link.

In order to choose the most likely particle based on the information structure, we adopt an additional heuristic. The system prioritizes the patterns of information structure according to the conditions below. They are arranged from the highest priority to the lowest.

- (216) *a.* The subject + verb is the theme. Thus, the subject is a part of the theme. (Code 12)
- b.* The predicate is the rheme. Thus, the subject is the theme. (Code 49)
- c.* The predicate is the rheme and is one-place and negative. Thus, the subject is a contrastive theme. (Code 50)
- d.* The predicate is the theme. Thus, the subject is the rheme. (Code 51)
- e.* The adjectival predicate is the rheme. Thus, the subject is the theme. (Code 82)
- f.* The subject + verb is the rheme. Thus, the subject is a part of the rheme. (Code 88)
- g.* The verb is the rheme. Thus, the subject is a part of the theme. (Code 91)
- h.* The verb is the theme. Thus, the subject is a part of the rheme. (Code 99)

The general idea is to choose a larger theme. So far, no obvious errors have been observed due to the above prioritization.

### 6.3.5 Prediction of Particle Choice in Japanese

The last step of the automatic process is prediction of particles in Japanese. Following the analysis in Chapter 5, the procedure simply predicts *wa* for the matrix-level subject that is a part of the theme, and *ga* otherwise.

The prediction procedure in Chapter 5 includes special cases such as parallel clauses, one-place negative predicates, and one-place stage-level predicates. Among these, only the case of one-place negative predicate has been implemented.<sup>11</sup> The other two cases may result in incorrect predictions. In particular, our lexicon does not yet reflect stage and individual-level distinction. There is one case where stage-level predicates appear in a coordination.

For the case of purely semantically-conditioned contrastive *wa*, there is no way of mechanically identifying them. But we have seen that very few of them are rhematic through the mini-corpus analysis in Subsection 5.2.3 and that there is none in our experimental data. Thus, we do not consider the case where a contrastive *wa* must be chosen within a rheme in place of *ga* marking.

To identify the matrix-level grammatical subject, we adopt a definition of subject as the least oblique argument of a predicate [Steedman, 1996, p. 21]. The system first checks for all-theme

---

<sup>11</sup>Although this case is implemented for the process of identifying information structure, it is not used in the evaluation process.

$\langle C, - \rangle$  and all-rheme  $\langle -, N \rangle$  cases. Once these possibilities are excluded, the system identifies the presence of a modifier-clause relation between  $C$  and  $N$  in  $\langle \underset{\text{contextual link}}{C}, \underset{\text{non-link}}{N} \rangle$ . After excluding the clausal modifier, if any, the process checks the matrix-level predicate-argument structure and detects whether the subject, the least oblique argument, is in  $C$  or  $N$ . Depending on whether the subject is in  $C$  or  $N$ , the system identifies its theme/rheme status and thus predicts *wa* or *ga*. The semantic representation of adjectival and passive predicates are treated similarly. When a *by*-phrase is present in a passive construction, it is placed as an argument more oblique than the subject.<sup>12</sup>

The actual output of the program looks like the following. The listing is to demonstrate the state of implementation, the detail does not concern us.

```
>>>> Utterance Number: 12-1 <<<<<

Seg: Osteoporosis in Active Women : Prevention , Diagnosis , and treatment (11 words)

Preprocessed: Osteoporosis in Active Women : Prevention < and [[ Diagnosis ]] and
> Treatment (14 words)

Result: cat(bas(np(com),[colon=yes]),osteoporosis-(pat_agt-(woman:pl//active-woman:pl))//colon(and:[prevention,diagnosis,treatment]),[nil,181,id])

Result: cat(bas(s(fin),_28902)-(/,bas(s(fin),_28902)-( bas(np(com),[colon=yes]))),((osteoporosis-(pat_agt-(woman:pl//active-woman:pl))//colon(and:[prevention,diagnosis,treatment]))^_28873)^_28873,[nil,182,id])

Number of parses: 2

CPU time: 1330 ms Elapsed: 1600 ms

>>>> Utterance Number: 12-2 <<<<<

Seg: Osteoporosis has been defined as ‘ ‘ a disease characterized by low bone mass and microarchitectural deterioration of bone tissue , leading to enhanced bone fragility and a consequent increase in fracture risk . ’ ’ (36 words)

Preprocessed: Osteoporosis has been defined as ‘ ‘ a disease characterized by low bone mass and microarchitectural deterioration of bone tissue , leading to enhance
```

---

<sup>12</sup>One evidence for this position is the binding phenomenon, e.g., “Felix is praised by himself” vs. “\*Himself is praised by Felix”. This suggests that the subject of a passive is in a ‘commanding’ position in whatever the structure assumed for binding process, e.g., predicate-argument structure in our case.

d bone fragility and a consequent increase in fracture risk . ' ' (34 words)

```
Result: cat(bas(s(fin),[be_verb=no]),aux(perf)-(define-risk-(as-and:[indef-(disease//characterize-(by-and:[mass-(? -bone)//low-(mass-(? -bone)),deterioration-(of-(tissue-(? -bone)//lead-(to-(fragility-(? -bone)//enhance-(fragility-(? -bone))-2371))-(tissue-(? -bone))))//microarchitectural-(deterioration-(of-(tissue-(? -bone)//lead-(to-(fragility-(? -bone)//enhance-(fragility-(? -bone))-2371))-(tissue-(? -bone))))])]-disease),indef-(increase-(in-fracture)//consequent-(increase-(in-fracture))))]-osteoporosis),[40,2951,cn])
```

*(54 other parses omitted)*

Number of parses: 55

CPU time: 218370 ms Elapsed: 331630 ms

\*\*\* IS analysis:

```
- Theme(osteoporosis/40):Rheme(_3268^(aux(perf)-(define-risk-(as-and:[indef-(disease//characterize-(by-and:[mass-(? -bone)//low-(mass-(? -bone)),deterioration-(of-(tissue-(? -bone)//lead-(to-(fragility-(? -bone)//enhance-(fragility-(? -bone))-3174))-(tissue-(? -bone))))//microarchitectural-(deterioration-(of-(tissue-(? -bone)//lead-(to-(fragility-(? -bone)//enhance-(fragility-(? -bone))-3174))-(tissue-(? -bone))))])]-disease),indef-(increase-(in-fracture)//consequent-(increase-(in-fracture))))]-_3268)/2951)
```

*(1 another information-structure analyses omitted)*

=> Particle prediction (matrix subject): >>wa<< (case 49)

>>>> Utterance Number: 12-3 <<<<<

Seg: Although anyone can develop osteoporosis , postmenopausal women and young females with menstrual irregularities are most commonly affected . (19 words)

```
Result: cat(bas(s(fin),[]),affect-and:[woman:pl//postmenopausal-woman:pl,female:pl-(prop-(irregularity:pl//menstrual-irregularity:pl))//young-(female:pl-(prop-(irregularity:pl//menstrual-irregularity:pl)))]-_61406//most//commonly//although-(aux(can)-(develop-osteoporosis-pron(anyone))),[3288,3440,cn])
```

*(2 other parses omitted)*

Number of parses: 3

CPU time: 4430 ms Elapsed: 7320 ms

```
*** IS analysis:
- Theme(_49161^(_49161//although-(aux(can)-(develop-osteoporosis-pron(anyone))))/
3288):Rheme((affect-and:[woman:pl//postmenopausal-woman:pl,female:pl-(prop-(irregu
larity:pl//menstrual-irregularity:pl))//young-(female:pl-(prop-(irregularity:pl//m
enstrual-irregularity:pl)))]-_49116//most//commonly)/3440)

=> Particle prediction (matrix subject): >>ga<< (case 89)
```

### 6.3.6 Potential Applications to Generation

Before concluding this chapter, let us briefly discuss the possibility of applying the identification process (of information structure) to natural language (NL) generation. As implemented in Prevost [1995] and Hoffman [1995], the basic idea is that certain linguistic forms are associated with either the theme or the rheme. Once we identify the information structure of an utterance, we can eliminate linguistic forms incompatible with the identified information structure. Here, we consider two examples. Text-to-speech generation in English and English-Turkish machine translation.

For the case of text-to-speech generation, we can identify the information structure of the utterances in the text. As we have discussed earlier, certain pitch accents are associated with the theme and the rheme [Steedman, 1991a], e.g., L+H\* for a theme and H\* for a rheme. Depending on whether a contrast falls within a theme or a rheme, we can predict the appropriate intonation. This process of predicting intonation applies to arbitrary word class, cf. particle choice for the matrix subject in Japanese.

In English-Turkish machine translation, we may adopt the function of word order in relation to information structure. For example, following Hoffman [1995], we may identify the utterance-initial element as a theme and the pre-verbal element as a rheme. Once we identify the information structure of the utterances in the texts in English, we can choose an instance of word order in Turkish that is consistent with the identified information structure. Thus, for the type of NL generation where the input is a text, the current approach can provide useful information for generating contextually appropriate linguistic forms with respect to information structure.

## 6.4 Summary

In this chapter, we demonstrate that our CCG parser performs reasonably well for the purpose of information-structure analysis. The most critical element in the implementation is elimination of spurious ambiguity. We show that the semantic equivalence check can be extended to the case where structured meanings are also involved.

The module that analyzes information structure is realized as a straightforward implementation of the specification of contextual link and integration of structured meaning. In addition, several procedural aspects are addressed and integrated in a modular fashion. This allows us to upgrade the system with new specification for these procedural aspects.

Note: CL for a contextual-link status and NL for a non-contextual-link status

- **Initial context** (once at the beginning of a text): Set CL for the following:
  - Pronouns and situation words: *he, it, such, these, they, this, those, today, we*
  - Nouns available as domain-specific knowledge: *physician, clinician, patient*
- **Lexical processing** (for each lexical instance):
 

Set CL for the following:

  - Function words: modals, prepositions, etc. (specified as `cont=(func,FuncType)`)
  - Two-place nouns (specified by the feature `implicit_arg=req`) with no argument
  - Entries sharing the content information (specified as `cont=(cont,Class:Content)`) with a contextually-linked category (i.e., morphological variation)

Set NL for numerals (specified by the category `num`)

Designate the “project from itself” status for a denominal adjective with `denom=yes`
- **Composition** (two categories):
  - Special case: ignore dummy categories (e.g., punctuation) and function words as arguments; set CL for composition of two function words
  - Set CL status to the result of the following:
    - \* Definite determiner: for  $X_{[def=yes]}/Y + Y$
    - \* Indefinite generic: for  $X_{[def=no]}/Y + Y_{CL}$
    - \* Utterance-initial modifier: if the result is  $S/S$
    - \* Inverted phrase:  $S_{[inv=yes]}/X + X$
  - Set NL status to the result of the following:
    - \* Indefinite article: for  $X_{[def=no]}/Y + Y$
    - \* Numeral: for  $X/Y_{[cl=reset]} + Y$
  - Project the contextual-link status from an argument to the result:
 
$$\frac{X}{Status} \dots / Y_{[cl=proj(arg)]} + \frac{Y}{Status} \text{ (also directional variations)}$$
  - Project the contextual-link status from itself:
 
$$X \dots / Y_{[cl=proj(self)]} + Y \text{ (also directional variations)}$$
  - Project the contextual-link status either from an argument or from itself: utterance-initial modifier with a CL subject, i.e.,  $S/S + S$  where both  $S/S$  and the subject of  $S$  are CL
- **Coordination** (three categories): Project CL if both conjuncts are CL, set NL otherwise

Figure 6.3: A Summary of the Procedure to Identify Contextual-Link Status