Lexical Realization in Natural Language Generation

Sergei Nirenburg, Rita McCardell, Eric Nyberg, Scott Huffman, Edward Kenschaft
Carnegie Mellon University

Irene Nirenburg
Carnegie Group Incorporated

Abstract This paper describes a procedure for lexical selection of open-class lexical items in a natural language generation system. An optimum lexical selection module must be able to make realization decisions under varying contextual circumstances. First, it must be able to operate without the influence of context, based on meaning correspondences between elements of conceptual input and the lexical inventory of the target language. Second, it must be able to use contextual constraints, as supported by collocational information in the generation lexicon. Third, there must be an option of realizing input representations pronominally or through definite descriptions. Finally, there must also be an option of using elliptical constructions. The nature of background knowledge and the algorithm we suggest for this task are described. The lexical selection procedure is a part of a comprehensive generation system, DIOGENES.

1 Our place on the generation research map.

Natural language generation is traditionally divided into two stages: the utterance planning ('what to say') stage and the lexical and syntactic realization ('how to say it') stage. The latter stage consists, essentially, of a large set of realization choices for the various meanings of the input, using the morphological, syntactic and lexical means of expression in the target language (TL). Research reported here deals with the process of lexical selection during this second stage of generation. Many of the existing generation systems have been conceived as components of natural language interfaces to database systems. In such generators the lexical inventory can be strongly constrained without jeopardizing the quality of the interaction (cf, e.g., McKeown, 1985). Such systems necessarily concentrate on choosing appropriate TL syntax - indeed, generators are expected to produce adequate syntactic structures. Lexical selection becomes more important when it is difficult to constrain the types of output in generation, and, consequently, when the lexicon becomes large. Machine translation and automatic text summarization are among applications that by nature require a wide range of outputs and have to use a sizable lexicon. Note that of these two, the former does not involve utterance planning and concentrates on lexical and syntactic realization.

In the natural language generation community, attention to the task of lexical selection attention has recently grown. Of course, it has always been recognized as an important problem (cf. Danlos, 1984; Jacobs, 1985; Bienkowski, 1986; and the survey Cumming, 1986) and was addressed in a well-known early generation project (Goldman, 1975). However, most major NLG efforts of the early 1980s (e.g., Mann and Matthiessen, 1985; McDonald, 1983; McKeown, 1985, Appelt, 1985) concentrated more on syntactic and stylistic realization as well as to text planning than to lexical selection. Among those who dealt with lexical selection was the SEMSYN project (Rösner, 1986). Currently, work on lexical selection is going on at the University of Pennsylvania (e.g., Marcus, 1987) and at University of California at Berkeley (Ward, 1988). Still, the set of problems facing this field is significant. One motivation for our research was that we agree with Marcus (1987, p. 211) that at present 'most generation systems don't use words at all,' and we believe that the quality of generation output will improve significantly once an adequate lexical selection component becomes a standard part of a NLG system.

2 The Task

Research reported in this paper was performed within the DIOGENES project (Nirenburg, 1987), whose objective is to provide a high-quality generator for a knowledge-based interlingual machine translation system. The *input* to this generator is a set of a) world concept instances that represent the propositional content of the original text, and b) sets of text parameter values that represent its pragmatic content (see Section 4 for more details). In this paper we deal with a subset of the generation task, namely, the selection of open-class lexical items to realize the meanings of object, event and property tokens in the input. Thus, the *output* of the generation module described here is a lexical unit or a pronoun in the target language.

Our approach (and especially the expected input) to text generation is similar to that of the SEMSYN project (e.g., Rösner, 1986). Lexical selection is not, however, an immediate concern of and is not discussed at any length in SEMSYN descriptions (see Laubsch et al., 1984, p. 492), and a published analysis of practical difficulties encountered by the project (Hanakata et al., 1986) does not address this issue at all. Furthermore, since until very recently that project had to generate sentence-length texts (article titles), the problem of definite descriptions, pronominalization and ellipsis did not become acutely important.

3 Why is it a difficult task?

Lexical choice is not a straightforward task. Suppose we have to express in English the meaning 'a person whose sex is male and whose age is between 13 and 15 years.' What knowledge do people use in order to come up with an appropriate choice out of such candidate realizations as those listed in (1).

(1) *boy, kid, teenager, youth, child, young man, schoolboy, adolescent, man*

Without a sentential context the choice, based on closeness of the meaning match and generality of meaning, should be *boy.* For a computer program to be capable of making choices like this, it has to possess a preference-assigning capability on the matches between the meanings of the candidate lexical realization on the one hand and the input meaning unit (see the discussion of the matching metric below) on the other.

3.1 Collocations

Lexical choices are, however, typically made in context. Contextual relations among lexical units reflect meaning-induced constraints on cooccurrence (selectional restrictions: *admire* takes a human subject). Sometimes, however, it is difficult to formulate a cooccurrence constraint in terms of selectional restrictions alone. Thus, for example, the causative construction with the English *influence* requires *exert;* its Russian equivalent *vlijanie* requires *okazyvat'*, and the latter is not a Russian correlate of *exert* other than in the above and very few similar syntagmatic constructions. Why do we use, in English, *shed* with *tears* or *leaves* but don't usually say *shed water out of a bucket* or *they drop tears every time when* <...>? Such properties of the lexical stock of a natural language are called *collocational.* We will now illustrate the concept of collocation through several examples of 'semantic' or syntagmatic collocations.

Consider the conceptual operator of *a large quantity of,* a (relative) value for measuring quantities (of materials, forces, qualities, properties, etc.). It is realized in English in accordance with collocational properties of the lexical units that are used as its operands. Not every quantity goes with every realization of the above operator. Members of the set <*big, enormous, great, high, large, strong, wide*> of potential realizations of *a large quantity of* can cooccur with every one of the members of the set <*amount, difficulty, expanse, selection, voltage*> of quantities. We say *high voltage* but *a large amount.* It would be inappropriate for a generation system to produce something like *high selection* or *large difficulty.* (Note that in *parsing* the problem of assigning a similar semantic marker to all the various expressions from the example can, in principle, be tackled through a mechanism of metaphor processing, whereby a *general heuristic rule* is developed for processing metaphorical input belonging a single class, such as, for instance, *a large quantity of...* - see Lakoff and Johnson, 1980, for an extensive listing of potential metaphor classes; in generation, however, the task is the opposite - to *produce* fluent metaphorical language. Since this depends not on regularities of meaning, but rather on the idiosyncrasies of meaning realization in the various natural languages, the general rules will be more difficult to come by and formulate.)

An additional class of collocations are the paradigmatic collocations. These are best exemplified by the 'set-complement' collocations such as the English *left* and *right or parents* and *children.* The knowledge of these collocations, for instance, is necessary for the process of lexical selection of conjoined constructions, such as *ladies and gentlemen.*

Collocational relations are defined on lexical units, not meaning representations. The study of collocations ascends to Firth (1951); it is a central part of the *Meaning - Text* school of linguistics — cf. Mel'čuk, 1974; 1981. The importance of collocational properties in generations has been recognized (cf. Cumming, 1986), but relatively few systems actually include collocational information in their decision processes.

3.2 Ellipsis and Anaphora

Certain contexts completely alleviate the problem of open-class lexical selection. Consider the following (gloss of an) input segment

(2) Clause$_1$: Buy(John$_3$ book$_7$), time$_1$, focus: book$_7$
Clause$_2$: Bring(John$_3$ book$_7$ office$_1$), belong-to(office$_1$ John$_3$),
time$_2$: time$_2$ > time$_1$, focus: office$_1$
Clause$_3$: Read(John$_3$ book$_7$), aspect: inchoative, time$_3$: after(time$_2$)

One of the adequate ways of realizing it is:

(3) *John bought a book. He brought this book to his office and started to read it.*

There are seven instances of the three object-type concepts in the case-role slots of the input propositions above. Each of the three concepts is realized lexically only once. In two cases these meanings were realized through pronominalization and in one each through definite description and an elliptical construction. This example shows that non-lexical realization is an integral part of the process of lexical selection in generation.

In what follows we briefly describe the system architecture, the knowledge structures and the algorithm we use for selecting open-class lexical items in generation.

4 The System and the Knowledge

DIOGENES is a distributed natural language generation system featuring a blackboard-type control structure. The processing in it is concentrated in the *knowledge sources* which are triggered by the state of the various blackboards. The latter contain the *input* to generation as well as all intermediate and final results of DIOGENES operation, represented uniformly in the frame-oriented knowledge representation language FrameKit (Carbonell and Joseph, 1985 and Nyberg, 1988). The implementation vehicle for DIOGENES is CMU CommonLisp running on an IBM PC RT. (Description of the DIOGENES control structure and data handling is beyond the scope of this paper.) Background knowledge in DIOGENES includes the following components relevant to the task of lexical selection:

- a *concept lexicon,* a set of knowledge structures that describe object and event-types in the (sub)world of the texts to be generated (the first application of DIOGENES is, for example, in the domain of computer hardware manuals)
- a *generation lexicon* that links (sub)world concepts (or, more accurately, their instances) with particular lexical units of the target language.

The above description is necessarily incomplete. See Nirenburg, 1987 for an extensive specification of all the facets of DIOGENES.

The input to DIOGENES, known as interlingua text (ILT), is a set of FrameKit frames representing the propositional and non-propositional meanings of the source language text input into the translation system (see Nirenburg et al., 1986, 1987b; Nirenburg and Carbonell, 1987 for a detailed description). The ILT is produced during the analysis stage of the MT process by the parser, the semantic interpreter and, if needed, with human help. This latter stage is needed because our requirements for the input to generation are such that no current analysis system can produce them in a completely automatic fashion.

The following is a sample InterLingua Text.

```
(make-frame clausel
  (clauseid (value clausel))
  (propositionid (value propositionl))
  (speechactid (value speech-actl)))

(make-frame propositionl
  (propositionid (value propositionl))
  (clauseid (value clausel))
  (process-type (value action))
  (is-token-of (value *throw))
  (aspectid (value aspectl))
  (space (value propositionl.space))
  (time (value (at propositionl.time)))
  (subworld (value everyday-world))
  (modality (value real))
  (manner (value quickly))
  (agent (value role1))
  (object (value role2))
  (instrument (value role3))
  (source (value role1))
  (destination (value role4)))

(make-frame aspect1
  (phase (value end))
  (iteration (value 1) )  ;in seconds
  (duration (value 1)))

(make-frame speech-act1
  (speech-act (value assertion))
  (direct? (value no))
  (speaker (value author))
  (hearer (value reader))
  (time (value speech-act1.time)))

(make-frame role1
  (roleid (value role1))
  (clauseid (value clause1))
  (comment (value "a young boy"))
  (referent (value person1))
  (description (value role1))
  (is-token-of (value *person))
  (age (value (10 11)))
  (sex (value male)))

(make-frame role2
  (roleid (value role2))
  (clauseid (value clause1))
  (comment (value "a green rubber ball"))
  (referent (value ball1))
  (description (value role2))
  (is-token-of (value *ball))
  (color (value green))
  (made-of (value *rubber)))

(make-frame role3
  (roleid (value role3))
  (clauseid (value clause1))
  (comment (value *arm*))
  (referent (value arm1))
  (description (value role3))
  (is-token-of (value *arm))
  (part-of (value role1)))

(make-frame role4
  (roleid (value role4))
  (clauseid (value clause1))
  (comment (value "a big fast car"))
  (referent (value vehicle1))
  (description (value role4))
  (is-token-of (value *vehicle))
  (medium (value road))
  (propulsion (value fuel))
  (wheels (value 3   4 ) )
  (size (value big))
  (velocity (value (50 1 8 0 ) ) ) )
```

Given the above ILT, DIOGENES produces the text

The young boy quickly threw a green rubber ball at a big fast car.

```
GL-entry              ::= ( <meaning-pattern> <TL-pattern>* )
<meaning-pattern>     ::= ((is-token-of (value <CL-concept>))
                           [(<relation> (value <value>*)
                           (importance <importance-value>))]* )
<CL-concept>          ::= {any concept in the Concept Lexicon}
<relation>            ::= {any relation from the Concept Lexicon}
<value>               ::= {any concept or attribute (scale) value
                            in the Concept Lexicon}
<importance-value>    ::= 1 | 2 | ... | 10
<TL-pattern>          ::= (<TL-lexeme> <lex-info> <collocation> )
<TL-lexeme>           ::= (<language>
                            TL-lexical-unit | (synonym TL-lexical-unit*))
<language>            ::= english | spanish | russian | japanese | ...
<lex-info>            ::= ((<syntactic-info>) (morph <inflection-type>))
<syntactic-info>      ::= {the contents of a syntactic dictionary
                             (cf. e.g. Ingria, 1987)}
<inflection-type>     :: = {an indication of irregularities in forming word
                            forms, e.g., goose - pl. geese}
<collocation>         ::= ( {<dimension> <dimension-value>*} * )
<dimension>           ::= {the name of a lexical collocation relation
                            based on the Concept Lexicon slot names
                            for the concept in question}
<dimension-value>     ::= {a TL lexical unit (word or expression) that can
                            ordinarily collocate with the TL lexical unit in
                            <TL-lexeme> above and connected to the TL unit
                            on a specified dimension; can be recursive}
```

Figure 1: The Structure of the Generation Lexicon

4.1 The Generation Lexicon

The main static knowledge source for the generating of open-class items is a specialized generation lexicon (GL). The structure of an entry in the generation lexicon in DIOGENES is shown in Figure 1 (the BNF is incomplete wherever obvious).

The *importance value* serves to distinguish the saliency of the various relations for the identity of the entry head. Thus, for instance, generating *youth* instead of *boy* seems to be less a deviation than generating *girl.* This is why the importance of the *sex* slot in the example below is greater than that of the *age* slot.

(The sample GL entries below do not contain a full complement of collocation relations.)

```
(make-frame boy
  (is-token-of (value *person))
  (sex (value male)
       (importance 10))
  (age (value (2 15))
       (importance 4))
  (lexeme (value "boy"))
  (syntactic-info (lexical-class noun)
                  (noun-type class))
  (morphological-info (plural regular))
  (para-collocation (synonym lad kid child)
    (antonym girl adult)
    (hypernym person))
  (syn-collocations-in (value boy.syn)))
```

```
(make-frame boy.syn
  (agent-of (value play throw run jump)
    (strength 0.5))
  (place (value school playground ball field)
            (strength 0.5)))


(make-frame toss
  (is-token-of (value *throw))
  (direction (value up)
     (importance 3))
  (altitude (value (70 150))) ;in feet
    (importance 3))
  (velocity (value (0 30))) ;in miles/hour
    (importance 9))
  (lexeme (value "toss"))
  (syntactic-info (lexical-class verb)
                 (verb-type transitive))
  (morphological-info (past-tense regular)
                    (past-participle regular))
  (para-collocation (antonym catch)
                   (synonym cast propel toss fling hurl pitch))
  (syn-collocations-in (value toss.syn)))

(make-frame toss.syn
  (agent (value gambler)
  (strength 0.7) )
  (object (value coin)
          (strength 0.7)))


(make-frame fast
  (is-token-of (value *velocity))
  (velocity (percent-of-range (51 80)))
  (lexeme (value "fast"))
  (syntactic-info (lexical-class adjective))
  (morphological-info (comparative regular)
                     (superlative regular))
  (para-collocation (antonym slow)
                   (synonym quick)))

(make-frame car
  (is-token-of (value *vehicle))
  (medium (value road)
          (importance                                8))
  (propulsion (value fuel)
              (importance 10))
  (wheels (value 3 4)
          (importance 9))
  (lexeme (value "car"))
  (syntactic-info (lexical-class noun)
                 (noun-type class))
  (morphological-info (plural regular))
  (para-collocation (synonym automobile auto))
  (syn-collocations-in (value car.syn)))

(make-frame car.syn
  (object-of (value dent)
             (strength 0.8))
  (size (value big)
        (strength 0.8))
  (velocity (value fast)
            (strength 0.8)))
```

4.2. The Matching Metric

The ILT frames and the meaning patterns of GL entries against which they are matched are collections of slots whose fillers are members of domains (or value sets) predefined for each slot.

The slot fillers can be .symbols, e.g., `(temperature (value tepid))` or *numbers, e.g.*, `(weight (value 141.5))`.

The domains (or value sets) from which slot filler values are taken can be *unordered,* ordered discretely (hereafter, *ordered)* or ordered continuously (hereafter, *continuous).* An example of an unordered set would be `occupations`, an ordered set `months-of-the-year` and a continuous set `height`. Note that only the slot fillers of a continuous set can contain numeric values.

The cardinality of slot fillers in meaning patterns can be *single* as in `(sex (value female))`, *enumerated* as in `(subjects (value physics math history))` or *ranges* as in `(age (value 0 100))`.

The matching metric (see Nirenburg *et al.,* 1987a) is designed on the basis of the following heuristics:

- If the types of slot fillers or the domains of the filler values in the ILT and GL meaning pattern differ, then the match is declared inadequate and a maximum distance is assigned.
- The quality of the match is proportional to the size of the intersection of the actual slot fillers in the ILT and GL frames, which is then modified by the size of the domain itself.
- Each slot in a GL meaning pattern is rated with respect to its importance for the meaning expressed by the frame. A mismatch on a less important slot will have a smaller influence on the overall score of the match.
- The quality of a match between two frames is a weighted sum of the quality of the matches of the individual frame slots.

For the matching program, it is assumed that the ILT frame contains all the slots found in the GL meaning pattern. If this is not the case, it is assumed that a special module in the generator (called the *augmentor,* see Nirenburg, 1987) will infer the necessary slots and fill them with default values that are stored in the IL ontological hierarchy, i.e., the Concept Lexicon. Conversely, the ILT frame can (and regularly will) contain more slots than the GL frame. This is due to the fact that an ILT frame corresponds not to a *single* lexical unit, but rather to an *entire* phrase (such as a noun phrase, complete with the various modifiers). Thus, lexical selection proceeds in two phases: first, the heads of the phrases (typically, nouns and verbs) are generated and then the modifiers (typically, adjectives and adverbs, but frequently such recursive elements as prepositional phrases and relative clauses) are selected.

4.3 The Matcher

The matching program is the actual computer implementation of the heuristics listed in the previous section. It matches an ILT frame's meaning pattern against the meaning patterns of GL entries to determine the appropriate lexical item for the frame. The components of the GL entry that the matching program uses are the meaning pattern section (which is a frame in the ILT format) and the TL string. A modified GL entry can be seen in Figure 2.

The meaning pattern portion of a GL entry is an instance of a frame, containing the specific slot fillers that should be present for the lexical item to be appropriate.

Since the matching program performs flexible matching, it must decide (unless there is a perfect match) exactly how far the input ILT frame is from a perfect match with the GL pattern. After all GL entries for a given concept (or token) such as *person* have been examined in this fashion, the one with the best score is selected. All other mismatches are assigned penalties proportional to their "imperfectness".

```
((is-token-of (value *person))
 (sex  (value female)
       (importance 10))
 (age  (value (0 12))
      (importance  6))
 (lexeme girl)
)
```

Figure 2: The GL Meaning Pattern for *girl*

In this section, the actual empirically derived scoring function is described, along with the algorithm that uses the scoring function to create an aggregate score for all the slots in the ILT input frame and the particular GL entry.

For each slot which is present in *BOTH* the ILT and the GL entry, the fillers are compared. If the fillers are equal, then the slots match perfectly, and no penalty is assigned. If the fillers do not match perfectly, a penalty is computed for that slot. The function used to calculate the penalty depends on the type of slot filler (e.g., SINGLE, ENUMERATED, or RANGE). Once the penalty is computed, it is weighted according to the importance of that slot to a successful match, and the weighted number is added to the cumulative penalty for the match thus far.

The scoring for the various combinations of domain sets (there are 3 sets) and the cardinality of slot filler types (there are 3 types) are discussed in the next five sections. Note, however, that out of a maximum of 9, there are only 7 *meaningful* combinations:

- Unordered, Single
- Unordered, Enumerated
- Ordered, Single
- Ordered, Enumerated
- Ordered, Range
- Continuous, Single
- Continuous, Range

4.3.1 Ordered Domains, Single Element Fillers

An ordered domain with single element fillers can be exemplified by a set of single `temperature`[1] values in Figure 3.

Suppose we are trying to match TEMPERATURE slots in an ILT frame and a GL entry:

```
ILT: (temperature (value cool) )
GL: (temperature (value lukewarm))
```

The penalty assigned to a mismatch depends on two variables:

- D: the distance between the fillers in the *ordered* set of values

[1] Temperature is one of the physical object attributes within the ontological *is-a* hierarchy from which ILTs are produced (Nirenburg *et al.,* 1987b).

```
(make-frame TEMPERATURE
  (instance-of (value slot))
  (element-type (value symbol))
  (domain-type (value ordered))
  (cardinality (value single))
  (elements (value (cold cool tepid lukewarm warm hot scalding))))
```

Figure 3: The FrameKit Attribute Entry for TEMPERATURE

• C: the size of the domain of values

The quality of the match depends, of course, on the distance between the two fillers; however, it is important to consider the size of the filler domain as well. The larger a domain, the higher the chance that the two fillers will diverge. Consequently, the penalty is lessened for larger domains, the distance between the fillers being equal. (If there are only two elements in the domain, there is a 50% chance of the correct filler appearing in the input, so a larger penalty is assigned if it does not.)

A simple equation relates these two variables:

$$P = \frac{W \times D}{\mathcal{F}(C)} \tag{1}$$

where W is a numerical weight on the distance between the fillers and $\mathcal{F}$ is a damping function on the size of the domain. These parameters are added to the equation for calibration purposes (calibration is discussed in Section 4.3.7).

4.3.2 Unordered Domains, Single Element Fillers

The matching of single element fillers from unordered domains is quite similar to matching single filllers in ordered domains. The difference lies in the fact that the distance between the fillers is no longer meaningful, since they are not ordered in any way with respect to one another. The penalty assigned to the match becomes a function merely of the size of the domain (and hence the probability of the correct filler appearing):

$$P = \frac{W}{\mathcal{F}(C)} \tag{2}$$

4.3.3 Continuous Domains, Single Element Fillers

As stated before, only numbers can be represented in a continuous domain. The elements of the domain are defined by giving the endpoints of the domain (or closed interval) and the unit size of representation to be used in computing the distance between fillers. When defined in this manner, equation 1 can be used to compute the penalty for continuous domain sets as well. As an example, consider an alternate definition of the attribute TEMPERATURE in Figure 4.

As before, suppose we are trying to match temperature slots between an ILT and a GL frame:

```
ILT:(temperature (value 93.7))
GL: (temperature (value 98.6))
```

Before evaluating equation 1, both the distance between the two fillers and the distance between the endpoints of the closed domain interval are computed using the defined unit-size value. Then the evaluation of equation 1 proceeds as before.

```
(make-frame TEMPERATURE
  (instance-of   (value  slot))
  (element-type   (value number))
  (domain-type   (value  continuous))
  (cardinality   (value  single))
  (unit-size   (value  1))
  (elements   (value   (0  100))))
```

Figure 4: A *Continuous* Attribute Entry for TEMPERATURE

4.3.4 Ordered Domains, Range/Enumerated Fillers

Regardless of the domain, the function used to calculate the penalty for mismatched range fillers is identical to that for enumerated fillers. In the following discussion, it is assumed that the set operations of "size of filler" and "size of filler intersection" are executed directly on the enumerated fillers; the sizes and intersections of range fillers are calculated using the `unit-size` slot value. The abbreviation "r/e" is used to refer to the combination of both filler types, with the assumption that the appropriate preprocessing of ranges takes place prior to evaluation.

When two r/e fillers do not match exactly, several factors are taken into account when calculating the size of the penalty: the size of each filler, the size of the intersection between the fillers, and the size of the domain.

The calculation of the filler size is straightforward. However, the size of the intersection between the fillers is much more relevant, and a little bit more difficult to obtain. The intersection is conceptually equivalent to D (the distance between the fillers in equation 1). However, the intersection can occur in one of four ways:

1. The ILT filler and the GL filler overlap, and the resulting intersection is smaller than either filler.
2. The fillers overlap, and the ILT filler is contained within the GL filler.
3. The fillers overlap, and the GL filler is contained within the ILT filler.
4. The fillers do not overlap.

Although the actual integer that represents the number of shared elements can be calculated by the same function for 1, 2, and 3 above, it is important for the matcher to distinguish between 2 and 3. In general, a match where the GL filler is contained within the ILT filler is better than a match where the ILT filler is contained within the GL filler. (On a certain abstract level, this is equivalent to a match where some slots are present in the GL but not in the ILT.)

When the fillers do not overlap, the intersection is indicated by a negative number that represents the distance between the nearest endpoints of the two fillers.

The function used to calculate the penalties for ordered domains with r/e fillers employs the following variables:

- S_i: the size of the ILT filler
- S_g: the size of the GL filler
- I: the size of the intersection between the two fillers where negative values indicate the distance between disjoint fillers
- C: the size of the domain

- W_i: a weighting function dependent upon the ILT filler size
- W_g: a weighting function dependent upon the GL filler size
- W_{int}: a weighting function dependent upon the size of the intersection
- $F(C)$: a damping function dependent upon the domain size

The overall mismatch penalty (at the slot level) is computed by the following function:

$$P = \frac{W_i(S_i - I) + W_g(S_g - I) - W_{int}(I)}{\mathcal{F}(C)} \tag{3}$$

The first two terms in the numerator are designed to account for the two types of intersection in 2 and 3, above. If some of the ILT filler falls outside the intersection, then $S_i - I > 0$, and the first term will add to the penalty. If some of the GL filler lies outside of the intersection, the second term will add to the penalty. The determination of just how much is added in each case is dependent upon the weighting functions W_i and W_g. The penalty is also decreased by the size of the third term, which is the weighted intersection. (This counteracts the additive factor of I in the first two terms when $I < 0$.) Finally, the whole penalty is divided by the weighted size of the domain, which accounts for the varying probability of matching in larger domains.

4.3.5 Other Domain/Filler Types

The matching of an unordered domain with enumerated filler types and of a continuous domain with a range filler type are also scored using equation 3.

4.3.6 The Frame Level

Once all of the slots have been compared using the appropriate equation, the *overall* score for the match between the ILT frame and the GL entry is produced, according to the following formula:

$$S = \frac{\sum_{i=1}^{n} I_i \times P_i}{\sum_{i=1}^{n} I_i} \tag{4}$$

where the P_i's are the mismatch penalties between each slot of the ILT frame and the GL entry and the I_i's are the *importance values* for the corresponding slots of the GL entry meaning pattern. (The slots in an ILT frame do not contain importance values.)

4.3.7 Calibration

The functions in equation 3 can be calibrated to yield results with an intuitive interpretation. For example, for most applications, the W_g function should be quite low, since a subrange of a given range should reveal a valid instance of the concept. Conversely, the function W_i should be quite high, signifying an entry which does not fit into the specified range. However, the values which are suitable for any particular application must be produced empirically.

5 The Lexical Selection Algorithm

In the DIOGENES generator, an instantiation of a head-selecting knowledge source is triggered simultaneously for every *proposition* and *role* instance in the input The results of their operation are posted to a blackboard (the Lexical Selection BlackBoard, LSBB), so that all knowledge source instances can draw on this knowledge in their own decision processes. The knowledge sources responsible for selecting modifiers are triggered when the heads of their phrases have already been selected.

```
function lexical-selection (current-frame);

If anaphoric-realization-indicated (current-frame) = true
      then do-anaphoric (current-frame)
      else
  begin
            candidate-set    := GL-search    (current-frame GL) ;

       if cardinality  (candidate-set)   = 0
          then candidate-set   := interactively-augment-GL (current-frame) ;
       tryagain
       candidate-set   := collocationally-constrain (candidate-set) ;
       case cardinality (candidate-set)
            0: candidate-set  := interactively-augment-GL (current-frame);
                   goto tryagain;
            1: lexical-selection := candidate-set;
 otherwise: lexical-selection  := select-best  (candidate-set
                                                    current-frame)
   end;

if modifier-realization-indicated (current-frame lexical-selection) = true
     then modified-lexical-selection := select-modifier (current
                                                -frame lexical-selection);
```

Figure 5: The Lexical Selection Algorithm

```
function  GL-search  (current-frame GL) ;

slot-value   := get-is-token-of (current-frame);
candidate-set   := first-pass-search  (slot-value GL);
how-many  := cardinality  (candidate-set) ;

if how-many = 0
      then candidate-set :=interactively-augment-GL (current-frame GL)
      else candidate-set := union (candidate-set how-many) ;

post-intermediate-candidate-set (candidate-set) ;
```

Figure 6: The *GL-search* Algorithm

```
function collocationally-constrain  (candidate-set) ;

if number-of-potential-collocations (current-frame IBB) <> 0
        then collocationally-constrain := intersection
                                   (candidate-set neighbor.collocation.!frame)
     else
     begin
                If toss-a-coin   (current-frame)   = heads
                          then try-later
                          else collocationally-constrain := candidate-set;
     end;

post-intermediate-candidate-set (candidate-set) ;
```

Figure 7: The *collocationally-constrain* Algorithm

Figure 5 shows a simplified version of the top-level lexical selection algorithm. Extensions exist a) for the case when a change must be made in the lexical choice due to late evidence of collocational incompatibility, b) for selecting modifiers and c). for selecting definite descriptions in the anaphoric realization mode.

The predicate *anaphoric-realization-indicated* returns *t* if an input frame is co-referential with a previously instantiated IBB unit and when relevant heuristic rules determine that it should be realized pronominally, elliptically or through a definite description.

The function *do-anaphoric* determines the appropriate realization as above and posts it on the LSBB.

The function *GL-search* in Figure 6 searches the generation lexicon and produces a set of candidate lexical realizations by obtaining all entries that have the same *is-token-of* value that is found within the ILT frame.

It is clear that the acquisition of the generation lexicon is a major and extremely labor-intensive task. The acquisition of this dictionary, especially of the collocational information cannot at present be done automatically. But the efficiency of a team of lexicographers can be increased dramatically through the use of a specialized intelligent interactive aid, which will be both called through the *interactively-augment-GL* procedure during a generation session as well as being used for acquiring GL 'off-line.' We have developed one such *Knowledge Acquisition Maintenance System* (cf. Nirenburg et al., 1988), called ONTOS, for the acquisition of domain knowledge and are at present extending it for acquiring generation lexicons as well. In the first months of running the generator in every new domain, we expect to use the on-line augmentation facility extensively, until all the relevant entries in GL are covered.

The function *collocationally-constrain* in Figure 7 first uses the function *nunber-of-potential-collocations* to find out whether a lexical choice has already been made in the current proposition that can render some of the members of the current candidate set inappropriate collocationally. Collocational constraints are checked between a) the proposition head and all the role heads, b) heads and modifiers and c) all conjoined concept instances in the input. If no potential collocations are found, the current process can either wait and check again later or exit *collocationally-constrain* without any filtering actually achieved. (If there is no such provision there is a danger of a deadlock when all the processes will be in the waiting mode.) This eventuality is taken care of by rigging the probabilities of the two outcomes in *toss-a-coin*[2]. Heuristically, the choice of the main verb (proposition head) is the most independent, followed by the choice of noun phrase heads. *Collocationally-constrain* uses the collocation information in the lexicon entries to match

[2] This mechanism is similar to the one used in the *Friendly-Neighbors* algorithm of the PARPAR parallel parser (Lozinskii and Nirenburg, 1986).

and filter the candidate sets.

If the residual set has cardinality one, the result is posted on the LSBB. If there exist more than one candidate, the function *select-best* (i.e., the Matcher) is called to perform context-independent lexical selection based on the quality of the match between the meaning pattern of the ILT frame on the one hand and the *weighted* meaning patterns of the GL entries in the candidate set on the other.

The predicate *modifier-realization-indicated* returns *t* if there exist properties in the ILT frame that were not accounted for in the head realization of the lexical item, i.e., realized as either a noun or a verb, roughly speaking (see Figure 5).

The function *select-modifier* selects the appropriate modifier realization for the above phrase head, i.e., selecting an adjective for a noun and an adverb for a verb. This function uses essentially the same set of algorithms, i.e., *GL-search, collocationally-constrain* and *select-best,* for choosing phrase modifiers as was employed for selecting phrase heads. The primary difference between these sets of algorithms lies in the fact that the modifiers are constrained to collocate with only their respective phrase heads.

6 Status and Future Work

The blackboard architecture, the process handler and the inexact meaning matching module have been implemented; the collocation treatment module has also been implemented, but extensive testing has not been performed due to the lack of a large-scale lexicon. The anaphora treatment module has been implemented for pronominalization only. Our current tasks in the lexical selection module of DIOGENES thus include the acquisition of a sizable lexicon, complete with collocation information, extensions to the treatment of reference to incorporate definite descriptions, and the development of the module facilitating interaction between lexical and syntactic means of realization.

References

Appelt, D.E. 1985. **Planning English Sentences**. ACL Series — Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Bienkowski, M.A. 1986. A Computational Model for Extemporaneous Elaborations. CSL Report 1, Cognitive Science Laboratory, Princeton University.

Carbonell, J. and R. Joseph. 1985. FRAMEKIT, A Reference Manual. Department of Computer Science. Carnegie-Mellon University.

Cumming, S. 1986. The Lexicon in Text Generation. Paper presented at the LSA Linguistics Institute Workshop on Lexicon, New York.

Danlos, L. 1984. Conceptual and Linguistic Decisions in Generation. Proceedings of COLING-84. Stanford, pp. 501-504.

Firth, J.R. 1951. Modes of Meaning. In: J.R. Firth, **Papers in Linguistics**, London.

Goldman, N. 1975. Conceptual Generation. In: R. Schank (ed.), **Conceptual Information Processing**. Amsterdam: North Holland, pp. 289-372.

Hanakata, K., A. Lesniewski, S. Yokoyama. 1986. Semantic-Based Generation of Japanese-German Translation System. Proceedings of COLING-86. Bonn, pp. 560-562.

Ingria, R. 1987. Lexical Information for Parsing Systems: Points of Convergence and Divergence. In: D. Walker, A. Zampolli and N. Calzolari (eds.), **Automating the Lexicon: Research and Practice in a Multilingual Environment.**

Jacobs, P. 1985. A Knowledge-Based Approach to Language Production. Ph.D. Dissertation, University of California at Berkeley.

Lakoff, G. and M. Johnson, 1980. **Metaphors We Live By**. Chicago: University of Chicago Press.

Laubsch, J., D. Rösner, K. Hanakata, and A. Lesniewski. 1984. Language Generation from Conceptual Structure: Synthesis of German in a Japanese/German MT Project. Proceedings of COLING-84. Stanford, pp. 491-494.

Lozinskii, E. L. and S. Nirenburg. 1986. Parsing in Parallel. Journal of Computer Languages, Volume 11, Number 1, pp. 39-51.

Mann, W. and C.M.I.M. Matthiessen. 1985. Nigel: a Systemic Grammar for Text Generation. In: Freedle (ed.), **Systemic Perspectives on Discourse.** Norwood, NJ: Ablex.

Marcus, M. 1987. Generation Systems Should Choose Their Words. Proceedings of TINLAP-3. Las Cruces, NM, February.

McDonald, D. D. 1983. Natural Language Generation as a Computational Problem: an Introduction. In: Brady and Berwick (eds.), **Computational Problems in Discourse.** Cambridge: MIT Press.

McKeown, K. 1985. **Text Generation.** Cambridge: Cambridge University Press.

Mel'čuk, LA. 1974. **Towards a Theory of Linguistic Models of the Meaning-Text Type.** Moscow: Nauka.

Mel'čuk, I.A. 1981. Meaning - Text Models: A Recent Trend in Soviet Linguistics. **Annual Review of Anthropology.** Vol. 10, pp. 27-62.

Nirenburg, S., V.Raskin and A. Tucker. 1986. On Knowledge-Based Machine Translation. Proceedings of COLING-86. Bonn, pp. 627-632.

Nirenburg, S. 1987. A Distributed System for Language Generation. Technical Report CMU-CMT-87-102. Carnegie-Mellon University. May.

Nirenburg, S. and J. Carbonell. 1987. Integrating Discourse Pragmatics and Propositional Knowledge for Multi-Lingual Natural Language Processing. In: Computers and Translation, Number 2, pp. 105-116.

Nirenburg, S., E. Nyberg, and E. Kenschaft. 1987a. Inexact Frame Matching for Lexical Selection in Natural Language Generation. CMU-CMT Unpublished Memo.

Nirenburg, S., V. Raskin and A. Tucker. 1987b. The Structure of Interlingua in TRANSLATOR. In: S. Nirenburg (ed.), **Machine Translation: Theoretical and Methodological Issues,** ACL Series - Studies in Natural Language Processing. Cambridge: Cambridge University Press, pp. 90-113.

Nirenburg, S., I. Monarch, T. Kaufmann, I. Nirenburg and J. Carbonell. 1988. Acquisition of Very Large Knowledge Bases: Methodology, Tools and Applications. Submitted to the Second International Workshop on Knowledge Acquisition. Banff, Canada.

Nyberg, E. 1988. The FrameKit User's Guide. CMU-CMT Memo.

Rösner, D. 1986. When Mariko Talks to Siegfried. Proceedings of COLING-86. Bonn, pp. 652-654.

Ward, N. 1988. Issues in Word Choice. To appear in Proceedings of COLING-88, Budapest.