Choosing A Surface Realiser: Exploring the Differences in Using KPML/NIGEL and FUF/SURGE

Victor R. Essers and Robert Dale

Microsoft Research Institute,
Macquarie University,
Sydney NSW Australia

Abstract

Natural language generation (NLG) is a complex process, concerned with issues of both deciding what to say, and determining how best to express that content. One sub-task in NLG is to map a specification of the content of a sentence to a grammatically correct surface sentential form. This linguistic realisation task is addressed by a number of existing software packages developed within the research community, but each embodies subtly different assumptions about the nature of the task. Faced with a choice between these packages, this makes it difficult to determine which is best used in a given situation.

This paper presents an experiment which is part of a larger research program concerned with questions of modularity in the development of NLG systems. In the experiment, the two most well-known packages, KPML/NIGEL and FUF/SURGE, were used to provide realisation capabilities in the WEATHERREPORTER system, which generates short multi-sentential weather summaries.

We explore the differences in using the two realisers and conclude that, of the two systems, FUF/SURGE uses a more syntactically motivated approach to rhetorical constructs, so that the microplanning stage prior to surface realisation must determine appropriate ways to realise a rhetorical construct. On the other hand, KPML/NIGEL needs more access to external knowledge sources such as generalised and domain ontologies. We comment on the problem of imperfect grammatical coverage, and when semantically- or syntactically-appropriate constructs can be used in sentence plans. Finally, we conclude that KPML/NIGEL requires its user to have a better acquaintance than FUF/SURGE with the grammatical theory underlying the system, with mixed ramifications.

1 Introduction

The task of building a complete natural language generation (NLG) system is difficult and complex. Rather than start from scratch, it makes sense to make use of whatever reusable components are available, just as someone building a complete natural language analysis system would be likely to make use of an existing parser for syntactic analysis. In the context of NLG, the state of the art is such that a number of reusable components are available for that sub-task in the NLG process generally referred to as LINGUISTIC REALISATION.

The two most well known realisers are KOMET-PENMAN MULTILINGUAL (KPML) (Penman Natural Language Generation Group, 1989; Bateman, 1996) along with its associated grammar of English, NIGEL (Mann and Matthiessen, 1983), and the Functional Unification Formalism (FUF) (Elhadad, 1992; Elhadad, 1993) along with its associated grammar of English, SURGE (Elhadad and Robin, 1996).[1] Each realiser is a program which combines a broad coverage grammar of English with a mechanism for mapping from an input specification of a sentence (or

[1] Although in each case the realiser and its associated grammar are distinct components, we will for conve-

sub-sentential unit) to a surface form expressed as a text string that realises the sentence (or sub-sentential unit).

This brief description belies the differences between the two systems. Each uses a quite different processing mechanism to move from a sentence specification to a sentence: ideally, of course, this much should be invisible to the user who wishes to treat a realiser as a black box. There are also inevitable differences in grammatical coverage, reflecting the particular developmental paths the systems have taken. Much more importantly, however, even a cursory examination of the two systems shows that the inputs they expect are substantively different: each encompasses a different view as to the real nature of the linguistic realisation task. To our knowledge, these differences, and their impact on system construction, have not been explored overtly in the literature.[2]

This paper presents some work that is part of a larger program aimed at exploring possible modularities for processing in a natural language generation system: this larger research endeavour is an important one if we are to move towards genuine reusability in the field. Here, our most immediate concern is to identify arguments and justifications for what should be incorporated in the task of linguistic realisation and what should not. In order to explore this question, we have been using the two linguistic realisation packages mentioned above within the context of the same overall generation task, so that we can better compare the advantages and disadvantages that come with adopting either system's view of the linguistic realisation task. The aim of our experiment is to make it easier for other researchers faced with a choice between these two systems to determine which is most appropriate in their situation. This is not to say that one system is better than the other: which is best is likely to depend on many aspects of the context of use. Our work is intended as a step towards identifying just what those aspects of the context of use are, so that decisions can be made in a reasoned way.

The paper is structured as follows. In Section 2 we provide some background context, describing the WeatherReporter system and its architectural assumptions. In Section 3 we provide a brief overview of KPML/Nigel and FUF/SURGE, the two realisation packages we have used in this experiment. In Section 4 we sketch in general terms the issues that have to be considered in integrating an existing realiser into an NLG system like WeatherReporter, and in Section 5 we look at the specific concerns that are raised when integrating KPML/Nigel and FUF/SURGE in particular. Finally, in Section 6 we draw some conclusions and make some observations with regard to what needs to be done next.

2 Background: The WeatherReporter System

2.1 The WeatherReporter Architecture

WeatherReporter is being developed by the authors as a relatively simple NLG system that produces natural language text from an underlying database of numerical meteorological data collected automatically by weather-sensing devices. The idea of generating text from a numeric data source, as opposed to a sophisticated knowledge base, is not new and has been explored in a number of existing systems.[3] An important practical benefit of such systems is that many large numeric data sets are available, and could benefit from natural language reporting; whereas, at the current time, sophisticated large-scale knowledge bases are somewhat rare. Our aim in the WeatherReporter system is to explore issues of modularity that permit the easy transfer of NLG components from one domain of application to another.

nience frame our discussion in terms of realiser–grammar pairs, since this in practice is the way they are used. Although it is possible to provide the realisers with alternative grammars, most users are likely to make use of the grammars provided with the systems.

[2] At least, not by third parties: the developers of both systems have on various occasions made reference to specific points of contrast.

[3] Among these systems are ANA (Kukich, 1983), SEMTEX (Rosner, 1987), FOG (Bourbeau et al., 1990), LFS (Iordanskaja et al., 1992), PLANDOC (Kukich et al., 1993), STREAK (Robin, 1994), and SumGen (Maybury, 1995).

The system views natural language generation as being composed of three distinct subtasks, as Figure 1 illustrates. The DOCUMENT PLANNER is concerned with determining both

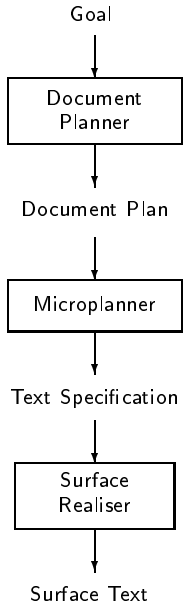

Figure 1: NLG System Architecture.

the content and the overall structure of the text to be generated. In the case of WEATHERREPORTER, this means selecting relevant information from the underlying numeric data source and packaging this into MESSAGES, a form of data representation we will return to below. These messages are placed by the document planner in a structure we call a DOCUMENT PLAN, which indicates any discourse relationships that hold between the messages.

The MICROPLANNER is concerned with mapping fragments of the document plan into structures that can be used as input to the SURFACE REALISER. In practice this means making decisions about how information should be packaged into sentence-sized chunks—the messages in the document plan do not necessarily correspond one-to-one to sentences—and also selecting appropriate lexical items and referring expressions to realise conceptual content in a contextually appropriate manner. The result is a tree structure called a TEXT SPECIFICATION which provides sufficient information for the final component in the process, the surface realiser, to produce well-formed output texts. Each leaf node in this structure is what we will refer to here as a SENTENCE PLAN. The point here is that we want to avoid, as far as is possible, having the microplanner reason about idiosyncracies of the particular natural language being generated; by encapsulating these idiosyncracies within the realisation component, we may begin to move towards a sensible modularity of the knowledge sources required in the generation task.

As is by now probably obvious, our interest in the present paper is to explore how the KPML/NIGEL and FUF/SURGE realisers can serve in the role of surface realisation components in this architecture.

2.2 Microplanning in WEATHERREPORTER

To focus our discussion, we will concentrate on one sentence within a larger text to be generated. Figure 2 shows a collection of three messages which have been constructed from data selected from the underlying information source by the document planner. Each message corresponds to a unit of information which could be, but won't necessarily be, realised as a separate sentence;

Figure 2 provides English glosses of what the resulting sentences might be if we pursued such a one-to-one mapping. The reason for maintaining a degree of abstraction between messages and sentence content is precisely to allow the microplanner to make decisions about the best way to package this information into sentence-sized chunks.

Message #1 *The month was warmer than average*

```
((message-id 1)
 (message-type monthlytemperature)
 (period ((month 6) (year 1994)))
 (absolute-or-relative relative-to-average)
 (relative-difference ((magnitude ((unit degrees) (number 3.0)))
                       (direction +))))
```

Message #2 *The month was wetter than average*

```
((message-id 2)
 (message-type monthlyrainfall)
 (period ((month 6) (year 1994)))
 (absolute-or-relative relative-to-average)
 (relative-difference ((magnitude ((unit millimetres) (number 10.0)))
                       (direction +))))
```

Message #3 *The month had a dry spell from the* 2^{nd} *to the* 5^{th}

```
((message-id 3)
 (message-type rainspellmsg)
 (period ((begin ((day 2) (month 6) (year 1994)))
          (end ((day 5) (month 6) (year 1994)))))
 (duration ((unit days) (number 5)))
 (amount ((unit millimetres) (number 0.0))))
```

Figure 2: The input WeatherReporter messages for the generation example.

The microplanner is assisted in this task by other information provided by the document planner. The output of the document planner is not just a bag of messages; as mentioned above, it is a tree structure that shows the discourse structural relationships that hold between these messages. For the purposes of the experiment described in this paper, we will assume that the document planner has already determined that the messages in Figure 2 are structurally related in the manner shown in Figure 3.

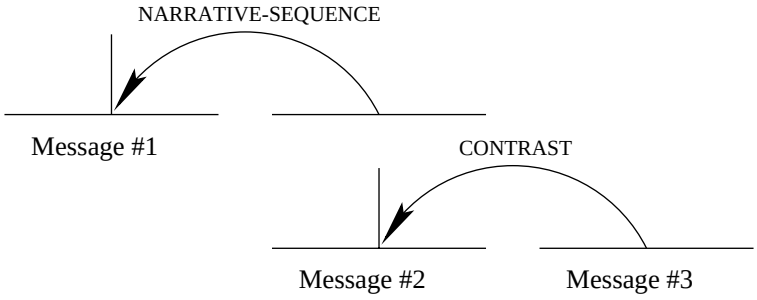

Figure 3: A document plan produced by WeatherReporter from the three input messages in Figure 2.

Thus, the microplanner knows that Message #1 and Message #2 are related by a discourse relationship of NARRATIVE-SEQUENCE, reflecting a domain convention that the two pieces of in-

formation in these messages are generally expressed in sequence; and that Message #3 provides information that is in CONTRAST to that provided in Message #2.[4]

We will assume that the microplanner has decided that this fragment of the document plan should be realised by means of a single sentence; in so doing we are glossing over many of the issues that arise in the determining the most appropriate aggregation of messages, but these questions are independent of our current concerns. Given this fragment of the document plan as input, the goal of the microplanner is to produce a sentence plan which, when provided to the surface realisation component, will result in the following sentence:

> June was warmer and wetter than average, although there was a dry spell from the 2^{nd} to the 5^{th}.

In the remainder of this paper, we look more closely at what is involved in achieving this result using the two surface realisation components introduced above.

3 Two Surface Realisers

3.1 KPML/NIGEL

A multilingual extension of the PENMAN system (Penman Natural Language Generation Group, 1989) developed by the Penman Project at ISI/USC,[5] KOMET-PENMAN MULTILINGUAL (Bateman, 1996) is based on SYSTEMIC FUNCTIONAL GRAMMAR (Halliday, 1985). There is insufficient space here to detail the internal workings of KPML, or to discuss the details of how the grammatical description is encoded: the reader is referred to the source literature for this information. In brief, the grammar is encoded as a SYSTEM NETWORK: an interconnected series of fine-grained choices specified in terms of communicative functions, whereby each choice made leads to other more specific choices, and each choice optionally adds a piece of surface grammatical description to constrain the final output sentence. KPML traverses such a grammar from left to right, considering questions and decision points of increasing detail. The end result is a set of REALISATION STATEMENTS—fine-grained constraints on the surface form—that characterise the particular sentence to be generated.

As Section 1 states, we choose to work with the NIGEL (Mann and Matthiessen, 1983) grammar for English, which is commonly used with KPML to generate English text. Input to KPML can be provided in the form of sentence plans in SPL (Kasper, 1989), the SENTENCE PLAN LANGUAGE (Kasper, 1989) developed for this purpose. Figure 4 shows an SPL expression that corresponds to our target sentence.

3.2 FUF/SURGE

The FUNCTIONAL UNIFICATION FORMALISM (FUF: see Elhadad's Ph.D. thesis (Elhadad, 1992) and the FUF User Manual (Elhadad, 1993)) has its origins in FUNCTIONAL UNIFICATION GRAMMAR (Kay, 1979), and uses graph unification to combine an input structure that corresponds to a sentence specification with a grammar of the output natural language, the result being a syntactically-specified structure which is then linearised to produce the required sentence. Both the input specification and the grammar itself are expressed as FUNCTIONAL DESCRIPTIONS (FDs), these being recursive attribute–value matrices whose expressive vocabulary permits the encoding of functionally-motivated elements; FUF shares with KPML the notion that in generation one is concerned with mapping from function to form.

[4] The form of analysis here is clearly influenced by work in Rhetorical Structure Theory (RST; (Mann and Thompson, 1988)). We have suggested here that Message #3 is presented in contrast to Message #2; however, an analysis that mirrors more closely the structure of the sentence to be generated would suggest that Message #3 is presented in contrast to Messages #1 and #2 combined. The differences between these two analyses are irrelevant to the present discussion.

[5] The Information Sciences Institute at the University of Southern California

```
(rst / rst-concessive
  :domain
    (l / greater-than-comparison
      :tense past
      :exceed-q (l a) exceed
      :domain (m / one-or-two-d-time :name June)
      :standard (a / quality :lex average :determiner zero)
      :range ((wa / sense-and-measure-quality :lex warm)
              (we / sense-and-measure-quality :lex wet)))
  :range
    (sp / existence
      :tense past
      :domain (s / abstraction
                 :lex spell
                 :determiner a
                 :property-ascription (d / quality :lex dry))
      :source (2nd / one-or-two-d-time
                 :lex 2nd
                 :determiner the
                 :destination (5th / one-or-two-d-time
                                :lex 5th
                                :determiner the))))
```

Figure 4: An SPL expression used as input to KPML/NIGEL.

As Section 1 states, we choose to work with the SURGE grammar for English, a grammar organised along systemic functional grammar lines, but which also borrows from HPSG (Pollard and Sag, 1994) and descriptive linguistic works (Quirk et al., 1985). Figure 5 shows an input FD that can be used by FUF/SURGE to produce our target output sentence.

4 Interfacing to the Realisers

4.1 General Issues in Microplanning

The role of the microplanner in our architecture is to provide an interface between the document planner's output and the realiser's input. In general terms this means the microplanner has to do three things:

- It performs aggregation, identifying situations where messages can be combined to produce more fluent text than would result if these messages were realised one-per-sentence.
- It performs lexicalisation, determining which lexical items should be used to realise concepts that appear in the messages.
- It performs referring expression generation, determining the appropriate noun phrase content required to identify entities in the domain.

All three functions must be carried out, irrespective of which of the two realisers is used. To make these activities clearer, we will indicate instances of each that occur in the generation of our target sentence.

Given the messages we have to work with as input, there are two instances of aggregation involved in producing the target sentence. First, the microplanner has to decide that all three messages will be realised within one sentence: in our discussion so far we have taken this for granted, but it is important to bear in mind that it does not just happen automatically. Second, the microplanner has to recognise that Messages #1 and #2 are sufficiently similar in structure that they can be expressed within one clause using a predicate that conjoins the two properties to be expressed. Aggregation operations of the latter kind are common in the

```
((cat clause)
 (tense past)
 (proc ((type ascriptive) (mode attributive)))
 (partic
   ((carrier ((cat proper) (lex "June")))
    (attribute
      ((cat ap)
       (complex conjunction)
       (common ((cat ap) (comparative yes)))
       (distinct ~(((lex "warm")) ((lex "wet"))))
       (qualifier ((cat pp) (prep === "than")
                   (np ((cat common)
                        (lex "average")
                        (definite yes)
                        (denotation no-determiner)))))))))
 (circum
   ((concession
       ((cat clause)
        (position end)
        (binder ((lex "although")))
        (tense past)
        (proc ((type existential)))
        (partic
          ((located
             ((cat common)
              (lex "spell")
              (describer ((cat ap) (lex "dry")))
              (definite no)
              (qualifier ((cat pp) (prep === "from")
                          (np ((cat common)
                               (lex "2nd")
                               (definite yes)
                               (qualifier
                                  ((cat pp) (prep === "to")
                                   (np ((cat common)
                                        (lex "5th")
                                        (definite yes)))))))))))))))))
```

Figure 5: A SURGE-compatible FD used as input to FUF.

literature; operations of the first kind, which we might think of as 'sentence scoping', are less widely discussed. The two operations require the use of knowledge about how to build good sentences: this is an area where much research remains to be done.

The microplanner also has to determine which lexical items will be used to realise the concepts in the messages: this means making use of knowledge that, in our present example, indicates that a situation with more rain is referred to as being *wetter* whereas a situation with a greater temperature is referred to as being *warmer*. This requires the microplanner to have access to appropriate domain knowledge.

Finally, the microplanner has to decide to refer to the month being described as *June* rather than as *last month*, *the sixth month of the year*, or some other equally true description. This requires the microplanner to make use of knowledge of the discourse context, so that, for example, it can choose to use a pronoun to refer to an entity when this is an appropriate thing to do.

4.2 An Aside on Grammatical Coverage

It is a reality of using packages like KPML/NIGEL and FUF/SURGE that, inevitably, one finds that there are gaps in the systems' grammatical coverage: there will always be grammatical structures that one wishes to build that are not catered for by the existing grammars. This is,

of course, no different to the situation that occurs with the use of existing grammars for parsing: no matter how broad coverage such a resource is, one always finds there are structures that it does not cater for. From this point of view, it is best to take the view that the SURGE and NIGEL grammars are still under development. We have found the developers of both systems extremely helpful in assistance with the making of grammatical extensions, but clearly this kind of support is not sustainable on a wider basis unless resources are specifically allocated to the task.[6]

Generation grammars also share with parsing grammars the property that, if one wishes to extend the grammar to cover the new phenomenon in question, this really needs to be done with a proper understanding of the theory underlying the model of grammar used; otherwise, the result will be at best something of an *ad hoc* solution, and at worst may result in unforeseen interactions with elements of the existing grammar. In order to avoid the latter problem, one needs a familiarity with the entire existing grammar: this is probably too much to ask of an end user of the system, and so until grammars such as NIGEL and SURGE reach a level of coverage where lacunae are relatively rare, this is a serious obstacle to genuine reuse of these resources.

In our experiments with these systems, we found grammatical limitations in both cases. By definition, the extent to which others find this problem will depend on how much their required grammatical coverage differs from that already provided in the systems, and again this is unpredictable without a reasonable familiarity with the systems' existing grammatical resources. Some of the apparent oddities in the sentence plans shown in Figures 4 and 5 are due to these limitations; most notably, it is not possible in NIGEL to express a time range. We are thus forced to use SPL's `:source` and `:destination` keywords to treat the start time and end time as a source and destination respectively, as in the final lines of Figure 4. The SURGE grammar has a similar limitation, in that temporal ranges are not yet expressible, and so we simply treat the start and end times of the range as prepositional phrases with the prepositions *from* and *to*, as in the last lines of Figure 5.

A simpler but related problem is that of limitations in lexical coverage. This is generally easier to fix than limitations in syntactic coverage; below we indicate how new lexical items can be introduced.

5 Comparing the Realisers

In this section we consider a number of specific phenomena that need to be handled in the microplanner's output in order to provide appropriate input for the two realisers, and comment upon the consequences this has for the microplanning process. These phenomena include the following:

- Rhetorical structures, e.g. contrast, concession, etc.
- Conjunctions of predicates and other constituents.
- Distinguishing proper names from common noun phrases.
- Creating new lexical items when required.
- Working around gaps in grammatical coverage.

5.1 Mapping Rhetorical Structure into Sentence Plans

The first two input messages of Figure 2 specify temperature and rainfall properties of the month in question, and the last message modifies the second by describing a dry spell over a certain time range. If we realise the information in the first two messages as the sentence

[6] Making serious use of either of these systems enforces an initially steep learning curve upon any user. Both surface realisation systems suffer from a lack of descriptive documentation on their respective grammars, the existing documentation for these grammars being mainly example-based. This tends to make the construction of inputs for the systems a predominantly trial-and-error based process.

June was warmer and wetter than average.

then the caveat that Message 3 represents is best expressed as a circumstantial adjunct to this sentence, with the surface form

however there was a dry spell from the 2^{nd} to the 5^{th}.

The two realisers use very different methods to express this type of adjunct. The KPML/NIGEL system relates the adjunct to the matrix clause via the rhetorical construct `rst-concessive`, where the top-level `:domain` keyword's value is the matrix clause and the top-level `:range` keyword's value is the circumstantial adjunct (as in the SPL expression in Figure 4). The FUF/SURGE system, on the other hand, represents such rhetorical constructs as circumstantial adjuncts to the matrix clause: a top-level attribute `circum` ('circumstantial'; in addition to the `proc` = 'process' and `partic` = 'participants' attributes) has as its value the attribute `concession` with the FD corresponding to the circumstantial adjunct as its value (see Figure 5).

5.2 Representing Conjoined Predicates in Sentence Plans

In combining Messages #1 and #2, the microplanner recognises that there is shared structure: both messages compare a property of the month's weather to the average for that month.

In the SPL expression of Figure 4, we use the top-level semantic type GREATER-THAN-COMPARISON (as both the temperature and rainfall are greater than their respective averages), and fold the warmness and wetness expressions into a conjunction which is the value of the `:range` keyword, as below:

```
:range ((wa / sense-and-measure-quality :lex warm)
        (we / sense-and-measure-quality :lex wet))
```

In the surge FD of Figure 5, the same information is expressed as a conjunction of adjective phrases, with the slight difference from the KPML/NIGEL approach being that features common to all conjuncts, such as `(cat ap)` and `(comparative yes)`, may be extracted and listed once only:

```
((cat ap)
 (complex conjunction)
 (common ((cat ap) (comparative yes)))
 (distinct ~(((lex "warm")) ((lex "wet")))))
```

5.3 Introducing Proper Names

We must specify in our inputs to both realisers that we require that the month be a proper name (which implies that no determiner precede it). This is done in an SPL expression (see Figure 4) with the `:name June` keyword-value pair, while in a SURGE FD (see Figure 5) we explicitly specify the type of the noun with the sub-FD `((cat proper) (lex "June"))`.

With these decisions made, the sub-part of the SPL expression corresponding to the topic becomes

```
:domain (m / one-or-two-d-time :name June)
```

and the corresponding sub-FD for SURGE becomes

```
(carrier ((cat proper) (lex "June")))
```

The distinction between these two approaches is that we must specify a *semantic* type for 'June' in the SPL expression, but a *syntactic* type in the SURGE FD.[7]

[7] Although we must also specify a syntactic type for 'June' in its lexical entry.

5.4 Adding New Lexical Items

The two realisers differ substantially in their approach to lexical items. The KPML/NIGEL system requires an entry in the lexicon for all words traditionally considered content words, and for some function words also. In addition, the lexical item is necessarily associated with a semantic type in the SPL expression. The FUF/SURGE system, however, requires that we explicitly include all lexical items in the SURGE FD for FUF to process, as this system does not have a lexicon in the same sense as KPML.[8]

An example declaration of a lexical item for KPML/NIGEL is the following, for the sentence topic:

```
(lexical-item  :name June  :spelling "June"
               :features (noun countable ...))
```

5.5 Working Around Incomplete Grammatical Coverage

As Section 4.2 discusses, the coverage of the NIGEL and SURGE grammars is less than perfect, the lack of expressibility of a time range exemplifying this.

An ideal surface realiser would facilitate grammatical coverage 'workarounds', by allowing us to specify features within the sentence plan independent of the grammar associated with the realiser. A few comments on how the KPML/NIGEL and FUF/SURGE systems deal with this problem follow, but specific examples of grammatical coverage limitations are left to Section 5.

There are two mechanisms available to both realisers to facilitate working around grammatical coverage limitations:

Generalised phrases: FUF/SURGE allows to specify phrases of a non-specific category and a `lex` feature containing the desired phrase, while KPML/NIGEL allows phrasal lexical entries, which however must be of a specific phrase type licensed by the NIGEL grammar.

Templates: If we combine the use of generalised phrases in FUF/SURGE with use of the `pattern` attribute, we effectively have a template mechanism. In KPML/NIGEL, templates are formalised in the SPL specification with a `:template` keyword.

The distinction between the FUF/SURGE approach and the KPML/NIGEL approach to partially template-driven generation is that the former inserts general strings between typed constituents, whereas the latter inserts typed constituents into positions in a general string.

6 Conclusion

Even a cursory inspection of the inputs required by the two systems makes it clear that they each assume the inputs provided to be at different levels of abstraction. Our analysis of the example presented in this paper, as well as others explored during this work, enables us to comment rather more specifically on the practical effects of this difference. We can categorise the effects in terms of what they mean for the microplanner in three domains: discourse structure, the lexicon, and grammatical structure.

Discourse Structure: In our architecture, the document planner is responsible for determining which discourse relations hold between the elements that make up the text to be generated. The microplanner is then concerned with how these structures correlate with decisions about paragraph and sentence content. As we saw in Section 5.1, the two systems require the microplanner to do quite different amounts of work. In the case of KPML/NIGEL, the microplanner simply decides that some discourse relation will be expressed within the bounds of a sentence, and leaves the realiser to determine how to

[8] The FUF system does however have some rudimentary morphological information, maintaining a list of morphological variations of noun plurals and irregular verbs.

realise that discourse relation in terms of syntactic structure and lexical choice. In the case of FUF/SURGE, the microplanner must do more work, and as we saw in the extended example, must specify the particular syntactic constructs to be used to carry the elements related within the discourse structure. FUF/SURGE thus requires the microplanner to have more knowledge of how to realise discourse relations than KPML/NIGEL appears to require.

The Lexicon: The different approaches the realisers take to lexical representation we discussed in Section 5.4, noting that the main distinction between the two systems is that KPML/NIGEL requires us to specify lexical items separately from the sentence plan. More importantly, to facilitate KPML's traversals through the NIGEL grammar, KPML requires us to attach a semantic type to the lexical item in the sentence plan. This is a consequence of the fact that, in this system, the lexicon is more closely related to the grammar than it is in FUF/SURGE.

This difference in approaches to specifying lexical information is symptomatic of a more general point: FUF/SURGE is a more self-contained system, while KPML/NIGEL allows the user to connect lexical concepts to external knowledge sources such as the UPPER MODEL (Bateman et al., 1990), a linguistically-motivated ontology, so that and a lexicon. The positive side of this requirement is that the information inherited from the UPPER MODEL and lexicon facilitates KPML's traversals through the NIGEL grammar, so that we need to specify less information in the input sentence plan.

Grammatical Structure: Section 5.5 discussed the methods each realiser uses to offset the problem of imperfect grammatical coverage, Section 4.2 exemplifying this by illustrating how we represented the concept of a time range with each realiser. Representing a start and end time as a source and destination, respectively, is semantically inappropriate; representing a start and end time as prepositional phrases, while making no concessions to the semantic nature of the entities, is at least syntactically appropriate. This may be preferable to using semantically inappropriate terms (or needing to update the grammar before being able to generate surface text expressing time ranges).

Generalising across these three areas, it is clear that the more abstract semantically-oriented input KPML/NIGEL requires implies that the microplanner has to do less work in building sentence plans, whereas the more syntactically-oriented input required by FUF/SURGE implies that the microplanner must know something about syntactic possibilities and must be able to map elements of the document plan into these. This may seem like a distinct disadvantage to using FUF/SURGE; on the other hand, using a higher level of abstraction for input means that one has to be familiar with the vocabulary of that representation. This was perhaps most obvious in our discussion of realising the temporal range construct in our target sentence: there, a limitation in grammatical coverage meant that we had to subvert the proper use of semantic constructs in KPML/NIGEL in order to achieve the results we required. Doing this, of course, requires knowing what the syntactic effects of using those semantic constructs will be, whereas in FUF/SURGE we can adopt a workaround which is more directly grammatical in nature.

In summary, both systems are excellent resources for anyone who needs to incorporate a surface realiser into a natural language generation system. Both, however, suffer from limitations in coverage which require the user to know something of their internal behaviour so as to be able to develop workarounds. In the case of KPML/NIGEL, the more abstract nature of the input representation appears to require a greater commitment to the underlying theory. Whether this is acceptable depends largely on the nature of the host system: in many simpler generation systems, there is already a tendency at the document planning and microplanning stages to talk in terms of syntactically-motivated informational elements, so the abstractions made available by KPML/SURGE may be unnecessary. In systems which reason in more conceptually-oriented structures, however, the advantages that come from being able to utilise the Upper Model mean that the microplanner needs to do less work than is needed to produce inputs appropriate for FUF/SURGE.

References

Bateman, J. A. (1996). KPML Development Environment. Technical report, IPSI, GMD, Darmstadt, Germany. Documentation on this system is at `http://www.stir.ac.uk/english/communication/Computational-tools/kpml.html`.

Bateman, J. A., Kasper, R. T., Moore, J. D., and Whitney, R. A. (1990). A general organization of knowledge for natural language processing: the PENMAN upper model. Technical report, USC/Information Sciences Institute, Marina del Rey, California.

Bourbeau, L., Carcagno, D., Goldberg, E., Kittredge, T., and Polguere, A. (1990). Bilingual generation of weather forecasts in an operations environment. In *13th International Conference on Computational Linguistics, COLING-90*.

Elhadad, M. (1992). *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. Ph.D. Dissertation, Graduate School of Arts and Sciences, Columbia University.

Elhadad, M. (1993). *FUF: The Universal Unifier. The User Manual Version 5.2*. Department of Computer Science, Ben Gurion University of the Negev, 84105 Beer Sheva, Israel.

Elhadad, M. and Robin, J. (1996). An overview of SURGE: a reusable comprehensive syntactic realization component. In *Demonstrations and Posters of the 8th International Workshop on Natural Language Generation (INLG-96)*, pages 1–4, Herstmonceux, England.

Halliday, M. A. K. (1985). *An Introduction to Functional Grammar*. Edward Arnold, London.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B., and Polyguere, A. (1992). Generation of extended bilingual statistical reports. In *14th International Conference on Computational Linguistics, COLING-92*.

Kasper, R. T. (1989). A flexible interface for linking applications to Penman's sentence generator. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, USC/Information Sciences Institute, Marina del Rey, CA, USA.

Kay, M. (1979). Functional grammar. In *Proceedings of the 5th meeting of the Berkeley Linguistics Society*, pages 142–158. Berkeley Linguistics Society.

Kukich, K. (1983). *Knowledge-based report generation: A knowledge-engineering approach to natural language report generation*. Ph.D. Dissertation, University of Pittsburgh.

Kukich, K., McKeown, K., Morgan, N., Phillips, J., Shaw, J., and Lim, J. (1993). User needs and analysis and design methodology for an automated documentation generator. In *the Bellcore/BCC Symposium on User-Centered Design: 'People and Technology'*, Piscataway, N.J.

Mann, W. C. and Matthiessen, C. M. (1983). Nigel: A systemic grammar for text generation. Technical Report ISI/RR-83-105, Information Sciences Institute. 4676 Admiralty Way, Marina del Rey, California 90292-6695.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Maybury, M. T. (1995). Generating summaries from event data. *Information Processing and Management*, 31(5).

Penman Natural Language Generation Group (1989). *Penman Project. PENMAN documentation: The Primer, the User Guide, the Reference Manual and the Nigel Manual*. USC/Information Sciences Institute, Marina del Rey, CA, USA.

Pollard, C. and Sag, I. A. (1994). *Head-driven Phrase Structure Grammar*. Studies in Contemporary Linguistics (Ed. John Goldsmith, James D. McCawley and Jerrold M. Sadock). The University of Chicago Press, Chicago and London.

Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A comprehensive grammar of the English language*. Longman.

Robin, J. (1994). *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. Dissertation, Columbia University.

Rosner, D. (1987). The Automated News Agency: Semtex - A text generator for German. In *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics (G. Kempen Ed.)*, pages 133–148. Martinus Nijhoff Publishers.