Phrasing a text in terms the user can understand*

John A. Bateman and Cecile L. Paris
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Abstract

When humans use language, they show an essential, inbuilt responsiveness to their hearers/readers. When language is generated by machine, it is similarly necessary to ensure that that language is appropriate for its intended audience. Much of previous research on text generation and user modelling has focused on building a user model and selecting appropriate information from the knowledge base to present to the user. It is important, however, that the *phrasing* of a text be also tailored to the hearer - otherwise it may be just as ineffective as texts which wrongly direct attention or which rely on knowledge that the hearer does not have. This research proposes a new mechanism which allows the text planning process to specifically tailor syntactic phrasing to the hearer type. This is done in the context of an expert system explanation facility that needs to produce explanations of the expert system's behavior for a variety of different users - users who differ in goals, expectations, and expertise concerning both the expert system and its domain.

1 Tailoring - the importance of making language appropriate for its audience

Humans show an essential, inbuilt responsiveness to their hearers/readers in their use of language. It is similarly necessary to ensure that the language generated by machine is appropriate for its intended audience. Much text generation research in the past has focused on the selection of text *content* and *organization* in order to accomplish *speakers'* goals (e.g., [McKeown, 1985, Ilovy, 1988]). The presentation of that content is generally also made responsive to *hearers'* states of focus of attention (e.g., [Grosz and Sidner, 1986]). More recently, it has been recognized that it is necessary in addition to make generated text sensitive to the hearers' goals and knowledge about domains, that is to take a user model into consideration (e.g., [McKeown *et a*/., 1985, Appelt, 1985, Jameson, 1987, Ilovy, 1988, Paris, 1988, Carberry, 1988]). These are all important factors if the text generated is to be both informative and understandable to the user.

The language used by specific groups of people, however, often possesses syntactic patterns and lexical features that are distinctive to those groups; question answering systems can only be effective, therefore, if they appropriately customize their *phrasing* as well as their content and textual organization according to each distinctive group of users - otherwise generated texts may be just as ineffective as texts which wrongly direct attention or which rely on knowledge that the hearer does not have. In contrast to most previous research, which has focused on the selection and organization of information from the knowledge base for presentation to the user, this research addresses the issue of expressing the selected information in language specifically tailored to the hearer.[1] This is done in the context of an expert system explanation facility.

2 What is involved in 'tailoring'

In tailoring a response, whether it be to a user's goals for asking a question or to that user's level of expertise in a domain, a generation system first has to choose which information from the knowledge base at hand is most appropriate and organize its overall text structure. The result of this phase is an organized collection of the particular propositions to be expressed in English. Most generation systems take these propositions as the inputs for the realization components of grammatical and lexical selection (e.g., [McKeown, 1985, Moore and Swartout, 1989]). However, the output of this phase is typically not detailed enough to control the many possibilities for expression that current grammar components provide. There is a large gap between the

*The research described in this paper was supported in part by the Defense Advanced Research Projects Agency (DARPA) under a NASA Ames cooperative agreement number NCC 2-520, by AFOSR contract F49620-87-C-0005, and by DARPA contract MDA903-87-C-641.

[1]This issue was partially addressed in the HAM-ANS system [Morik, 1985], where, based on a user model, the system would decide whether to produce an anaphora. The work presented here is different as we are more concerned about *systematic* linguistic differences that exist in the language used by various groups of users.

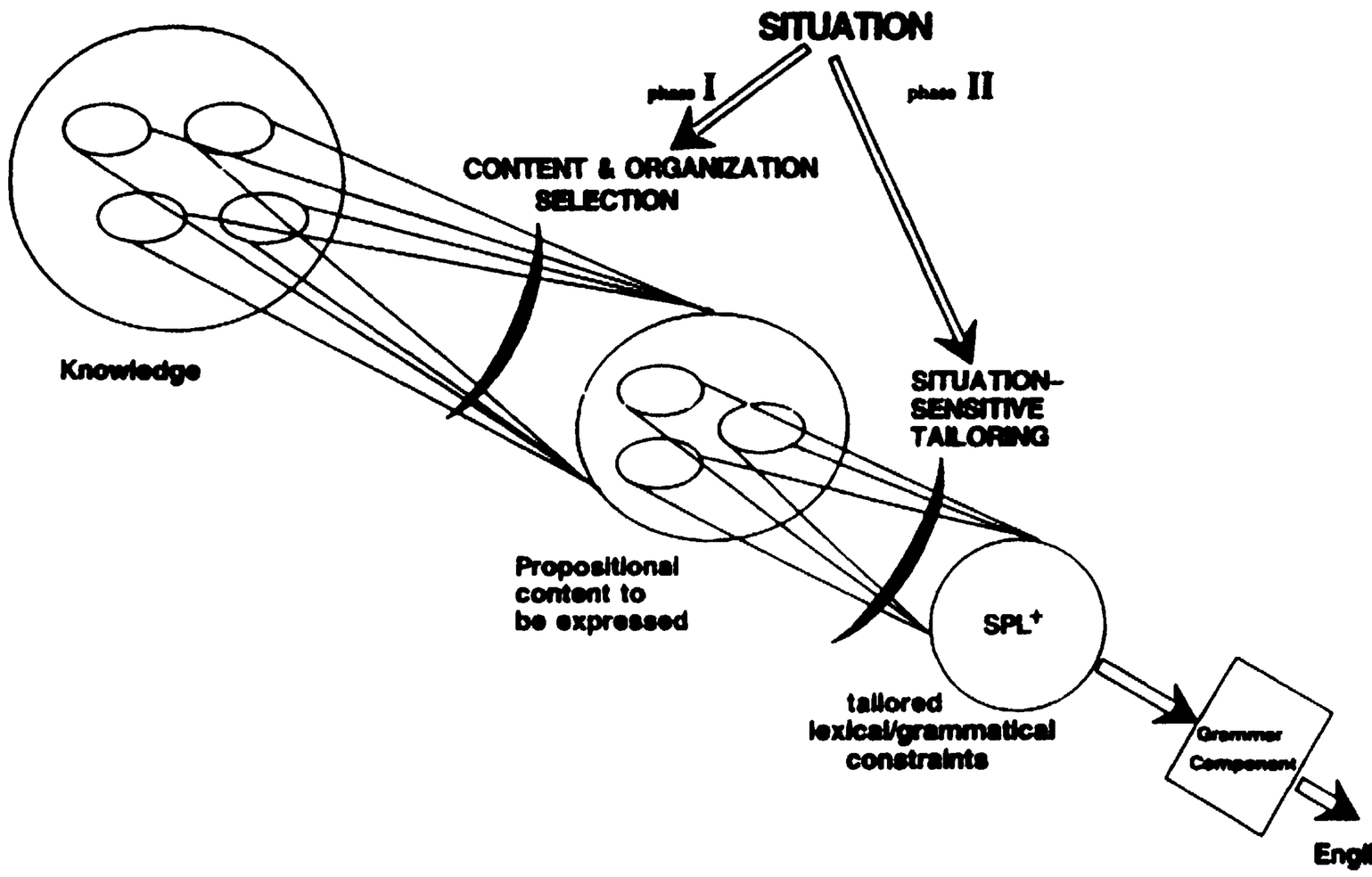

Figure 1: Schematic generation model involving tailoring of both content and phrasing

level of detail of the output of the text planner and the input required by a grammar.

This is problematic for natural text generation for two reasons. First, generated texts could unintentionally be inappropriate in many contexts; second, a text planner's lack of control over the grammar could prevent it from producing the text it intends to generate. Because different classes of users have different ways of speaking, involving systematic differences in the phrasal and grammatical patterns that they employ, a generator needs to have theoretical control of phrasal organization just as it does of text organization. The generation of natural text requires a second phase of planning that bridges this gap. This paper presents a mechanism that achieves this.

Control of the phrasing task has mostly remained below the level of detail that the text planning process has under theoretical control. Phrasing involves choosing appropriate syntactic and lexical structures from all the available possibilities that express the same propositional content. The few attempts to gain control of phrasing in accordance with general text planning goals (e.g., [Appelt, 1985, Morik, 1985, Hovy, 1988]) have either been restricted in the areas of phrasing they have examined, or have not maintained sufficient separation between text planning and realization.[2] In our work, we are constructing a general phrasing control component that interfaces between the output of the first phase of planning and the input to the grammar. (This is illustrated in Figure 1.) This component decides both *which aspects* of the output of phase one are most appropriate for each user type and *how they are to be phrased* for that user. It is only the result of this second phase of selection/organization that provides sufficient guidance to the grammar and lexical selection components.

We use the notion of *register* [Halliday, 1978, Patten, 1988], which specifies the linguistic consequences of using language in particular situations. Registers may be seen as describing the 'argots' used by different classes of users. Our approach has been to restrict our attention to the particular kinds of language required in our domain. We have worked *back* from these to construct sets of 'terms' to be employed when generating language for particular users. This is equivalent to a linguistic description of specifically the situation-specific aspects of some instances of language use. We can then define a phrasing algorithm which uses these sets to effectively constrain the second phase of the generation process.

[2]It is generally accepted that a text planner should not maintain detailed knowledge of the grammatical possibilities offered by the grammar; to do so complicates the planning process considerably by requiring the text planner to concern itself with details from an inappropriately low level of abstraction. Similarly, the grammar should not include detail at an inappropriately 'high' level of abstraction.

3 The expert system explanation task

3.1 The expert system

The system for which we are generating language is an expert system constructed using the Explainable Expert System (EES) framework [Swartout and Smoliar, 1987]. In this framework, an expert system includes support knowledge for explanation. The support knowledge contains, for example, terminology definitions, so that a user can ask for the definition of specific terms used by the system and conditions of evaluation for states of affairs within the domain (i.e., whether a particular state is good or bad).

We are using a particular instantiation of this framework, an expert system designed to diagnose digital circuits. This system includes in its domain knowledge a variety of facts concerning objects in the domain of dig-

```
(define-type-attributes faulty-system
 :defining-conditions: ((and (E (o in (output-part self))
                                (not (equal (expected-value (signal-part o))
                                            (actual-value (signal-part o)))))
                             (A (i in (input-part self))
                                (equal (expected-value (signal-part i))
                                       (actual-value (signal-part i))))))))
```

Figure 2: Defining conditions of a `faulty-system`

```
The system is faulty, if there exists a O in the set of the output terminals of the system such that
the expected value of the signal part of O does not equal the actual value of the  signal part of O
and for all I in the set of the input terminals of the system, the expected  value of the signal part
of I equals the actual value of the signal part of I.
```

Figure 3: Text generated from input shown previously, tailored to interaction group 1

```
The system is faulty, if all of  the expected values of  its input terminals equal their actual values
and the expected value of one of its output terminals does not equal its actual value.
```

Figure 4: Text generated from input shown previously, tailored to interaction group 2

ital circuits, how they may be related, and what can go wrong with them. While explaining its reasoning, the system sometimes needs to provide a definition of some domain object, such as a faulty-system. To do so, the explanation routine first examines all the information contained in the knowledge base about faulty systems to determine what to include in the text; this is phase one of the generation process (cf. Figure 1). Suppose the generation routine decides to select only the conditions under which a system is faulty (called *defining conditions).* This information (shown in Figure 2) is contained in one slot of the object frame. It is represented as predicate calculus and constitutes the basic propositional content that is to be expressed in the response. This information is *highly structured*. The structure is defined by the EES grammar which specifies the permissible predicate calculus formulae. Such definitions are parsed according to this grammar so that specific tools, including a syntactic matcher, can be used to traverse the parse tree and provide access to its constituents.

Given this propositional content, however, it is still possible to phrase this definition in several ways, each being appropriate for some different type of user.

3.2 The types of language and users required

This expert system is expected to support interactions with at least the following groups of users:

Group 1 : System developers who want to make sure that the knowledge base is correctly represented and that the system is working properly.

Group 2 : End-users who want to follow the system's reasoning, but who do not know much about expert system technology (or even about computer science).

Each of these groups demands rather different language to be employed in their interactions with the system. We call these groups *interaction groups.*[3]

[3]These two user types are being considered for the initial study. As the system is used further, it is likely that we will have to handle more interaction groups. The question of extensibility therefore becomes quite important.

For example, in response to the user question 'What is a faulty system?', while the first phase of the generation process gives rise to the internal predicate calculus form shown in Figure 2, a variety of significantly different phrasings are required. For users in group 1, a definition in an English form as close as possible to the exact definition expressed in predicate calculus is needed, as these users employ the explanation facility for debugging purposes. There is therefore a need to be very precise and literal. The text generated by our current algorithm for group 1 is shown in Figure 3. For users in group 2, however, this text is likely to be confusing at best. Figure 4 shows the more appropriate text that is generated by our algorithm when tailoring phrasing for users in group 2. The next section describes how our generation component is able to generate these two very different texts from the same internal content.

4 Description of the algorithm

4.1 The grammatical resources available and methods for controlling them

The grammar used in this work is Nigel [Mann and Matthiessen, 1983], a systemic-functional grammar [Halliday, 1985] of English. Control of the selection of grammatical features in Nigel is achieved by employing a set of semantic *inquiries* that access grammar external sources of information, such as the knowledge base and the input to the grammar.

An interface language (the Sentence Plan Language or SPL) that provides for very flexible control over the grammar has recently been developed for Nigel [Kasper, 1989]. This language provides for the convenient control of all aspects of Nigel at a variety of levels of abstraction, including several that provide for direct responses to Nigel's inquiries, SPL can be seen as providing a set of constraints that must be met by the language generated.[4] Given a specification of propositional con-

[4]This interpretation is compatible with general functional unification-based schemes of control and so is not particularly bound to the Nigel grammar as currently implemented.

```
(g1 / give  :actor (p1 / Person :name Cecile)
            :actee (b1 / Book)
            :beneficiary (p2 / Person :name John)
            :speechact assertion)
```

Figure 5: Simple example of SPL: input specification for: *Cécile gives a book to John*

```
(e1 / existence
      :domain (b1 / Book
                :relations (g1 / give  :actor (p1 / Person :name Cecile)
                                       :actee b1
                                       :beneficiary (p2 / Person :name John)))
      :speechact assertion)
```

Figure 6: SPL example: input specification for: *there is a book that Cécile gives to John*

```
(define-register-terms *register-for-system-developers*
((*mode*  (RHETORICAL-RELATIONS ifthen iff and or))
 (*field* (RELATIONS exist forall not equal object-relation-ascription material-property-ascription
                     signal-part connected-to)
          (OBJECTS expected-value actual-value value signal output-terminal input-terminal system variables)
          (OBJECT-MODIFIERS input-part output-part part-of))
  (*tenor* nil)))
```

```
(define-register-terms *register-for-end-users*
((*mode*  (RHETORICAL-RELATIONS ifthen iff and or))
 (*field* (RELATIONS not equal connected-to material-property-ascription)
          (OBJECTS expected-value actual-value value signal output-terminal input-terminal system)
          (OBJECT-MODIFIERS exist forall input-part output-part part-of))
 (*tenor* nil)))
```

Figure 7: Head status definitions used for the two registers shown above

tent (such as that shown in Figure 2), our phrasing component constructs SPL expressions that are then passed to the grammar.

SPL input specifications consist of a recursive structure of entities and their features to be expressed in English. Each entity needs to be allocated a *type* which is interpreted with respect to a knowledge base of general conceptual categories called the *upper model.* The upper model is typically used to mediate between the organization of knowledge found in an application domain and the kind of organization that is most convenient for implementing the grammar's inquiries. Upper model concepts possess specified roles that define the possible semantic relations that may appear in the SPL. An example of the SPL representation of a sentence is given in Figure 5. This specification states that there is a relation of type *give* to be expressed as an *assertion,* holding over three participants, an *actor,* an *actee,* and a *beneficiary.* These participant roles are drawn from those defined in the upper model for relations of type *give.* The fillers of these roles are themselves SPL expressions.

An important source of variation in phrasing occurs in the construction of the SPL specification. For example, in Figure 6, the *existence* of the book has been made salient. The SPL term for that book now occurs at a higher level within the structure than the process of *giving* it partipates in; we say that it is of a *higher rank.* The central typed terms at each level of the SPL structure (e.g., *g1, p1, b1*, and *p2* in Figure 5) correspond to the *heads* of the linguistic expressions that realize them.[5] Any particular SPL expression commits the language that will be generated to expressing a single head-modifier organization.

The allocation of heads and their respective rankings have a significant effect on phrasing. Our mechanism for controlling this allocation relies on the notion of register. A register defines the entity and relation types that are allowed to become heads in the language of that register; this is described in detail in Section 4.2.

Once the SPL expression is constructed, however, a number of distinct sentences satisfy its constraints, i.e., the expression is underspecified. Consider for example the SPL expression shown in Figure 5. At least three distinct sentences satisfy its constraints: *Cecile gives John the book] Cecile is giving a book to John; to John, Cecile gave a book.* If the text planner is to control these alternative phrasings, then additional guidance is required; our mechanism for this is also defined using registers and is described in Section 4.3.

4.2 Constraint on head selection

Definitions[6] of the available heads and their relative rankings for the two registers used to generate the examples tailored to interaction groups 1 and 2 are shown in

The head of a linguistic expression is the 'independent variable' of that expression; it is often considered to be more salient than the other constituents of the expression. The head places constraints on those constituents (the head's modifiers) and thus constrains the expression's overall syntactic structure. The head-modifier organization of a linguistic expression is also recursive.

The definitions divide heads among three types, the *field, mode,* and *tenor.* This imposes a further dimension of ranking in the search process drawn from the linguistic theory of register. We will not discuss these details here however.

```
register term and:      "(and !assertion#1 !assertion#2)"
input constituents:     and, assertion1, assertion2

register term exist:    "(E (!variable-symbol in !set-valued-expression) !assertion )"
input constituents:     exist, variable-symbol, set-valued-expression, assertion
```

Figure 8: Patterns linking register terms *and* and *exist* to the input specification language

```
(a1 / conjunction    :conjunct1  <spl constructed for assertion1>
                     :conjunct2  <spl constructed for assertion2>)
```

Figure 9: Constructed SPL fragment

Figure 7. The terms used in these definitions are linked into Nigel's upper model; this guarantees their expressibility. They are also linked to the input specification language so that instances of their occurrence can be recognized.

The first definition establishes the relative importance of entities and relations according to the language used by system developers, the second according to the language of other end-users. The definitions state that the terms labeled RHETORICAL-RELATIONS are to be preferred for higher ranking than those labeled RELATIONS, which are in turn to be preferred to those labeled OBJECTS, etc. Importantly, the groupings are labeled in terms of their *linguistic* realization, and *not* in terms of their status in the knowledge base. For example the predicate calculus term *exist* gets realized in English as a *relation* (process) in the register for system developers (giving rise to 'there exists a <entity> ...') while it gets expressed as an *object modifier* in the register for end users (giving rise to '... each <entity>'). Thus, whereas in the register for system developers the relations *exist* and *forall* are given high ranking, they are given the lowest ranking in the register for end-users. Moreover, whereas the term *variables* is available in the register for system developers, it does not occur in the register for end-users.

The same propositional content is phrased in various ways, depending on the entities and relations available in the register and their relative rankings. By choosing a register the text planner can systematically control the phrasing of a given propositional content. Phrasing is decided upon by successively finding the highest ranking head offered by the register at each level of structural decomposition of the input, as is illustrated below.

Given the input specification of Figure 2, for example, the program searches the list of register terms for the highest ranking available head whose pattern matches the structure of the input. In the register for system developers, one of the highest allowable heads is the rhetorical relation *and.* The recognition of terms in the input specification language is defined using structural patterns that may be matched against the input. Patterns are expressed in terms of the *input* syntactic categories (specified by the EES grammar) that define the constituents of the input at that level; the pattern for the rhetorical relation *and* is shown in Figure 8. Continuing with our example, this pattern matches the top level of structure shown in Figure 2 and so offers an appropriate head for describing the input.

Accordingly, an SPL term is constructed that:

1. specifies the register term found, namely *and,* as an appropriately ranking head,
2. places the remaining input constituents in appropriate modifier relationships. These relationships are drawn from the upper model definition of the type of the head which, in this case, are *conjunctl* and *conjunct 2.*

This gives rise to the basic SPL fragment shown in Figure 9.

This process is then applied recursively for each constituent, namely *assertionl* and *assertion2,* searching for permissible lower ranking heads. Both assertions in the definition of a faulty system are quantifier expressions. (The pattern for *exist* is shown in Figure 8.) These expressions have a structural decomposition in terms of the quantifier type, the variable to which the quantifier applies, the variable range and the assertion about that variable. Therefore, for both *assertionl* and *asseriion2*, patterns for predicate calculus quantifiers *(there-exists* for *assertionl* and *for-all* for *assertion2)* match. The SPL is grown accordingly, and the process recurses.

If no applicable head from the active register matches the input constituent at that level, the intrinsic structuring of the input is used to select where to search next for an applicable head. Elements of the input that are not allocated to heads at that level are retained for possible consumption later in the process. Examples of this can be seen in the differential treatment of variables and relation-types in the two registers. Whereas variables may become heads in the system developer register and are expressed as noun phrases, there are no heads corresponding to variables in the end-user register and so they do not appear in the English of that register. Similarly, some relation types, such as signal-part, do not appear in the end-user register and thus are not consumable in the construction of SPL for the generation of the English in that register.

4.3 Constraint on grammatical feature selection

As we saw in Figure 5, the straightforward propositional content specified in SPL head and modifier terms does not restrict the possible result to a single sentence. A variety of aspects of the phrasing will not be controlled and need to be constrained. We also provide this type of constraint in a register-driven way by associating particular

```
(defspl-default high-precision
      :posture-of-existence-q posture                  :selection-particularity-q particular
      :register-q notdistinguished                     :deictic-part-q notpart
      :possessor-modification-q nopossessor            :empty-number-q nonempty
      :empty-gender-multiplicity-q nonempty            :extension-precedence-q precedes)
```

Figure 10: Definition of grammar constraints

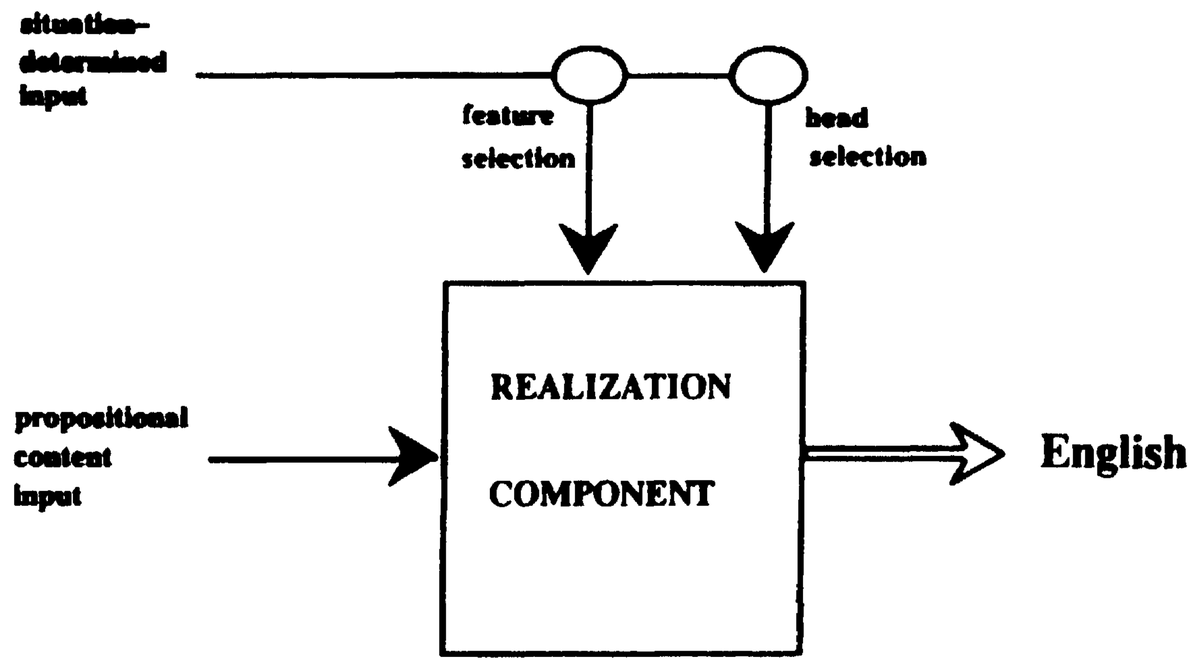

Figure 11: Inputs to the generation process

sets of grammatical features with particular selections of register. This is implemented in terms of SPL 'default environments' within which the theoretically available grammatical alternatives are restricted to some specified subset. An example of a default environment activated when the system developer register is in force is shown in Figure 10.

Default environments consist of a set of Nigel inquiries (cf. Section 4.1) and the responses those inquiries are to be given if they are needed during generation. By predisposing inquiry responses, areas of the grammar as a whole can effectively be removed from consideration so that the alternatives they offer no longer appear relevant. This is equivalent to the dynamic construction of a 'sub-grammar'.

The sub-grammar created by Figure 10 has the following properties: processes of existence will be lexicalized by the word *exist* rather than some more neutral word, such as be, as in *there is a book* (:posture-of-existence-q); when possible, selection of a member of a set will be made using specific expressions, e.g., *one X* rather than *some X* (:selection-particularity-q); since variables are generally refered to in the singular, the distinction between *all* and *every* is neutralized for variables (:register-q); 'de-emphasizing' expressions such as the possessive modification of *'X's Y'* are avoided in favor of the more equal emphasis of *' Y of X'* (:deictic-part-q and :possessor-modification-q); pronominalization is not attempted (: empty-number-q and empty-gender-multiplicity-q); and the order of conjuncts and disjuncts in the input will not be altered in the output by thematic planning of any kind (:extension-precedence-q). This sub-grammar is therefore oriented to language that is precise and explicit, and which is natural for system developers. This aspect of our approach is similar to the method employed by Patten [1988].

4.4 Summary of the constraints

Figure 11 summarizes the inputs to the generation process and the location of situation-specific constraints in that process. In addition to the basic propositional content input specification, we have experimented with two further sources of control. *Feature* constraints that hold sway whenever a text of a particular type is being generated, and *head* selection, which is specific for each specification of propositional content provided as input.

5 Generalizeability of this approach

Given the need to be able to expand the use of an expert system to users of different types, it is important that the tailoring techniques employed are readily extensible. We have shown here how the phrasing phase of tailoring can be controlled in terms of a set of entities that the type of language required uses and a set of restrictions on the general capabilities of the grammar. Both aspects would need to be addressed in any case when setting up an expert system to interact with a new user group since, otherwise, the language best suited to that group would not have been determined.

We illustrate the ease of extensibility of the technique shown here by setting up a new user type and showing the language that is generated for that user type given the input of Figure 2. Imagine a class of users who are not expert in the domain of digital circuit analysis and who just need to be presented with responses in as simple terms as possible. First, we do not need to invoke the grammatical feature restrictions that lead to pre-

```
(define-register-terms *register-for-naive-users*
  ((*mode*  (RHETORICAL-RELATIONS  ifthen iff and or))
   (*field* (RELATIONS  material-property-ascription connected-to )
            (OBJECTS  output faulty input system)
            (OBJECT-MODIFIERS input-part output-part part-of))
   (*tenor* (EVALUATION fine wrong))))
```

```
The system is faulty, if the inputs are fine and the output is wrong.
```

Figure 12: Head status definitions and generated English for the naive user register

ciseness shown in Figure 10. Second, we provide a set of significant entities, shown in Figure 12, in which quantifiers and variables do not appear, the space of available objects that may be referred to is reduced, and 'evaluations' of domain states of affairs are permitted. The text generated when our algorithm runs with this register in force is also shown in the figure.

6 Conclusion and future work

In this paper we have described a mechanism which controls the fine detail phrasing of generated language according to a definition of the type of language that is required. In the short term, this work has already improved the control of phrasing in a text planner; the mechanisms we have designed allow for a far more flexible, and yet systematic, expression in English of *single specifications* of propositional content than has previously been possible. In the longer term, we are aiming towards a single, unified component module of the text planning/generation process that adds flexible tailoring to the user during both the content selection and phrasing phases of text planning.

Acknowledgments

The authors would like to thank Eduard Hovy, Johanna Moore and William Swartout for comments on earlier versions of this paper.

References

[Appelt, 1985] Douglas E. Appelt. *Planning Natural Language Utterances.* Cambridge University Press, Cambridge, England, 1985.

[Carberry, 1988] S andra Carberry. Plan recognition and user modelling. *Computational Linguistics,* 14 (3), September 1988.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions and the structure of discourse. *Computational Linguistics Journal,* 12 (3), 1986.

[Halliday, 1978] Michael A. K. Halliday. *Language as social semiotic.* Edward Arnold, London, 1978.

[Halliday, 1985] Michael A. K. Halliday. *An introduction to functional grammar.* Edward Arnold, London, England, 1985.

[Hovy, 1988] Eduard H. Hovy. *Generating natural language under pragmatic constraints.* Lawrence Erlbaum, Hillsdale, New Jersey, 1988.

[Jameson, 1987] Anthony Jameson. How to appear to be conforming to the ' maxims' even if you prefer to violate them. In G. Kempen, editor, *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics.* Kluwer Academic Publishers, Boston/Dordrecht, 1987.

[Kasper, 1989] Robert Kasper. SPL: A sentence plan language for text generation. Technical report, USC/ISI, 1989.

[Mann and Matthiessen, 1983] William C. Mann and Christian Matthiessen. Nigel: A systemic grammar for text generation. Technical Report ISI/RR-85-105, USC/ISI, February 1983.

[McKeown *et ai,* 1985] Kathleen R. McKeown, Michael Wish, and Kevin Matthews. Tailoring explanations for the user. In *Proceedings of IJCAI-85,* Los Angeles, Ca., 1985. International Joint Conference on Artificial Intelligence.

[McKeown, 1985] Kathleen R McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, England, 1985.

[Moore and Swartout, 1989] Johanna D. Moore and William R. Swartout. A reactive approach to explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, MI, August 20-25 1989. IJCAI.

[Morik, 1985] Katharina Morik. User modelling, dialog structure, and dialog strategy in HAM-ANS. In *Proceedings of EACL-85,* Geneva, Switzerland, 1985. European Association of Computational Linguistics.

[Paris, 1988] Cecile L. Paris. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics,* 14 (3), September 1988.

[Patten, 1988] Terry Patten. Compiling the interface between text planning and realization, August 1988. Workshop on text planning and Natural Language Generation, Sponsored by AAAI.

[Swartout and Smoliar, 1987] William R. Swartout and Steve W. Smoliar. Explaining the link between causal reasoning and expert behavior. In *Proceedings of the Symposium on Computer Applications in Medical Care,* November 1987.