Surface generation for a variety of applications

by DAVID D. McDONALD
University of Massachusetts
Amherst, Massachusetts

ABSTRACT

If a knowledge-based computer program is to produce English texts that are fluent and appropriate to the context of their audience and situation, the program’s model of the conceptual information to be communicated must be augmented by a model of the English language, including linguistic information of the kind one finds in grammars and books of rhetoric, such as its constraints, alternative formulations, styles, carrying capacity, etc. Since this information is for the most part uniform across subject matters and tasks, it is included as part of any task-specific generation system, thereby relieving other designers of the need to be concerned with its unavoidable complexities. This paper describes the design issues that arise in developing such a linguistic component, using illustrations from a system that has been successfully interfaced to eight text-planning programs, and a variety of knowledge representation formalisms.
INTRODUCTION

When we want to have a conventional program, such as a compiler or an operating system, give a message to its human user, our job is straightforward; we decide where the message should go in the program's code and insert the text that we want to use, perhaps including some formatting statements or some fill-in-the-blank variables that will be substituted for at runtime with a simple name, number, label name, etc. We can get away with this simple a procedure because we can anticipate the exact situations in which our conventional program is going to have to give its messages, and so can decide once and for all on how its texts should read. We can be as polished as we like, and have no difficulty ensuring that the texts contain the right information or are grammatical because we are writing them ourselves.

Messages from knowledge-based systems, especially those that are intended to act as assistants or expert advisors, cannot be handled so easily. The simple regimen of pre-selecting what to say and including it directly in the program breaks down in the face of the impossibly large number of subtly varying situations that such a system will have to work in.

A well designed AI system will be sensitive to differences in the goals and experience of the people using it, to variations in the relative importance of the parts of what it has to say and the perspective from which they should be described, and to the content and rhetorical structure of the discourse that has come before. These factors are crucial components of what are coming to be called intelligent interfaces, and are beginning to be part of the criteria by which we evaluate any knowledge-based system. If its communications are not clear, fluent, and appropriate to the situations and audience, the system will not be accepted by its users.

Achieving good communications involves many factors, such as an expressive underlying representation, a good model of the user, a well-designed graphic interface, and careful planning including deliberations over alternative presentations and their consequences. This paper considers an additional factor, that of understanding linguistic facts and the process of producing utterances in a natural language.

Linguistic facts are elaborate, complexly interrelated, and not at all related to the nature of the information being communicated. Thus, the task of designing an efficient linguistics component, the part of the system that is responsible for the realization of the system's goals and internally represented information as a cohesive grammatical text, is exceptionally difficult. To make progress in a way that can be generalized, one generally has to bring together the specialized knowledge of linguists, AI specialists in representation, and sophisticated symbolic programmers. One should also not be in a hurry.

Only a few research groups have ever developed programs that could generate high quality texts by general methods, and the task is generally acknowledged to be more difficult than its obverse, natural language comprehension.

The difficulty the abovementioned problems pose is mitigated by the fact that if one is careful, the task need only be done once. Linguistic facts and the heuristics for using them are uniform across subject matters and tasks, thereby making it possible to capture this information in a common program module that can then be included as a part of any task-specific communications system, thus relieving other designers of the need to be concerned with its complexities. I have developed such a "linguistics component," and with my colleagues at the University of Massachusetts, continue to extend its abilities and apply it to tasks such as scene description, tutoring, and summarizing narratives.

In this paper we will look at the place of a linguistics component (LC) within a knowledge-based system with an intelligent interface, consider the kinds of tasks that it performs, look at diagrams of some of the information it uses, and close by examining the character of the interface between an LC and the rest of the system.

WHAT A LINGUISTICS COMPONENT KNOWS

The Position of the Component within the Larger System

Figure 1 shows the model that we use. The LC is a part of the user/system interface and is positioned analogously to that of the natural language comprehension module. Between the LC and the system proper, but still within the interface, is a text planning module. The LC and this planner are the pipeline through which information flows from the system to its user; they act as transducers, interpreting, transforming, and passing on information and intentions that originate within the decision-making apparatus of the knowledge-based system (KBS).
The KBS produces a specification, couched in the KBS's representation language, of the information to be given, and the "communicative force" it is to have. Figure 2 is an example of one of these specifications (note that most frames have more slots than are displayed in this figure). In this example, there is no special force to this specification beyond "communicate this information." The KBS plays the role of an attorney beginning to describe a case.

The effect is the same whether the specification is a new structure produced by the KBS specifically for the occasion, or whether it consists of relations among already existing objects in the KBS's knowledge base (as in this case). The specification is examined (and possibly modified and annotated) by the text planner, then passed incrementally to the LC for realization as a proper English text.

Text planning plays a crucial mediating role in its position between the KBS and the LC (see the work of McDonald, Woolf and McDonald, and Cook, Lehnert and McDonald for detailed discussion). Suffice it to say, the principal responsibilities of this intermediate component are to determine what parts of the specified information can and should be omitted from the text because the user will infer them from other things that are said, and to organize the information in the best manner for presentation in textual form. This entails adding rhetorical specifications, determining the sequence in which its information will be presented, and making some of the decisions about what words will be used.

What Tasks the Linguistics Component Takes On

In the task context that the specification of Figure 2 normally has in our research project, the text generated for it is as follows.

I represent a client named RCA Victim who wants to sue SWIPE Inc. and Leroy Soleil for misappropriating trade secrets in connection with software developed by my client.

This is the unmarked realization produced without any additional annotation of the specification by the text planner; its form thus derives entirely from the actions of the LC, given its present default settings. What have these actions done? A look at Figure 2 shows that nothing very profound has been done in the way of selecting English words for terms in the representation. Nearly every word in the output text has a counterpart in the name of a slot or a frame. This partly reflects our concentration on structural and grammatical problems rather than on concept-to-word mapping; it also reflects the genuine preference that we have for working with representation languages that encourage having a large number of very specific primitive terms whose interrelations and commonalities are given by explicit (and possibly redundant) relations, rather than with languages that encourage the decomposition of concepts into complex expressions involving a very small number of primitives.

Choice of vocabulary aside, the contributions of the LC to our text involve the following categories of "linguistic" decisions and background knowledge.

Style

The five framed KBS specification was realized as a single sentence; this, and the decision to say a client named and in connection with is a matter of adhering to a particular prose style, in this case that of one lawyer in consultation with another. By definition, choices of wording or sentence construction that are dictated by style cannot be determined by conceptual meaning (i.e., they will not follow from anything in the specification from the KBS). The style is instead determined by a concept-independent set of rules maintained in the LC and triggered by purely linguistic descriptions of the growing text (e.g., prefer complex over simple sentences, or prefer role-related descriptions over actual names).

Syntactic and morphological details

The conceptual-level representation maintained in the KBS is obviously not organized in the way that a natural language text is; once mapped into English words (mostly by the prior planning component), it must be linearized, morphologically marked, and if speech is to be produced, it must include its syntactic organization into clauses and phrases in order to initiate the correct intonational patterns. Nongrammatically-based generation components, for example those using the direct production technique typical in present KBSs, can accomplish linearization through the use of templates. However, morphological markings are quite difficult to achieve by ad-hoc means; even making subject and verb agree can be a programming ordeal, and marking phrases is impossible.

In a linguistically knowledgeable system like our LC, a thorough syntactic description of the text is constructed as an integral part of its generation. Figure 3 shows a portion of the syntactic tree that is constructed for the example text. The bulk of its detail is intended to constrain and direct the decision-making process that selects texts for incremental parts of the KBS specification. It serves a second purpose of indicating the morphological markings that the words should have. The label marked-gerundive-complement will force the verb it dominates (misappropriate) to take on its -ing form when it is spoken; in other contexts, alternative labels would force the infinitive, present tense, etc., all without any other part of the LC needing to be aware of the process. So-called grammatical function words such as to or for are introduced.
into the text stream as a reflex of its linguistic description without having to be deliberated over anywhere else.

Grammatically-forced restructuring of information

When concepts are mapped into words, the particular choice of wording that is made, and the crucial syntactic context in which it appears will constrain what information must be included or left out. A text planning system sometimes needs to be aware of these constraints when it chooses the words (for which purpose they should be simply stated in an easily manipulated representation), but more often the differences will not matter and should be left to the LC to carry out automatically. For example, when a verb appears in a main clause, it must carry information about the placement of its event in time that cannot be included when it is embedded as a complement, regardless of whether it was important to the KBS specification or not. The subject of a verb must be omitted in many complements, or sometimes a later argument suppressed because it is linked to a grammatically strategic element in a question or conjunction. From the KBS point of view, such things are details that it should not have to worry about. The rich linguistic representation that is used within the LC makes this possible by associating editing procedures, operating in the background, with the relevant terms in the representation and having them perform the restructuring "sight unseen" by the other components of the system.

Manifesting the speaker’s intent

Different text structures invariably carry with them different implications as to the relative importance or salience within the overall message of the information they convey. We know from our own writing that the same information should be expressed in different ways depending on the point we are trying to make. The question then is, how much knowledge of syntactic idiosyncrasies does the text planner or the KBS need to have in order to ensure that the most appropriate text structure will be used? With a well-designed LC, the answer to this question is “none at all,” because the LC will be using just this kind of knowledge to interpret the intentions given in the KBS specification, and to direct the selection of the text structure.

With this kind of division of labor, the KBS is able to couch its specification using terms like emphasize, contrast, or indicate similarity of relationship. The LC will understand these directives in terms of their implications for how the text is to be organized, for example, shifting the grammatical role of an object or choosing a relative clause over a new sentence as needed.

Maintaining cohesion in a discourse

Once a program needs to produce texts that are longer than a single sentence, or once the conversation consists of more than simply answering isolated questions and earlier statements cannot simply be forgotten, then a new kind of “grammaticality” becomes important, namely the manifestation through choice of wording and text structure of the links between the content of the present utterance and those that have preceded it. Having initially referred to a client named RAVICTIM, for example, the next reference to that KBS object should use the same kind of description (e.g., my client) or else it will either confuse the audience as to who is being referred to, or draw unintended attention to itself.

A well-designed LC can do this completely on its own without requiring the KBS to maintain any awareness of the discourse context, or to change the way it organizes its specifications. The LC maintains a representation of the relevant parts of the earlier discourse that is analogous to its representation of the text structure it is working on (Figure 3). It uses the former representation to decide when pronouns should be used or when descriptions should be abbreviated because part of their information will still be remembered. These rules of coherency are folded into the rules for selecting text structures by the writer of the LC, and participate in the text-level decision-making on an equal footing with the intentional directives.

Why the Lack of a Linguistics Component Hasn’t Bothered Most KBSs to Date

Only a few KBSs presently incorporate an LC, and probably only because the researchers who built them have had an independent interest in the problems of language generation. Present systems rely instead on template-based direct production from their internal representations along with as much ad-hoc grammar as their programmers have time to accomplish. This has worked thus far only because it has had to work (there have not been off-the-shelf LCs available for incorporation with minimal effort), and because the demands that actual KBS developers have placed on natural language output have been minimal; present KBSs still carry out only one or two system/user tasks, thus, their internally represented objects and relations have only needed to be presented in a single way which could then be captured in a template. Because the concepts that present KBSs have employed call for only simple syntactic realizations (e.g., there have been almost no systems yet that have had to talk about expecting or
persuading), the syntactic contexts that their text templates impose on the variables that get substituted into them have been very simple and have not called for any sophisticated syntactical transformations from their normal format. Without an LC you can instantiate templates, but not easily blend them together because the complexity of the grammatical adjustments that have to be made in order to keep the template combination grammatical goes well beyond what can be achieved in an ad-hoc treatment. This problem, which applies to some of the substitutions that would be wanted in instantiating templates, too, is probably the most serious limitation on the template style of language generation. These two reasons will be valid only a little while longer, however, so the designers of KBSs should begin to consider what it would be like to work with a full-scale LC.

INTERFACING TO A LINGUISTICS COMPONENT

If the builders of KBSs are to gain from the eventual availability of some off-the-shelf linguistics component from another research group, they will need to understand the kind of constraints that the LC is going to place on their designs. Assuming that the (sometimes overriding) mechanical interfacing constraints have been met (e.g., both KBS and LC are available in the same programming language dialect and consequently can be loaded simultaneously into the same program image), the remaining constraints are of two kinds, formal and ontological.

By formal constraints, I mean restrictions or preferences on the kinds of representational structures that are used for the KBS specification of what it wants to be communicated. We can always assume that the purely data-structure level compatibility of the interface will be trivially taken care of (e.g., we will not care whether a specification is given as a list of symbols, an array, a structure of flavor instances, etc.). For operations at that level, the LC will have defined a set of interface functions to do the actual movement of data between components; each new KBS will supply its own definitions for those functions, thus abstracting away from the idiosyncratic implementation decisions that went into programming the components.

The more substantial representational questions remaining can be thought about in terms of the implications of the choice of representation language within the KBS; different established languages fall into different categories according to whether their properties make them easy or difficult for an LC to deal with. In the case of my LC, experience with a wide range of representation languages including the predicate calculus, Lisp symbols and property lists, PLANNER-style assertions, the “frame” languages OWL and FRL, KL-ONE, home-brew type/token systems based on Lisp structures, a part/whole hierarchy built out of Flavors, and a version of RLL has been acquired. Two conclusions can be drawn from this body of experience; first, that object-based representations are preferable to expression-based representations; and second, that an efficiently implemented specialization hierarchy and meta-level are great time-savers.

The difficulty with expression-based representations is that no single manipulable locus can be identified with each of the represented entities or relations in the system; instead, their identity is distributed across a set of otherwise disconnected expressions. In order to manipulate such distributed objects, an LC must continually parse and reparse them, or else maintain a redundant shadow representation for them on its own. An LC needs to continually ask itself questions of the general form, “Have I seen this entity before and if so in what linguistic context?” Such questions are quite burdensome when working with a representation that does not maintain a unique first class object for each item in its world.

A facility for representing meta-level relationships allows an LC to record generation-specific information that is applicable to patterns in the way entities are represented by reifying the pattern and making it an object like those at the ground level. A specialization or AKO hierarchy provides a similar service by allowing generation tactics that apply to a great many related entities to be stated compactly, with variations captured through parameterization and specialization of associated generation information in parallel with those of the entities in the domain.

Weaknesses in the formal structure of a representation (from a generation perspective) will make dealing with the representation awkward for an LC, but not impossible; however, weaknesses in the ontology that it supplies for the world it is representing can impose severe restrictions on the quality and fluency of the language an LC can produce for it. Natural language has evolved to carry the communicative load that humans place on it, and to make it easy to express the relationships and attributes that humans see in the world and consider important. If a KBS does not have the same breadth of knowledge that a person does for a given subject matter, it will be at a loss when the language demands a value for an ontological category that people always know, but the KBS does not; you cannot talk about things you don’t know. All KBSs, regardless of their choice of representation language, will stumble over this point indefinitely because humans support a much richer system of distinctions and kinds in their thought and use of language than any yet captured in a knowledge base. The solution to this problem will not come from technical improvements in the design of LCs, but is an issue to be resolved by philosophers, psychologists, and people doing basic AI work on knowledge representation.

REFERENCES