Integrating Text Planning and Linguistic Choice Without Abandoning Modularity: The IGEN Generator

Robert Rubinoff*

Natural language generation is usually divided into separate text planning and linguistic components. This division, though, assumes that the two components can operate independently, which is not always true. The IGEN *generator eliminates the need for this assumption; it handles interactions between the components without sacrificing the advantages of modularity.* IGEN *accomplishes this by means of annotations that its linguistic component places on the structures it builds; these annotations provide an abstract description of the effects of particular linguistic choices, allowing the planner to evaluate these choices without needing any linguistic knowledge. This approach allows* IGEN *to vary the work done by each component independently, even in cases where the final output depends on interactions between them. In addition, since* IGEN *explicitly models the effects of linguistic choices, it can gracefully handle situations where the available time or linguistic resources are limited.*

1. Introduction

Programs that generate natural language have generally been divided into two largely independent components: a text planning component that organizes the information to be expressed, and a linguistic component that converts the information into grammatical sentences of some natural language (e.g., Thompson 1977; McKeown 1985; McDonald 1983; Meteer 1989; Reithinger 1990; Dale 1989). This division seems natural, and has indeed proven useful, because the two components deal with different kinds of data and involve different kinds of reasoning. The planning component reasons about information and text structures to plan a coherent discourse, while the linguistic component arranges the lexical elements and syntactic constructions available in some language to ensure the resulting text's grammaticality and comprehensibility.

This division into independent components, though, is only tenable if the decisions each component must make can be made independently. If some of the generator's work involves both text planning and purely linguistic issues, there will be no way to make the necessary decisions without violating the generator's modularity. As we shall see, natural language generation does indeed involve such decisions. The need to handle interactions between text planning and linguistic concerns thus seems to require either abandoning the division into separate components (e.g., Danlos 1987; Kantrowitz and Bates 1992), with the resulting increase in complexity, or foregoing the ability to handle these interactions at all (as advocated in Reiter [1994]). A third option is to identify possible interactions in advance and handle them as special cases, but this only allows the generator to handle those interactions that have been antici-

* AnswerLogic, Inc., 1111 19 St. NW, Suite 600, Washington, DC 20036. E-mail: rrubinoff@answerlogic.com. The research described here was done at the University of Pennsylvania and Carnegie Mellon University.

pated in advance. Furthermore, handling the interactions often requires violating the modularity of the system, because detecting when the special case has occurred still requires both planning and linguistic knowledge.

The IGEN generator solves this problem. IGEN handles interactions between the text planning and linguistic components without having to sacrifice any of the generator's modularity. The key to IGEN's approach is the use of annotations that the linguistic component attaches to each linguistic expression it constructs. These annotations abstract away from the details of the linguistic expressions, describing only those properties of the expressions that are potentially relevant to the planner. The planner can then evaluate the choices made by the linguistic component and determine how those choices interact with the text plan independently of the linguistic component's processes and data structures. As a result, IGEN can make decisions involving interactions between the components while retaining complete modularity. In fact, replacing IGEN's normal linguistic component with one for a different language involves no change in the planner despite the fact that the two languages have *different* idiomatic expressions for some of the sentences being generated; the planner's processing is identical in both languages. Furthermore, since the annotations allow IGEN to explicitly model the effects of its decisions, IGEN is able to gracefully handle the effects of limitations on processing time or linguistic resources; since IGEN always knows how well a given linguistic construction carries out its plan, it can choose the best available construction when it can't find an ideal one.

2. Modularity in Generation

Research in natural language generation has generally separated the task into distinct text planning and linguistic components. The text planning component selects and organizes the information to be expressed in some internal representation and then sends it to the linguistic component, which converts the information from the internal form into grammatical sentences of some natural language. The names given to the components vary; they have been called "strategic" and "tactical" components (e.g., McKeown 1985; Thompson 1977; Danlos 1987)[1], "planning" and "realization" (e.g., McDonald 1983; Hovy 1988a), or simply "what to say" versus "how to say it" (e.g., Danlos 1987; Reithinger 1990). The precise division of work between the components can also vary, as can the extent to which the text planner is responsible for selecting (as opposed to merely organizing) the information to be expressed. Much (if not most) work in generation, though, continues to rely on this modular approach for its basic design. For example, DIOGENES (Nirenburg et al. 1988), EPICURE (Dale 1989), SPOKESMAN (Meteer 1989), Sibun's work on local organization of text (Sibun 1991), and COMET (Fisher and McKeown 1990) all are organized this way. McDonald has even argued for extending the model to a large number of components (McDonald 1988), and several systems have indeed added an additional component between the planner and the linguistic component (Meteer 1994; Panaget 1994; Wanner 1994). Reiter describes a pipelined modular approach as a consensus architecture underlying most recent work in generation (Reiter 1994).

As this large body of work makes clear, the modular approach has been very useful, simplifying the design of generators and making them more flexible. In fact, in at least one case the "tactical" component of a generator was successfully replaced with a radically different independently designed one (Rubinoff 1986). A modular design,

1 Danlos uses "syntactic" rather than "tactical"; see the note on page 122 of Danlos (1987).

though, presumes that the work done by each module can be done independently. Separating generation into text planning and linguistic components thus implicitly assumes that text planning can be done without knowledge of the language being used and, conversely, that linguistic choices can be made without text planning knowledge. Unfortunately, this is not always true; both structural and lexical choices sometimes depend on interactions between the two parts of the generator. Thus generation must either compromise the modularity between the components or give up the ability to handle these cases properly.[2]

2.1 Interactions between the Modules

The modular approach to generation assumes that the linguistic component's decisions never matter to the planner (or whichever component(s) organize the information to be expressed). This is not the case, though, as can be seen from the alternations in (1)–(3):

(1) a. John killed him with a gun.

b. John shot him dead.

(2) a. John infected him with a virus.

b. *John virused him sick.

(3) a. *John homed him with an order.

b. John ordered him home.

The sentences (1a) and (1b) express essentially the same information, so if the generator is attempting to express this information, it must choose between them at some point. In a modular generator, though, there is no point at which the decision can be made. The planner can't make this choice, because the availability of the choice depends on the particular linguistic resources of English. This can be seen by comparison with (2) and (3), in which only one alternative is available. In fact, a different alternative is available in each case. Since the planner doesn't know which alternative(s) is/are available, it can't choose between them; the linguistic component must make the choice.

On the other hand, the decision *has* to be made by the planner, since it can depend on and/or affect the goals the generator is trying to achieve. The choice between (1a) and (1b) should depend (in part) on what the generator is primarily trying to talk about. (1a) is more appropriate if the generator is going to continue talking about the gun, whereas (1b) is more appropriate if the main concern is the ramifications of the victim's death. Since the planner is the component that deals with this information, it must choose between the alternatives.

Also, the choice between (1a) and (1b) determines what information can be easily omitted; cutting off the end of the sentence leaves out mention of the use of a gun in (1a) and the death of the victim in (1b). Since the planner knows the consequences of omitting information, it must make the choice of which alternative to use and whether to abbreviate it. It might seem that the planner could simply indicate exactly what

2 Note that these interactions aren't the result of the particular details of how the work is divided between the components. As we shall see, there are some decisions that depend on both the underlying goals driving the generator and the details of what can be expressed in a particular language. Any architecture that deals with these issues in different components will encounter the problems described below.

information it wants included in the utterance, but that would require the generator to always assume a strategy of saying as little as possible.

Furthermore, decisions about what information to include may interact with other decisions. For example, the generator may want to emphasize the victim's death but not care about the means of death; it might then choose (1b) for the emphasis even though (1a) would let it skip mention of the gun. This kind of decision can only be made by the planner.

The same kinds of interactions arise in the process of lexical choice. It would seem that lexical choice has to be handled by the planner, since it depends very much on what the generator is trying to accomplish. For example, the choice of describing someone as either *firm*, *obstinate*, or *stubborn* should depend on what else the generator wants to say about the person, as should the choice between *meek* and *wimpy*. The generator might describe how justice was served by an *execution* rather than how the prisoner was *murdered* by the state. Similarly, the generator might deride the comments of a *dreamer*, but praise the insights of a *visionary*. These kinds of lexical choices can only be made by the component that handles the generator's goals.

On the other hand, there are a number of reasons why lexical choice has to be handled by the linguistic component. First of all, lexical choice is very dependent on the particular linguistic vocabulary of the language being generated. Thus French, for example, uses two different verbs (*connaître* and *savoir*) to express knowing a person and knowing information, but English just uses *know* for both concepts. Similarly, English uses *to be* to indicate both location and a state or property, whereas French uses *se trouver* for the former and *être* for the latter.

Furthermore, there is in general no guarantee that there will be any lexical item to express a given concept. For example, there is no word in English for the concept of a car with a removable door. There's no inherent reason why there couldn't be; after all, there's a word for a car with a removable roof. This is just a particular fact about English. Similarly, there is a word *giant* meaning "large man", but no corresponding word meaning "large car".

In addition, since lexical choice interacts with syntactic decisions, it cannot be done in advance of choosing syntactic structures. For example, a generator cannot decide to use *probable* instead of *likely* without knowing if the completed utterance could be the ungrammatical *he is probable to be early*. Similarly, the verb *drink* can't be chosen without knowing whether the clause will have a direct object; *he drinks apple juice* and *he drinks* actually have quite different meanings.[3] Note that the decision here doesn't depend just on whether the beverage is going to be explicitly mentioned; it depends on whether it's going to be mentioned in a specific syntactic position in the sentence. So here too it is impossible to assign the decision to a single component; the decision must be made by both components.

The need to handle interactions such as these forces compromises in the modularity of generators. In the TEXT system (McKeown 1985), for example, some decisions about what information to include are in fact encoded into the tactical component. For example, TEXT's tactical component omits the attribute value **WATER** (used in TEXT to indicate that some object travels in or under the water) when it must be

3 McDonald has argued that lexical choice should be done in the first step in generation; in cases where lexical and syntactic decisions interact, lexical choice will constrain subsequent syntactic decisions (McDonald 1991). This approach is certainly possible (although it's not clear how to prevent independent lexical choices from imposing incompatible syntactic constraints), but it assumes that the resulting syntactic constraints don't matter to the generator. For example, choosing *drink* may require the generator to include a direct object even if it would prefer not to indicate that information; by the time the generator discovers this requirement, it is already committed to the choice.

expressed as an adjective. This is because the only available adjective is the somewhat awkward *water-going*, and it turns out that in the sentences TEXT happens to generate, **WATER** only appears in an adjective context in cases where it's not important to say it. Thus, this strategic decision (that **WATER** can be omitted) is encoded permanently into the tactical component; there is no way for the strategic component to control this decision.

In MUMBLE (Meteer et al. 1987), in contrast, the interactions have pushed linguistic information into the text planner. For example, MUMBLE can take the already constructed phrase *Fluffy, Floyd's dog, buries bones* and modify it with the information that this was reported by Helga to produce *Helga reported that Fluffy, Floyd's dog, buries bones*. But in order to do this, it has to mark the information about Helga with the **new-main-clause** feature. So the planner has to know what clauses are, know that the earlier information was turned into a clause, and know that making *Helga reports* a new main clause is a useful thing to do. Many of the linguistic decisions are thus actually being made by the planner.

2.2 Facing the Dilemma

We are now faced with a dilemma. On the one hand, the separation of planning and linguistic realization into distinct components seems natural and useful. On the other hand, it precludes making decisions involving interactions between text planning and linguistic issues.

One possible response would be to abandon the separation; the generator could be a single component that handles all of the work. This approach has occasionally been taken, as in Kantrowitz and Bates (1992) and Danlos (1987)[4] and, at least implicitly, in Paris and Scott (1994) and Delin et al. (1994); however, under this approach, all of the flexibility and simplicity of modular design is lost.

The opposite approach is to simply ignore the limitations of a modular design and proceed as if there need be no interactions between the components. Whatever problems result will be handled as best they can, on a case-by-case basis. This approach is the one taken (implicitly or explicitly) in the majority of generators. In fact, Reiter has even argued in favor of this approach, claiming that the interactions are sufficiently minor to be ignored (or at least handled on an ad hoc basis) (Reiter 1994). While this certainly has appeal as a design methodology, it seems reckless to assume that problems will never appear. Certainly an approach to generation that *does* handle these interactions would be an improvement, as long as it didn't require abandoning modularity.

There have in fact been attempts to develop modified modular designs that allow generators to handle interactions between the components. These include devices such as interleaving the components (McDonald 1983; Appelt 1983), backtracking on failure (Appelt 1985; Nogier 1989), allowing the linguistic component to interrogate the planner (Mann 1983; Sondheimer and Nebel 1986), and Hovy's notion of restrictive (i.e., bottom-up) planning (Hovy 1988a, 1988c). All of these approaches, though, require that potential interactions be determined either by the tactical component or by the system designer in advance. The text planning component still has no way to detect and respond to unanticipated interactions on its own initiative.[5]

4 Danlos still has a separate low-level "syntactic" component, but essentially all of the generator's decisions are made by the strategic component.

5 In fact, adding additional components may make the problem even worse, as decisions may then be spread across three or more separate components. It might seem that a component placed between the planner and the realizer could handle interactions between the levels, but this would simply recreate the original problem within the "middle" component; coping with the interactions would still require dealing with the whole range of issues that come up in both planning and realization.

The IGEN generator resolves the modularity dilemma by means of a new approach that preserves complete modularity but allows the generator to handle interactions between the components. The key to IGEN's approach is that the linguistic component provides the planner with an abstract description of the effects of using particular linguistic structures. This abstract description is encoded in annotations that the linguistic component attaches to the structures it builds. In contrast with previous feedback-based approaches, in which the feedback only occurs when the linguistic component needs additional help from the planner, IGEN annotates *all* of its linguistic structures. This allows the planner to detect situations where linguistic choices cause problems at the planning level; the linguistic component need not be aware of them. Conversely, the planner can build communicative plans without concern for whether the linguistic component can actually express the elements of the plan; any such problems will be detected by the linguistic component and reflected in the annotations. The annotations allow a cycle of negotiation between the planner and the linguistic component; each component works entirely within its own level, drawing only on the *results* of the work at the other level.

3. IGEN: A New Model for Generation

The IGEN generator overcomes the limitations of the modular approach, without giving up its advantages, by the use of annotations that provide feedback from the linguistic component to the planner. These annotations abstract away from the details of the linguistic expressions, allowing the planner to detect interactions between decisions at the linguistic and planning levels without having to know anything about the knowledge or reasoning used by the linguistic component. The planner can then respond to these interactions by choosing among the expressions proposed by the linguistic component, using the annotations to evaluate how particular linguistic choices affect the successful achievement of the goal(s) driving the generator. This approach allows IGEN's components to interact while preserving the strict separation between their knowledge and processing.

The annotations are predicates that apply to the linguistic expression they are attached to, indicating some effect of or property of the expression. Most of them also take an explicit argument indicating a semantic expression whose relation to the option is indicated by the annotation. The types of annotations used by IGEN are shown in Figure 1. They fall into the following general categories:[6]

Meaning-based These relate the meaning of a linguistic option to the information it is supposed to express, both in terms of how explicitly and how completely the information is expressed.

Contextual These indicate dependencies and effects on the presence and status of elements in the discourse context, including not only availability for anaphoric reference but also effects on how various elements in the discourse context will be perceived.

Pragmatic These indicate various pragmatic and stylistic features of the option.[7]

6 There are also some annotation types used to keep track of IGEN's internal processing; these are not shown here.

7 Note that IGEN currently only uses one annotation of this type: concise-construction, used to mark constructions that are particularly concise.

Meaning-based:
```
makes-explicit
makes-implicit
indirectly-suggests
missing-info
extra-info
```

Contextual:
```
activates-in-context
from-context
relates-to
```

Pragmatic:
```
concise-construction
tone
marks-as-focus
```

Figure 1
Annotation types.

The IGEN generator uses these annotations to allow for interactions between its various components, shown in Figure 2. IGEN works as follows: The communicative planner responds to goals given to the generator by building an appropriate plan, which will contain a number of communicative actions, i.e., actions that communicate information, and places requests for ways to express that information into the workspace. The linguistic component then responds by producing a series of annotated linguistic options for each of the planner's requests. These are also placed in the workspace, where the planner can use the annotations to evaluate and rank the options and also, when appropriate, to modify the plan based on what the linguistic component is able to produce.

This process doesn't actually produce any output, though, since there is no way to determine when it is "finished." The linguistic component can always come up with yet another way to express something, and the planner can always do more reasoning about the interactions between some expression and the planner's specific and general goals. There must therefore be another component, here called the utterer, that is responsible for assembling utterances from the options preferred by the planner and shipping them off to be spoken or written. The utterer's job is to balance the planner's preferences against time pressure, producing output when whatever deficiencies the options have are outweighed by the consequences of remaining silent. The various components thus work together in parallel, incrementally building, improving, and producing utterances. Each component of the generator handles a particular aspect of the generation task, involving different kinds of knowledge and different constraints on the generator's work.

IGEN in some ways resembles a blackboard architecture (Nii 1986b, 1986a); like a blackboard, its workspace contains several possible utterances that are described at different levels of structure and are built up incrementally by various independent components. On the other hand, it doesn't have the strict hierarchical structure that usually characterizes blackboard systems; each request from the planner may correspond to several linguistic options, and the linguistic options may handle (parts of) more than one request. Also, blackboard systems usually have sequential scheduling of actions, whereas IGEN's planner, linguistic component, and utterer all run in parallel.

Furthermore, the relevant issue here for IGEN is whether there are real interactions between the modules, which a blackboard-style approach to generation doesn't

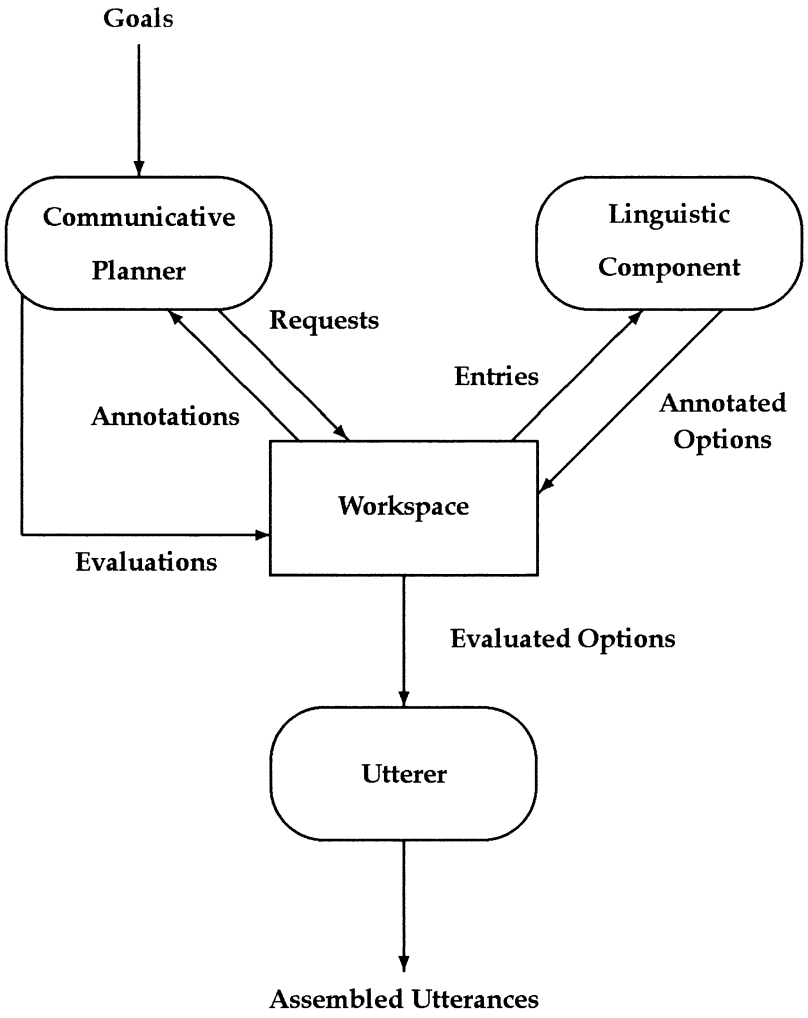

Figure 2
Architecture of the IGEN generator.

guarantee. Blackboard-based generation can still be structured in a way that makes such interactions impossible, either explicitly, as in DIOGENES (Nirenburg et al. 1988), which uses several blackboards successively, or implicitly, as in GLINDA, in which successive stages of rules depend on the results of previous stages for their triggers.[8]

3.1 The Communicative Planner

This is a special-purpose planner that accepts goals from the overall system and plans out ways to achieve them via communication. These can be "communicative" goals such as "transmit this information to the user" (or more precisely "get the user to believe this information"), but they can also be social interaction goals such as "establish a deferential (or collegial, or superior) attitude toward the user" or cooperative behavior goals such as "be straightforward" or "transmit as little information as possible"

8 As can be seen from the diagram on page 24 of Kantrowitz and Bates (1992) and the accompanying discussion.

or even more general goals such as "get the user to take his umbrella" or "convince the user to improve his diet."

The communicative planner draws on a set of specialized plans that capture how communication acts can achieve various goals. It draws on knowledge of the effects of communication to plan out how to achieve various kinds of goals by expressing various kinds of information. Since it has to evaluate how well particular linguistic expressions further those goals, the planner needs to build an explicit representation of its plan(s) that indicates how particular actions contribute to their achievement. The planner depends on the linguistic component to tell it how to carry out these actions, i.e., how to express particular information in language.

While the planner is specialized for planning communicative actions, it can in principle draw on anything in the system's general body of knowledge and belief that might support the goals it is given. Thus (as we shall see) in the second example in Section 4.3, the planner decides to mention the rain not because it bears any special relationship to anything in the input goal, but rather because mentioning it is part of a plan that achieves the goal. Even in cases where the planner's goal is simply to transmit some specific information, it can include any additional information that will help that transmission to succeed. The planner's role is not just to organize information, or even to collect information on some topic and organize it, but rather to identify a set of communicative acts that will achieve some goal.

3.1.1 Building an Initial Plan. The communicative planner starts off by building an initial plan to achieve the goals it is given. The form of that plan is important. It cannot be just a (partially or totally) ordered set of information to go into the text, as is the case in many generators. Since IGEN's planner needs to evaluate how well linguistic options achieve the plan's goals, the plan must record what each piece of information in the plan is intended to accomplish and how it supports the purpose of the larger text containing it. Sometimes the planner's goal may simply be to convey certain information to the user, but there may also be more indirect goals that conveyance is intended to achieve. Thus the plan must indicate in detail *why* the information is being expressed. An example of the type of plan the planner must build is shown in Figure 3 (Section 4.1 discusses the use of this plan in more detail). This model of text plans meshes well with the model of discourse structure developed by Grosz and Sidner (Grosz and Sidner 1985, 1986), in which the purpose of each discourse segment is an important part of the structure.[9]

IGEN constructs its plans using a hierarchical planning algorithm (Nilsson 1980). The planner first checks all of its top-level plans to see which have effects that match the goal. Each matching plan's preconditions are checked; if they are currently (believed to be) true, the planner then attempts to find all instantiations of the plan's body.[10] The body of a plan can be an action or sequence of actions, a goal or sequence

9 Moore and Paris also note that "a generation system must maintain the kinds of information outlined by Grosz and Sidner" (Moore and Paris 1989, 203). Their planner uses plan structures similar to IGEN's, except that the plan operators they use are generally instantiations of rhetorical relations drawn from Rhetorical Structure Theory (Mann and Thompson 1987). In IGEN, the plans can involve any goals or actions that could be achieved via communication.

Hovy has described another text planner that builds similar plans (Hovy 1988b). This system, however, starts with a list of information to be expressed and merely arranges it into a coherent pattern; it is thus not a planner in the sense used here (as Hovy makes clear).

10 Since text planning was not the primary focus of this work, IGEN is designed to simply assume that any false preconditions are unattainable. IGEN's planner divides the requirements of a plan into two parts: the preconditions, which are not planned for, and those in the plan body, which are. This has no

1. Overall Goal: Know(Hearer,[weather information over next few days])
2. Achieve (1) by achieving:

   Know(Hearer,[first day's information])
   Know(Hearer,[second day's information])
   ⋮
   Know(Hearer,[last day's information])

3. Support (2) by performing: Utter([first day's information])
4. Support (2) by performing: Utter([second day's information])

   ⋮

N. Support (2) by performing: Utter([last day's information])

Figure 3
A sample initial plan.

of goals, a subplan, or a goal decomposition.[11] To instantiate the body, the planner accordingly collects the action(s), recursively plans for the goal(s), instantiates the subplan, or carries out the goal decomposition. The net result of this process is a list of possible complete plans that achieve the original goal.

3.1.2 Responding to Linguistic Options. Once the communicative planner has built up an initial plan of what it wants to say, it puts requests for the individual bits of information contained in the plan into the workspace for the linguistic component to respond to. The planner, however, cannot assume that its requests will be followed precisely; the linguistic component may not be able to find a way of completely satisfying them. The planner must therefore examine and evaluate the responses that the linguistic component puts into the workspace (as described below in Section 3.2). These responses consist of linguistic structures together with the annotations that provide descriptions of how closely the option carries out the request(s) it responds to, as well as any other information or effects arising from the use of the option. Drawing on these annotations, the planner evaluates how well the various options fit into the plan and assigns them appropriate ratings along a scale from `VERY-HIGH` to `VERY-LOW`.

The evaluation algorithm is divided into two phases (as shown in Figure 4), which work roughly as follows: The first phase looks at how well the option matches the request, checking first to see whether the option misses any requested information; if there is no missing information, the rating is based on annotations indicating the relationship between the option's meaning and the request. The second phase then adjusts the option's rating based on how any missing or added information interacts with the plan structure. This ability to adjust preferences among linguistic options based on planning issues allows IGEN to overcome the usual limitations of a modular generator design. Without the annotations (or some similar feedback mechanism), the planner would have to supply the linguistic component with any information that might be relevant to choosing among the options. This could potentially include

practical effects, since the preconditions, by and large, are types of goals that IGEN doesn't have any plans for; any attempt to plan for them would immediately fail.

11 A goal decomposition is a way to decompose a goal into subgoals that can be planned for independently. For example, the **utter-sequence** plan works by decomposing the information to be uttered according to some relevant ordering.

EVALUATE(**Option,Request**)

Phase 1:

1. Let **Covered-Pieces** =
   UNION(**Request**,arguments of any `COVERS-OTHER-ENTRY` annotations).
2. If **Option** has a `MISSING-INFO` annotation whose argument is a member of **Covered-Pieces**, or has more than one `MISSING-INFO` annotation, evaluation is `:VERY-LOW`.
3. Else if **Option** has any `MISSING-INFO` annotations, evaluation is `:LOW`.
4. Else if **Option** has any `INDIRECTLY-SUGGESTS` annotations, evaluation is `:MEDIUM`.
5. Else if **Option** has any `MAKES-IMPLICIT` annotations, evaluation is `:HIGH`.
6. Else evaluation is `:VERY-HIGH` (because **Option** must have a `MAKES-EXPLICIT` annotation).

Phase 2:

1. Let **Adjustment** = 0.
2. For each `MISSING-INFO` annotation attached to **Option**: Subtract 1/2 from **Adjustment** if its argument is an item in the plan (1 if it's a critical item).
3. For each `ACTIVATES-IN-CONTEXT` or `EXTRA-INFO` annotation attached to **Option**:
   - If its argument is a critical item in the plan, add 1 to **Adjustment**.
   - Else if its argument supports or conflicts with a critical item in the plan or strengthens or weakens a goal or precondition of the plan or the head of the request, add or subtract 1 from **Adjustment** accordingly.
   - Else if its argument is a sequence, **Request** is a member of another sequence that is a critical item in the plan, and the two sequences are along the same scale but are disjoint or only overlap slightly, subtract 1 from **Adjustment**. For example, the sequences [1, 2, 3, 4] and [4, 5, 6, 7] would trigger this adjustment, but the sequences [1, 2, 3, 4] and [2, 3, 4, 5] wouldn't.
   - Else if its argument is an element in the plan, add 1/2 to **Adjustment**.
4. Adjust **Option**'s Phase 1 evaluation up or down **Adjustment** levels (ignoring 1/2s), e.g. an **Adjustment** of -2 or - 21/2 would change `:HIGH` to `:LOW`, and add an `ADJUSTED-UP` or `ADJUSTED-DOWN` annotation to **Option** as appropriate.

Note: a "critical" item in a plan is one that plays a central role in the structure of the plan, e.g. the basis of a goal decomposition or an action, goal or plan sequence.

Figure 4
The linguistic option evaluation algorithm.

anything in the plan, so the linguistic component would have to understand and reason about every aspect of the plan, defeating the whole purpose of a modular architecture.

In addition to evaluating options, the planner looks for opportunities to revise the plan based on the work of the linguistic component. For example, the same linguistic structure may appear as an option for several different parts of the plan, or an option for one part of the plan may also express information contained in (or related to)

another part of the plan. The planner can reorganize the plan to take advantage of these kinds of situations.[12] Thus the communicative plan can be modified based on linguistic information, again without violating the strict modularity of the generator.

3.2 The Linguistic Component

The linguistic component is the part of IGEN that produces the actual bits of language that make up the utterance. It acts like a "consultant" to the communicative planner, providing it with linguistic expressions that attempt to capture part or all of the information the planner wants to express.[13] The linguistic component monitors the workspace, looking for requests from the planner, to which it responds by suggesting linguistic expressions that capture some or all of the information in the request. These expressions are placed in the workspace as options for the planner to evaluate; the linguistic component continues suggesting options for a request as long as that request remains in the workspace.[14]

The linguistic options are pieces of surface structure; they can be at any level of grammatical structure and need only be partially specified. Thus an option could be a full sentence, phrase, or word, or a particular clause structure (e.g., a topicalization or an *it*-cleft) with no further detail filled in, or a noun phrase with a determiner but with the head noun left unspecified. The options are represented as feature sets indicating the various syntactic and lexical properties of the option.[15] For example, the following feature set represents the partial phrase "It will be ____**ADJP**" in which the predicate adjective has not yet been specified:

$$\begin{bmatrix} \text{CAT} & = & \text{S} \\ \text{SUBJ} & = & \cdots \texttt{\#<w:IT>} \cdots \\ \text{HEAD} & = & \begin{bmatrix} \text{CAT} & = & \text{V} \\ \text{LEX} & = & \texttt{\#<w:BE>} \\ \text{TENSE} & = & \text{FUTURE} \end{bmatrix} \\ \text{PRED} & = & \text{ADJP} \\ \text{ORDER} & = & (\text{SUBJ HEAD PRED}) \end{bmatrix}$$

The PRED role in this phrase has not been filled yet, so its value is ADJP, indicating that it must be filled by an adjective phrase.

The linguistic component is also responsible for annotating the options it puts into the workspace. The annotations are derived primarily from three sources: a comparison of the structure and position of the option's meaning and the request in IGEN's semantic network, the induced perspective shift of the option's meaning, and special annotations associated directly with the option.[16] The annotation algorithm works roughly

12 Another possibility would be for the planner to revise the plan when it detects problems with the options. For example, in a tutoring situation, if the linguistic component only returned options that involve concepts the student doesn't yet understand or know about, the planner might decide to revise the plan (perhaps to simplify what it's trying to say). IGEN doesn't implement this kind of plan revision, although the annotations provide the planner with the information necessary to do so.

13 IGEN's design allows for multiple linguistic components, each providing a different kind of linguistic knowledge, although only one was actually used; see Rubinoff (1992) for details.

14 As currently implemented, the linguistic component finds options by scanning through IGEN's semantic network, looking for items that are part of the meaning of some expression (Rubinoff 1992). The particular method used to produce the options, though, is incidental to the the overall functioning of the generator; it has no effect on the annotation or evaluation of the options, as the annotations depend only on the relationship between the option's meaning and the request.

15 There is also a MAP feature used to connect unfilled syntactic arguments with the corresponding semantic objects.

16 The induced perspective shift of the option's meaning is computed using a model of perspective built into IGEN's knowledge representation formalism; see Rubinoff (1992) for details.

as follows: First the linguistic component adds an annotation that indicates the kind of connection (if any) between the option's meaning and the request; this produces either a MAKES-EXPLICIT, MAKES-IMPLICIT, INDIRECTLY-SUGGESTS, or MISSING-INFO annotation. Next it adds EXTRA-INFO and MISSING-INFO annotations to indicate the differences between (the meaning of) the option and the request, and COVERS-OTHER-ENTRY annotations for any other requests that are included in the option's meaning. Finally, it adds ACTIVATES-IN-CONTEXT annotations to indicate any associated perspective shifts.

For an example, consider the situation described in Section 2.1 where the generator is trying to express the information that John shot and killed someone.[17] Among the annotated options produced to express #<SHOOT-AND-KILL(JOHN,BILL)> are:

- #<w:KILL>:
  (MAKES-EXPLICIT #<KILL(AGENT1,AGENT2)>)
  (MAKES-IMPLICIT #<SHOOT-AND-KILL(AGENT1,AGENT2)>)
  (MISSING-INFO #<INSTRUMENT(SHOOT-AND-KILL,GUN)>)
- #<w:SHOOT>:
  (MAKES-EXPLICIT #<SHOOT(AGENT1,AGENT2)>)
  (MAKES-IMPLICIT #<SHOOT-AND-KILL(AGENT1,AGENT2)>)
  (MISSING-INFO #<RESULT(SHOOT-AND-KILL,DEATH)>)

In each of these cases, there is a MAKES-EXPLICIT annotation for the meaning of the word, a MAKES-IMPLICIT annotation for the requested information, and a MISSINGINFO annotation for the property of the request that is not covered by the option. (Of course, MAKES-EXPLICIT annotations on the options that produce *with a gun* and *dead* indicate how the missing information can be added to the utterance.)

The full annotation algorithm is as follows:[18]

ANNOTATE(**Request**,**Option**)

1. Construct a MAKES-EXPLICIT annotation for each node or link in **Option**'s meaning and collect them in **Annotations**.
2. If **Option**'s meaning contains **Request** and nothing else, return **Annotations**.
3. If **Request** is an instance of an element in **Option**'s meaning and they have the same label, add a MAKES-EXPLICIT annotation for **Request** to **Annotations**.
4. Else if **Request** is an instance, subconcept, or subrange of an element in **Option**'s meaning or a position within (or identical to) an element in **Option**'s meaning, then add a MAKES-IMPLICIT annotation for **Request**, MISSING-INFO annotations for any links connected to **Request** that don't match links connected to the element in **Option**'s meaning, and EXTRA-INFO annotations for any links connected to the element in **Option**'s meaning that don't match links connected to **Request**.
5. Else if an element in **Option**'s meaning is an instance or subconcept of **Request**, add a MAKES-IMPLICIT annotation for **Request** and MISSING-INFO annotations as described in step (4) to **Annotations**.

17 While these annotations provide the necessary information for the planner to choose among the options as discussed in Section 2.1, IGEN's plan revision capabilities doesn't currently handle this case. As a result, IGEN currently produces either "John shot him dead" or "John killed him with a gun" for this example.

18 This description leaves out a few details; for example, in certain cases MISSING-INFO annotations are only generated for links considered "prominent" in the current perspective; see Rubinoff (1992) for details.

6. Else if an element in **Option**'s meaning is a "sibling" of **Request** (i.e., they are instances or subconcepts of the same element) or an "uncle" of **Request** (i.e., the sibling of an element of which **Request** is an instance or subconcept), add a `MAKES-IMPLICIT` annotation for **Request** and `MISSING-INFO` and `EXTRA-INFO` annotations as described in step (4) to **Annotations**.

7. Else if there is any other link between **Request** and an element in **Option**'s meaning, add an `INDIRECTLY-SUGGESTS` annotation for **Request** to **Annotations**.

8. Else add a `MISSING-INFO` annotation for **Request** to **Annotations**.

9. If **Option**'s meaning contains a node or link that is another part of the planner's request, then add a `COVERS-OTHER-ENTRY` annotation for the workspace entry containing the request to **Annotations**, and repeat steps (3) through (8) for that request.

10. Add `ACTIVATES-IN-CONTEXT` annotations for every element that would be shifted into perspective by an element in **Option**'s meaning to **Annotations**.

11. Add any special annotations associated with **Option** to **Annotations**.

12. Return **Annotations**.

3.3 The Utterer

The utterer is responsible for the final assembly and output of the utterance. Its main concern is balancing the desire for the most appropriate possible utterance against time pressure. Since the planner and the linguistic component build up and refine the utterance incrementally, they can continue working indefinitely; there's never a point where they can declare the utterance "finished" and quit working. On the other hand, IGEN can't go too long without uttering *something* or it will lose its turn, because the user will either utter (i.e., type) something else or get tired of waiting and leave (or quit the program). So the utterer must strike a balance between the advantage of improvements to the utterance that may yet be found and the disadvantage of continued delay.

The actual work of the utterer is fairly straightforward. It constantly monitors the workspace, checking the available options against a time-sensitive acceptability level. Whenever the best available utterance meets or exceeds the minimum acceptable evaluation, the utterer proceeds to output the utterance. The criterion of acceptability is determined via a set of delay thresholds that control how long to wait before lowering the minimum acceptable rating. For example, the default threshold settings are [0,1,2,12,16], meaning: if the time since the last utterance is more than 0, accept an option rated `VERY-HIGH` or better; if the time since the last utterance is more than 1, accept an option rated `HIGH` or better; and so on through all the ratings.[19] If the time since the last utterance is more than 16, then any option will be accepted; this essentially means that the generator would rather say something completely stupid than remain silent.

If any of the candidate utterances' evaluations meets or exceeds the current minimum acceptable rating, the utterer chooses the best candidate utterance and outputs it, removing the workspace entries it derives from. If there are two or more candidate utterances with the highest rating, the utterer breaks the tie by comparing the ratings

19 IGEN measures time in terms of cycles through its process scheduler, an arbitrary but workable approach that allows for consistent timing behavior.

of the individual options they are composed of. If there is still a tie, the utterer chooses, in order of preference, the candidate utterance that:

1. violates fewer constraints.[20]
2. was the only one whose rating was adjusted up by the planner, since this means there was something specific about the option that fit well into the plan.
3. is more concise, indicated by the presence of a CONCISE-CONSTRUCTION annotation.[21]
4. covers more entries in the workspace.
5. covers an earlier entry in the workspace.

If none of these criteria distinguish the candidate utterances, the utterer simply picks one at random.

4. Examples

The workings of the various components and representational machinery described above can be seen more clearly by looking at a few examples of IGEN at work. These examples demonstrate how IGEN can handle decisions that involve various kinds of interactions between linguistic and planning issues, despite the strict separation of the planning and linguistic components of the generator. In addition, the examples show how IGEN is sensitive to time pressure, improving its output when given more time to work and degrading gracefully when forced to work faster.

4.1 Supporting the Plan Structure

In this example, the generator is given the goal #<KNOW(HEARER,WEATHER-INFO)>, where #<WEATHER-INFO> is an unordered collection of facts about the weather over a three-day span starting on the current day. The final output in response to this goal is:

(4) It is warm on today. It will be warm on Monday. It will be cool on Tuesday.

Since the input goal simply involves transmitting a fixed set of information to the hearer, the construction of this response is fairly straightforward. Nevertheless, even in this simple example there are a few places where purely linguistic preferences are overridden by the planner in order to support the text plan structure.

4.1.1 Building the Initial Plan. Upon being invoked, IGEN sends its goal to the communicative planner, which constructs an initial plan. The planner starts by applying the **utter-sequence** plan, which involves decomposing the goal based on a relevant sequence. Since the members of #<WEATHER-INFO> all have connections to the sequence of days in the month, the planner is able to decompose the original goal into a sequence

20 The only possible constraint in the current implementation is a missing (obligatory) feature value; all other grammatical and conceptual constraints are strictly enforced.

21 This is actually controlled by a parameter (*conciseness-preference*) that determines whether the utterer will prefer concise or verbose options or neither. In theory, this parameter should be set by the planner based on its knowledge of the situation; in practice, it is set by hand to prefer concise options.

of goals based on the sequence of days involved in the goal. Further planning for the sequence of subgoals yields the following plan:

1. Overall Goal: #<KNOW(HEARER,WEATHER-INFO)>
2. Achieve (1) by sequencing information based on the #<SEQUENCE> of the three days involved
3. Support (2) with information about #<SEPT-25>
4. Support (2) with information about #<SEPT-26>
5. Support (2) with information about #<SEPT-27>

Requests to express the information about each day's weather are then passed on to the linguistic component.

In particular, the planner carries out step (4) of the plan by placing in the workspace the request:

```
#<OVER-TIME-SPAN(#<WITHIN-RANGE(TEMPERATURE,56DEG-75DEG-F)>,
                 SEPT-26)>
```

A separate workspace entry is created for each node and link in this request,[22] with the entry for the top-level #<OVER-TIME-SPAN> link marked as their "head" entry.

4.1.2 Producing and Evaluating Linguistic Options. The linguistic component then begins to produce and annotate linguistic options for each of these entries. It also runs the first phase of the evaluation algorithm (described in Figure 4) to produce an initial evaluation.[23]

For #<TEMPERATURE>, the linguistic component finds the following options:[24]

- #<w:WEATHER>

$$\begin{bmatrix} \text{CAT} & = & \text{NP} & \\ \text{HEAD} & = & \begin{bmatrix} \text{CAT} & = & \text{N} \\ \text{LEX} & = & \#\text{<w:WEATHER>} \end{bmatrix} \end{bmatrix}$$

  The linguistic component draws on the portion of IGEN's knowledge shown in Figure 5 to annotate this option. Since the option directly expresses the concept #<WEATHER>, the linguistic component attaches a MAKES-EXPLICIT annotation to the option. Since #<TEMPERATURE> has a #<PART> link connecting it to #<WEATHER>, it attaches an INDIRECTLY-SUGGESTS annotation for #<TEMPERATURE>.

22 This doesn't limit the linguistic component to working on only one node or link at a time; options can cover or depend on other entries, as, for example, the future tense option in Section 4.1.2.

23 This should really be done by the planner, but the first phase doesn't actually depend on any information about the plan, so it can safely be run by the linguistic component without compromising the separation of the components. The second phase, which does depend on the plan, will be run by the planner.

24 It actually produces several other options that are immediately ruled out as being completely inappropriate, e.g. the word *temperature* as an option for #<SEPT-26>, or because they require arguments that aren't present in the request, e.g. the verb *be* (which needs two arguments) as an option for #<TEMPERATURE>. Throughout these examples, such options will simply be ignored in the interests of brevity.

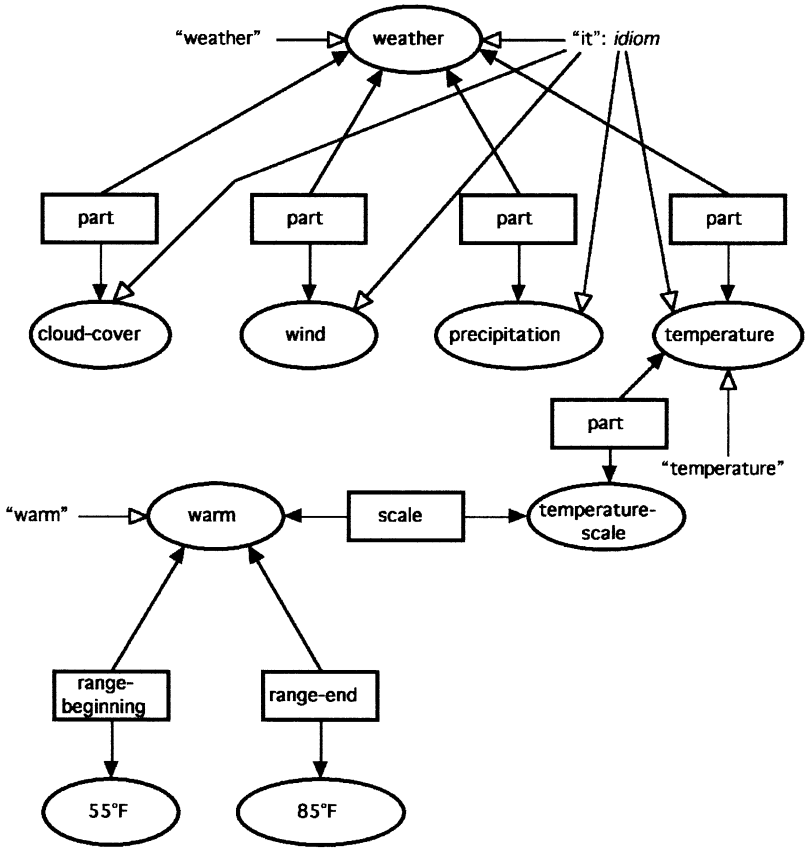

Figure 5
Some semantic network fragments used by the linguistic component.

- #<w:TEMPERATURE>

$$\begin{bmatrix} \text{CAT} & = & \text{NP} \\ \text{HEAD} & = & \begin{bmatrix} \text{CAT} & = & \text{N} \\ \text{LEX} & = & \text{\#<w:TEMPERATURE>} \end{bmatrix} \end{bmatrix}$$

  Since the meaning of #<w:TEMPERATURE> is the request itself, the only annotation attached is a MAKES-EXPLICIT annotation for #<TEMPERATURE>.

- #<w:IT>

$$\begin{bmatrix} \text{CAT} & = & \text{NP} \\ \text{LEX} & = & \text{\#<w:IT>} \end{bmatrix}$$

  Since the usage of #<w:IT> here is an idiomatic construction, it has some special handling in the lexicon to indicate that its meaning depends on what it is being used to describe. In particular, #<w:IT> here is taken to mean #<TEMPERATURE>. The linguistic component thus produces a MAKES-EXPLICIT annotation for #<TEMPERATURE>, as it does for the option using the word #<w:TEMPERATURE>. #<w:IT> is also marked as producing a CONCISE-CONSTRUCTION annotation, so the linguistic component attaches one to the option.

As these options are produced, they are evaluated based on their annotations according to the algorithm described in Figure 4. In the first phase, #<w:WEATHER> is rated MEDIUM, reflecting the fact that the option doesn't really express the request, but does suggest it indirectly by mentioning a closely related concept. #<w:TEMPERATURE> is rated VERY-HIGH, reflecting the option's exact match with the request, as is #<w:IT>, which also expresses #<TEMPERATURE> (albeit idiomatically). The second phase of the evaluation algorithm makes no changes to the ratings, since the options' annotations don't indicate anything that would interact with the plan, leaving #<w:IT> and #<w:TEMPERATURE> as the preferred options.[25]

For #<WITHIN-RANGE>, the only acceptable option the linguistic component finds is:

- #<w:BE>

$$\begin{bmatrix} \text{CAT} & = & \text{S} \\ \text{HEAD} & = & \begin{bmatrix} \text{CAT} & = & \text{V} \\ \text{LEX} & = & \text{\#<w:BE>} \end{bmatrix} \\ \text{SUBJ} & = & \text{NP} \\ \text{PRED} & = & \text{ADJP} \\ \text{ORDER} & = & (\text{SUBJ HEAD PRED}) \end{bmatrix}$$

The SUBJ and PRED features indicate #<w:BE>'s syntactic roles; the value of these features indicates the syntactic constraints on the structures that may fill the roles. In addition, the structure has a MAP feature indicating which workspace entries can provide options to fill the roles.

Since #<WITHIN-RANGE> is a subconcept of #<BE-LOCATED> (one meaning of #<w:BE> in IGEN's lexicon), the linguistic component produces (MAKES-EXPLICIT #<BE-LOCATED>) and (MAKES-IMPLICIT #<WITHIN-RANGE>) annotations for the option. These annotations lead the first phase of the evaluation algorithm to rate the option as HIGH; the second phase leaves this evaluation unadjusted. (Of course, since there is only one acceptable option here, its rating doesn't really matter.)

For #<56DEG-75DEG-F>, the options produced are (omitting the details of the structures to show only the annotations):

- #<w:WARM>:
  (MAKES-EXPLICIT #<WARM>)
  (MAKES-IMPLICIT #<56DEG-75DEG-F>)
- #<w:TEMPERATURE>:
  (MAKES-EXPLICIT #<TEMPERATURE>)
  (INDIRECTLY-SUGGESTS #<56DEG-75DEG-F>)

These annotations are based on the network fragment shown in Figure 5. #<w:WARM> is rated HIGH, because it does a reasonable job of describing the requested temperature range.[26] The #<w:TEMPERATURE> option is not as good, but it at least gives a sense of what is being talked about, so it gets the next lower rating of MEDIUM.

The entries we have seen so far have not involved any interactions between the

25 The utterer will ultimately choose #<w:IT> because it is marked as more concise.
26 This is determined using a rule-based inference mechanism described in Rubinoff (1992).

options and the plan. Our first instance of such interactions arises in the options for #<OVER-TIME-SPAN>:

- #<w:OVER>:
  (MAKES-EXPLICIT #<OVER-TIME-SPAN>)
  (ACTIVATES-IN-CONTEXT
  #<INSTANCE(SEPT-26,LINEAR-SPAN)>)
- #<w:ON>:
  (MAKES-EXPLICIT #<ON>)
  (MAKES-IMPLICIT #<OVER-TIME-SPAN>)
  (ACTIVATES-IN-CONTEXT
  #<INSTANCE(SEPT-26,LINEAR-POS)>)
- [TENSE = FUTURE]: [27]
  (MAKES-EXPLICIT #<OVER-TIME-SPAN>)
  (MAKES-EXPLICIT #<FUTURE>)
  (COVERS-OTHER-ENTRY entry for:#<SEPT-26>)
  (MAKES-IMPLICIT #<SEPT-26>)

The first phase evaluations suggest that #<w:OVER> is the preferred option. Here, though, the second phase of the evaluation algorithm modifies the ratings based on interactions with the plan. The ACTIVATES-IN-CONTEXT annotations for #<w:OVER> and #<w:ON> indicate that these options affect how the system (and the hearer) perceive the time span being talked about. Since the #<SEQUENCE> of days being talked about is the organizing principle of the plan, it is a "critical" piece of the plan, as are the #<MEMBER> links connecting it to the pieces of the original goal. The ACTIVATES-IN-CONTEXT annotation for #<w:OVER> indicates that it encourages the hearer to think of #<SEPT-26> as a #<LINEAR-SPAN>. This conflicts with the (critical) #<MEMBER> link between #<SEPT-26> and the #<SEQUENCE>, because it describes the day as a span of time rather than as a discrete point in a sequence. The second phase algorithm thus downgrades #<w:OVER>'s rating from VERY-HIGH to HIGH. Conversely, #<w:ON> encourages the hearer to think of #<SEPT-26> as a #<LINEAR-POS>, i.e., a discrete position within a linear span, supporting the critical #<MEMBER> link. The rating for #<w:ON> is therefore upgraded from HIGH to VERY-HIGH, and it becomes the preferred option.[28]

Note here how the use of annotations allows the planner to remain ignorant of purely linguistic issues. The planner has no way to distinguish between an option realizing #<OVER-TIME-SPAN> as a lexical item and one realizing it as the value of some feature of another option. Its decision is based solely on the contextual effects of the available options. The planner can only make distinctions based on the *effects* of the options, not on the *forms* of the options.

The final piece of the request is #<SEPT-26>, for which the linguistic component produces the following options:

- #<w:TOMORROW>:
  (MAKES-EXPLICIT #<TOMORROW>)

27 This is an example of how linguistic options need not be single words or complete phrases; this option is a feature value pair that will be added to another phrase.

28 #<w:ON> is preferred over [TENSE = FUTURE], which is also rated VERY-HIGH, because it was adjusted up in response to a specific plan interaction as discussed in Section 3.3.

(MAKES-IMPLICIT #<SEPT-26>)
(ACTIVATES-IN-CONTEXT #<FEW-DAYS>)

- #<w:MONDAY>:
(MAKES-EXPLICIT #<MONDAY>)
(MAKES-IMPLICIT #<SEPT-26>)
(ACTIVATES-IN-CONTEXT #<WEEK>)

The first phase of the evaluation algorithm rates both of these options as HIGH, since they match the planner's request equally well. The second phase, though, distinguishes between them on the basis of the particular time sequences they activate. The sequence #<WEEK> activated by the option #<w:MONDAY> is significantly different from the #<SEQUENCE> of three days around which the plan is organized, so #<w:MONDAY> is downgraded to #<MEDIUM>. The #<FEW-DAYS> sequence activated by the option #<w:TOMORROW>, on the other hand, largely overlaps with the plan's #<SEQUENCE>, so no adjustment is made. As a result, #<w:TOMORROW> becomes the preferred option.

4.1.3 Assembling the Utterance. As these options are produced and evaluated, the utterer watches the workspace, trying to assemble them into an acceptable utterance. Whenever an option appears that meets the current acceptability threshold, the utterer utters (i.e., prints) it. The final result is:

(5) It will be warm on Monday.

Note that this output does not simply use the highest-rated option from each entry in the workspace, because that would involve the ungrammatical phrase *on tomorrow*. In addition to responding to time pressure, the utterer also enforces syntactic constraints; it chooses the best possible combination of options that is syntactically acceptable.[29]

4.1.4 Summary. This simple example illustrates how the annotations allow IGEN to maintain the traditional division into two independent components while still handling interactions between decisions at the planning and linguistic levels. The planner never has to deal with linguistic structures, and the linguistic component never has to deal with plans or information structures. For example, the planner can't distinguish between lexical items and feature values; conversely the linguistic component has no access to the goals driving the planner or the plans it builds. Nevertheless, when there are decisions that depend on both informational and linguistic structures, the annotations allow IGEN to handle the interactions between the different levels. The effects of these interactions are relatively minor in this example, but subsequent examples will show how they can be more dramatic.

4.2 Revising the Plan

In addition to merely selecting among options, IGEN's planner can revise the plan in response to them. Consider a slight modification to the previous example, in which IGEN is given the same goal except that the temperature on the third day falls within

29 This is an oversimplification, as can be seen from the output in (4) in Section 4.1, where IGEN produces the phrase *on today*. What's really going on is that IGEN's lexicon has two entries for each of *today* and *tomorrow*; one as a noun phrase and one as a prepositional phrase. As it happens, the PP entry turns up first. In the case here, the option for *Monday* is produced and selected before the NP option for *tomorrow* turns up. In the first sentence of (4), there is no such option, so the linguistic component has time to produce the NP *today* option.

the range considered "warm." IGEN proceeds to build a plan that is identical to the previous one (except for the difference in the information to be expressed). As before, the planner places requests in the workspace and the linguistic component starts to construct options. At this point, IGEN is well on its way to generating text similar to (4):

(6) It is warm on today. It will be warm on Monday. It will be warm on Tuesday.

However, once the option for *warm* is produced, something new happens. The planner detects that there are several parallel structures in the workspace that can use the same linguistic structure (*warm*) to realize different concepts (the different temperature ranges) in corresponding positions within each structure. This provides the opportunity to combine the parallel structures into a single conjoined structure, which the planner proceeds to do. All of the workspace entries for the parallel structures are removed, and the planner enters requests for a new conjoined structure:[30]

```
#<OVER-TIME-SPAN(#<WITHIN-RANGE(TEMPERATURE,56DEG-75DEG-F)>,
                 #<AND(SEPT-25,SEPT-26,SEPT-27)>)>
```

IGEN then proceeds as in the previous example, ultimately generating:

(7) It will be warm on today, Monday, and Tuesday.

Note that the planner couldn't have created this conjoined structure initially because it didn't know that all three days' temperatures could be realized the same way. To the planner, each day has a different temperature range. This example appears exactly the same as the previous one. Once the linguistic component indicates that all three temperature ranges can be realized the same way, though, the planner can then determine that this results in a parallelism that can be reduced to a conjoined structure. The annotations allow the planner to modify its plan, based on the results of the realization process, despite not having any access to purely linguistic knowledge or processing.

4.3 Varying the Utterance in Response to the Plan

In the previous examples, we have seen how IGEN can use the feedback from the linguistic component to improve the organization of the text it generates. The annotations also allow IGEN to tailor the generated text so as to better achieve the goals behind the utterance. Consider IGEN's response to the goal "make the user happy":[31]

1. Overall goal: `#<BE-STATE(HEARER,HAPPY)>`
2. Achieve (1) by achieving subgoals:
   (a) Goal: Downplay unpleasant information

30 The temperature range used in the conjoined structure is chosen randomly from among the structures being combined. This is safe to do because it is certain that all of the temperature ranges can be realized the same way; that's what triggered the combination in the first place.

31 The construction of the plans in this section is admittedly ad hoc; IGEN certainly doesn't have a complete model of the psychological factors and issues they involve. What's important here, though, is not the construction of the plan but rather how the plan and the annotations are used to produce the kind of subtle effects that can't be produced by the linguistic component or the planner alone.

(b) Goal: Emphasize pleasant information

3. Achieve (2a) by expressing:
   #<MINIMAL-SIGNIFICANCE(#<CONSIST-OF(PRECIPITATION,
   RAIN)>)>
4. Achieve (2b) by expressing:
   #<WITHIN-RANGE(TEMPERATURE,60DEG-80DEG-F)>
5. Achieve (2b) by expressing:
   #<AT-TIME(#<CONSIST-OF(PRECIPITATION,NO-PRECIPITATION)>,
   THIS-AFTERNOON)>

The final result for this plan is:

(8) It is only drizzling. It is warm. It will be clear soon.

Most of this output is produced straightforwardly in a manner similar to the previous example. Some of the choices, however, are the result of interactions of a sort beyond what was seen there.

Consider the options for #<CONSIST-OF(PRECIPITATION,RAIN)> in step (3) of the plan; these include:[32]

- #<w:RAINS>: VERY-HIGH
  (MAKES-EXPLICIT #<CONSIST-OF>)
  (MAKES-EXPLICIT #<PRECIPITATION>)
  (MAKES-EXPLICIT #<RAIN>)
- #<w:DRIZZLES>: HIGH
  (MAKES-EXPLICIT #<CONSIST-OF>)
  (MAKES-EXPLICIT #<PRECIPITATION>)
  (MAKES-IMPLICIT #<RAIN>)
  (MAKES-EXPLICIT #<DRIZZLE>)
  (EXTRA-INFO #<STRENGTH(RAIN,WEAK)>)
- #<w:POURS>: HIGH
  (MAKES-EXPLICIT #<CONSIST-OF>)
  (MAKES-EXPLICIT #<PRECIPITATION>)
  (MAKES-IMPLICIT #<RAIN>)
  (MAKES-EXPLICIT #<POUR>)
  (EXTRA-INFO #<STRENGTH(RAIN,STRONG)>)

The first phase of the evaluation algorithm rates the word #<w:RAINS> highest, since it captures the intended meaning precisely. #<w:DRIZZLES> and #<w:POURS> are the next best options, since they capture the basic meaning, and are equally appropriate, since the linguistic component can't evaluate the significance of the extra information they present.

The second phase, however, determines that the EXTRA-INFO expressed by the option #<w:DRIZZLES> implies another part of the information requested by the planner (the #<MINIMAL-SIGNIFICANCE> relation). Its rating is therefore increased to VERY-HIGH. Conversely, the rating for #<w:POURS> is lowered to MEDIUM, since it

32 To simplify the discussion, I have included the first-phase evaluations of the options here.

contradicts the #<MINIMAL-SIGNIFICANCE> relation. This leaves both other options, #<w:RAINS> and #<w:DRIZZLES>, rated VERY-HIGH; the utterer will prefer to use #<w:DRIZZLES>, though, since there was something specific about the option that the planner approved of.

The decision here to say #<w:DRIZZLES> instead of #<w:RAINS> could not have been made by either the planner of the linguistic component alone. The decision depends on both the planner's knowledge that the concept #<DRIZZLE> supports another piece of the plan and the linguistic component's knowledge that there is a way to express #<DRIZZLE> without making explicit its distortion of the actual information that it's raining. It's only the interaction between the two components provided by the annotations that makes saying #<w:DRIZZLES> instead of #<w:RAINS> possible.

The point of using the annotations, though, was not just to handle these kinds of interactions, but more importantly to do so without compromising the principled separation of the two components. Among other things, this makes it possible for IGEN to express the same information in a different manner, based solely on differences in the plan. If the purely linguistic issues are the same, the linguistic component will do exactly the same work, even though the differences in the output may depend in part on that work.

We can see this by comparing the previous example with IGEN's response to the goal #<TAKE(HEARER,UMBRELLA)>, for which the planner constructs the following plan:

1. Goal: #<TAKE(HEARER,UMBRELLA)>
2. Achieve (1) by achieving subgoal:
   #<WANT(HEARER,#<TAKE(HEARER,UMBRELLA)>)>
3. Achieve (2) by achieving subgoal:
   #<KNOW(HEARER,#<CONSIST-OF(PRECIPITATION,RAIN)>)>
   depending on the preconditions:
   #<DANGEROUS(#<CONSIST-OF(PRECIPITATION,RAIN)>)>
   and
   #<PROTECTION(#<TAKE(HEARER,UMBRELLA)>,
   #<CONSIST-OF(PRECIPITATION,RAIN)>)>
4. Achieve (3) by expressing:
   #<CONSIST-OF(PRECIPITATION,RAIN)>

Note that the information to be expressed here in step (4) is also expressed in step (3) of the previous example. Since the linguistic component doesn't have access to the plan, it necessarily produces the same options as before. The planner, though, evaluates these options differently, since it is using them to carry out a different plan.

The three options #<w:RAINS>, #<w:POURS>, and #<w:DRIZZLES> are produced and annotated as before. Their evaluations are different, though. Step (3) here has a precondition that #<CONSIST-OF(PRECIPITATION,RAIN)> be #<DANGEROUS> in order to achieve its intended goal. The extra information expressed by #<w:POURS> supports this precondition, so #<w:POURS> is upgraded to VERY-HIGH. On the other hand, #<w:DRIZZLES> undercuts the precondition and is therefore downgraded to MEDIUM. The preferred option here is therefore #<w:POURS>, in contrast to the previous example, and the final result is then:

(9) It is pouring.

The final output differs even though the linguistic component does exactly the same processing in these two examples. From the linguistic component's point of view, the examples are exactly the same; the same request is placed in the workspace, so the same options are produced. The difference lies solely in the plan structure, not in the content of the explicit message to be realized, so only the planner behaves differently. The linguistic component's design and functioning are completely indifferent to the issues that concern the planner.

4.4 The Range of Variation

We can see similar variation in the handling of #<NO-PRECIPITATION>, which is realized in three different ways in three different examples because of the different roles it plays in the three plans. In Section 4.3, it was realized as *clear*; the two examples that follow illustrate how IGEN can select alternative realizations.

Consider first what happens when the generator is given the goal "get the user to conserve water." In response, it builds up a plan to do so by talking about the drought:

1. Goal: #<CONSERVE(HEARER,WATER)>
2. Achieve (1) by achieving subgoal:
   #<WANT(HEARER,#<CONSERVE(HEARER,WATER)>)>
3. Achieve (2) by achieving subgoal:
   #<KNOW(HEARER,#<SERIOUS-DROUGHT>)>
   depending on the preconditions:
   #<DANGEROUS(#<SERIOUS-DROUGHT>)> and
   #<PROTECTION(#<CONSERVE(HEARER,WATER)>,
   #<SERIOUS-DROUGHT>)>
4. Achieve (3) by expressing:
   #<CONTINUE(DROUGHT)>
   #<OVER-TIME-SPAN(CONSIST-OF(PRECIPITATION,
   NO-PRECIPITATION)),
   TODAY)>
   and
   #<OVER-TIME-SPAN(REPEAT(CONSIST-OF(PRECIPITATION,
   NO-PRECIPITATION)),
   TOMORROW)>

This plan is similar to the one built in Section 4.3 for the goal #<TAKE(HEARER,UMBRELLA)>; they both involve motivating the hearer to take some action by explaining some danger that the action will provide protection from. The main difference here is that rather than simply telling the user that there is a serious drought, the planner draws on a more specific (ad hoc) plan that involves expressing more detailed information.

The final result for this example is:

(10) The drought is continuing. It is dry today. It will be dry again tomorrow.

Note that IGEN uses #<w:DRY> here to express #<NO-PRECIPITATION>, in contrast with the first example in Section 4.3, where it was expressed by #<w:CLEAR>. This happens because #<w:DRY> has an (ACTIVATES-IN-CONTEXT #<DROUGHT>) annotation, and activating the concept #<DROUGHT> supports the sense of danger that is a precondition of step (3) of the plan. #<w:DRY>'s evaluation is thus upgraded, making

it the preferred option. In the example in Section 4.3, this activation doesn't interact with the plan, so #<w:CLEAR> remains the preferred option.

Yet another realization is selected when IGEN is given the goal of convincing the user to go to the park, resulting in the following plan:

```
1.  Goal: #<BE-LOCATED(HEARER,PARK)>
2.  Achieve (1) by achieving subgoal:
    #<WANT(HEARER,#<BE-LOCATED(HEARER,PARK)>)>
3.  Achieve (2) by achieving subgoal:
    #<KNOW(HEARER,
            AND(CONSIST-OF(PRECIPITATION,NO-PRECIPITATION))
               WITHIN-RANGE(TEMPERATURE,WARM))>
    depending on the preconditions:
    #<ENJOYABLE(#<BE-LOCATED(HEARER,PARK)>)>
            and
    #<SUITABLE(
           #<AND(CONSIST-OF(PRECIPITATION,NO-PRECIPITATION),
                 WITHIN-RANGE(TEMPERATURE,WARM))>,
           #<BE-LOCATED(HEARER,PARK)>)>
4.  Achieve (3) by expressing:
    #<AND(CONSIST-OF(PRECIPITATION,NO-PRECIPITATION),
           WITHIN-RANGE(TEMPERATURE,WARM))>
```

The resulting output is:

(11) It is sunny and it is warm.

The only difference from previous examples lies in the choice of #<w:SUNNY> to express the concept #<NO-PRECIPITATION> rather than using either #<w:CLEAR> or #<w:DRY>. This choice arises because of an ACTIVATES-IN-CONTEXT annotation indicating that the use of #<w:SUNNY> activates the concept #<SUNSHINE>, which strengthens the enjoyability precondition in step (3) of the plan. It is this sort of interaction between linguistic choices and the plan structure that allows IGEN to express #<NO-PRECIPITATION> three different ways in three different examples.

The point here is not just that IGEN can produce different lexical realizations for a particular concept. If that were the only goal, we could dispense with the feedback mechanism and simply design some sort of discrimination network (or similar device) to test various features of the information being expressed. The planner could supply whatever information is needed to drive the network.[33] Something like this approach is in fact used in some systems (e.g., Elhadad and Robin 1992; PenMan 1989; Hovy 1988a).

The problem with the discrimination network approach is that it can't handle the range of examples shown here without violating the modularity of the generator. The examples shown here have involved linguistic choices that depend on the plan structure, on other actions in the plan, and on preconditions of the plan. Furthermore, the plan dependencies can involve a nontrivial amount of reasoning (e.g., the example immediately above, which involves the connection between sunshine and enjoying

33 Of course, that still wouldn't allow for examples like the one in Section 4.2, where the planner modifies the plan in response to the work of the linguistic component.

being outdoors). Encoding these decisions in a discrimination net (or any other kind of processing) within the linguistic component would mean giving that component access to all of the plan and the reasoning used to construct it. Much of the planner's knowledge and processing would have to be duplicated in the linguistic component. IGEN, in contrast, handles these examples while maintaining strict modularity.

4.5 The Effects of Time Pressure

The discussion of the examples in Section 4.3 glosses over an important detail: IGEN will only generate the output shown there if the time pressure is eased by increasing the delay thresholds. In particular, the choice of #<w:DRIZZLES> and #<w:POURS> over #<w:RAINS> in those examples is made with the thresholds set to force the utterer to always wait at least five time units before uttering the next piece of text. This is necessary because the linguistic component produces the option for #<w:RAINS> sooner than the options for #<w:POURS> and #<w:DRIZZLES>.

If the delay thresholds are left at the default settings, the utterer will accept an option that is rated VERY-HIGH without any delay. Thus when the #<w:RAINS> option appears, the utterer immediately accepts and utters it. While #<w:DRIZZLES> and #<w:POURS> would be better options in these examples, the utterer never gets a chance to see them. Because of time pressure, the decision of how to realize the planner's request is made before the linguistic component has a chance to find #<w:DRIZZLES>. There's nothing wrong with this result; *it is raining* is a perfectly good way to express #<CONSIST-OF(PRECIPITATION,RAIN)>. It's just that if IGEN is given more time to work, it can produce a better utterance. The utterer's job is to balance the quality of the utterance against time pressure; in this example, changing the intensity of the time pressure induces a corresponding change in the refinement of the utterance.

Of course, if the time pressure is set sufficiently high (e.g., to simulate a situation where the the hearer is about to walk away), IGEN will sometimes produce an utterance that has some slight problems, because speed is more important than perfection. Consider what happens when the example in Section 4.1 is run with the delay thresholds set to accept anything that's not terrible (i.e., any option rated MEDIUM or higher) right away. The planning and initial production and evaluation of options proceeds as with the default thresholds. The handling of #<OVER-TIME-SPAN> works out differently, though. The option for #<w:OVER> is produced by the linguistic component several time units before the option for #<w:ON>. With the default thresholds, that's not a problem, because the utterer can wait long enough for #<w:ON> to be produced. With the increased time pressure, though, the utterer is forced to produce the utterance before the option #<w:ON> is available. The resulting utterance (for the second day's information) is *it will be warm over Monday*.

This result is not awful; the sentence still manages to express the intended information. It just doesn't fit quite as well into the intended overall structure of the text. So the main consequence here of increasing the time pressure is to generate a text that is slightly less coherent and fluent. In this case, IGEN responds to increasing time pressure with a slight degradation in the generated text.

4.6 The Effects of Vocabulary Limitations

Since IGEN is designed to always find the best utterance possible in the available time, without assuming any notion of a "correct" utterance, it can produce reasonable output even when generating in domains where it has limited linguistic resources. This can be seen by looking at the result of selectively disabling elements of its "vocabulary," i.e., the lexical and syntactic resources available to the linguistic component. The result

is a gradual, graceful degradation of the generated text as the generator's linguistic resources become poorer.

Consider again IGEN's response (under relatively mild time pressure) when given the goal #<TAKE(HEARER,UMBRELLA)>. The resulting plan, as we have seen, requires expressing the information #<CONSIST-OF(PRECIPITATION,RAIN)>, which is accomplished by uttering *it is pouring*. If the verb #<w:POUR> is removed from the generator's vocabulary, though, this utterance is not possible; instead the output becomes *it is raining*. Note that this is the same output IGEN produces if the time pressure is increased; the consequence of limiting either time or vocabulary is the same.

If the verbs #<w:RAINS> and #<w:DRIZZLES> are removed as well, the generator is left with no single option that can express all of the requested information. Instead, the utterer resorts to assembling individual options for the various pieces of the request, producing the sentence *the precipitation consists of rain*.[34] While this utterance does adequately express the requested information, it is fairly awkward. Still, it's not bad, given that the two best options are unavailable.

Beyond this point, further vocabulary removal starts to make things much worse. Removing the phrase #<w:CONSIST OF> leads to the utterance of *it will be clear*. This utterance is terrible; it completely misses the intended information. That's not surprising; after all, the options that should be used have all been removed from IGEN's vocabulary. On the other hand, it's not completely bizarre; it's still talking about the weather, and in fact it's even still describing the precipitation. That is, IGEN still has a vague sense of what it's trying to say; it's not simply producing an utterance at random. This can be seen again by removing the phrase #<w:BE CLEAR>; the resulting utterance is then *it snowed*. IGEN is still trying to get as close as it can to the intended information, even though it can't really get it right.

Eventually, of course, removing linguistic resources will make the generator collapse; if every word or phrase relating to precipitation of any sort is removed, then IGEN produces *it is medium*, which, though grammatical, is gibberish. If the linguistic component has absolutely no way to produce options related to what the planner wants to express, then there's really nothing IGEN can do. What's significant here, though, is that as the generator's linguistic resources are gradually impoverished, the generator's output degrades gracefully. Rather than immediately collapse when its preferred options are removed, IGEN continues to produce the best possible utterance it can build with its remaining linguistic resources.[35]

4.7 Generating in a Different Language

As a test of IGEN's modularity, a simple French linguistic component was developed and tested on the examples in Sections 4.1 and 4.3. Switching to a different language provides an extreme example of how the linguistic component can be varied without affecting the planner. The change primarily involved defining new lexicon entries for the French words, phrases, and feature values needed to talk about the weather. In addition, the routine that handles verb inflection had to be replaced, since French verb endings are different from English ones. Other than these changes, though, no mod-

34 The actual output is *precipitation consists of rain*, but this is just due to a limitation of the implemented grammar. Specifically, there's no requirement that noun phrases have determiners, so the linguistic component never bothers to suggest one for the NP headed by #<w:PRECIPITATION>.

35 Of course, in many situations it's best to put a lower bound on the quality of IGEN's output; this can be done simply by setting the appropriate delay thresholds to infinity. It would also be possible to set an overall time limit on IGEN's processing, after which an uncompleted plan would be assumed to have failed. IGEN could then modify the plan or take other appropriate corrective action.

ification to IGEN was necessary. The linguistic component was otherwise unaltered, and the planner and utterer required no changes at all.[36]

The main difference between the ways English and French describe the weather lies in the "weather *it*" construction. English allows weather descriptions that use *it* together with any of a class of verbs of being, seeming, or becoming. Thus the following sentences are all possible in English:

(12) It is warm.

(13) It seems warm.

(14) It feels warm.

(15) It's becoming warm.

(16) It's getting warm.

In these sentences, the meaning of *it* seems to be the weather or the temperature; thus in any of these sentences *it* could be replaced by *the weather* or *the temperature* without changing the meaning or acceptability. This phenomenon is modeled in IGEN by having a special word #<w:IT> whose meaning can range over the various items constituting the weather; this word is constrained to appear only as the subject of an appropriate verb. Thus, as we saw in Section 4.1, #<w:IT> is represented as meaning #<TEMPERATURE> and annotated appropriately.

A literal translation of any of the sentences in (12) to (16) into French, though, would be ungrammatical. The corresponding French construction uses the verb *faire* (literally 'to make' or 'to do'), as in (17):

(17) Il fait chaud.
It makes/does warm
It is warm.

(18) *Le temps fait chaud.
The weather makes/does warm
The weather is warm.

(19) *La température fait chaud.
The temperature makes/does warm
The temperature is warm.

(20) *Il semble chaud.
It seems warm.

(21) Il semble faire chaud.
It seems to make/do warm.
It seems warm.

Unlike *it* in (12) to (16), the *il* here is not referring to the weather or the temperature. Rather it is an expletive, i.e., a dummy subject required by the grammar of the language, similar to the *it* in the English *it seems John left*. This can be seen by comparison

36 It was necessary to add a few more elements to the semantic network (which is used by both components) to capture the meaning of the French word *sur*, which does not correspond exactly to the meaning of any English word, as discussed below. These new elements were used only by the linguistic component, not by the planner.

with (18) and (19), in which replacing *il* with *le temps* ("the weather") or *la température* ("the temperature") leads to an ungrammatical sentence. (20) and (21) demonstrate that the crucial element in the French construction is the verb *faire*; other verbs can be used only if they combine with *faire*. This contrasts with the English construction, in which *it* is the crucial element and can be used with a range of verbs. The French construction is thus represented in IGEN as a version of the verb *faire* that takes an expletive *il* as its subject and whose meaning ranges over specifications of the various elements of the weather.

A second difference between the two languages shows up in Section 4.1 in the options for #<OVER-TIME-SPAN>. In English, IGEN has to choose between using *on* or *over* to express this link. In French, though, these concepts are expressed by the same word (*sur*).[37] Furthermore, *sur* can't be used for temporal expressions, only for spatial expressions. So the options for #<OVER-TIME-SPAN> work out quite differently in French. French in general uses *à* ("at") rather than *sur* for temporal expressions; however *à* cannot be used with expressions describing days. Instead, these temporal modifiers are expressed using bare NP's with no preposition. This construction is modeled in IGEN by a null preposition whose object must be a definite NP.

Given all of this, IGEN produces the following results in French for the example in Section 4.1:

(22) Il fait chaud aujourd'hui. Il fera chaud lundi.
it does warm today it will do warm Monday
It is warm today. It will be warm Monday.

Il fera frais mardi.
it will do cool Tuesday
It will be cool Tuesday.

This is equivalent to the English result except for expressing the temporal modifier as a noun phrase rather than as a prepositional phrase headed by *on*, as discussed above.

Similarly, the variant in Section 4.2 produces:

(23) Il fera chaud aujourd'hui lundi et mardi.
it will do warm today Monday and Tuesday
It will be warm today Monday and Tuesday.

For the goal of making the user happy, IGEN produces the French output:

(24) Il bruine seulement. Il fait chaud. Le temps sera clair bientôt.
it drizzles only it does warm the weather will be clear soon
It's only drizzling. It's warm. It will be clear soon.

This is exactly the same as the English output except for the use of the French "weather *faire*" construction or the explicit use of *le temps* ("the weather") rather than the English "weather *it*" construction.

37 Note that the point here is not that *sur* is the (unique) *translation* of *on* and *over* into French; IGEN is doing generation, not machine translation. It's just that *sur* is the French word that most naturally expresses the particular concepts IGEN is using.

The goal of convincing the user to take his umbrella, though, produces the output:

(25) Il pleut.
it rains
It's raining.

because French doesn't have a verb corresponding to *to pour*. Since no option equivalent to the one for #<w:POURS> is proposed, the option for #<w:PLEUT> is the preferred option, just as #<w:RAINS> is the preferred option in English when lack of time or deliberate removal prevents the linguistic component from finding #<w:POURS>.

The point here is not simply that IGEN can handle these changes; after all, one of the primary motivations for dividing the generator into separate planning and linguistic components is to allow the components to be modified independently. The point is that IGEN allows the linguistic component to be modified so dramatically without limiting its ability to handle interactions between linguistic and planning issues. In fact, IGEN can handle these interactions equally with either version of the linguistic component without having to make any corresponding changes to the planner.

5. Summary

IGEN is designed to overcome the limitations, while retaining the advantages, of the modular approach to natural language generation. It does this by means of annotations that provide the planner with an abstract description of the effects of particular linguistic choices, allowing IGEN to handle interactions between the planning and linguistic levels while retaining the complete independence of the components. Thus IGEN can vary the work done by each component independently, even in cases where the final output depends on interactions between them.

As the examples in Section 4 show, IGEN can vary how it expresses information in response to the differing roles that the information plays in the plan and, conversely, in response to a change in the language being used. It can also revise its initial communicative plan based on the options suggested by the linguistic component. Furthermore, this variation requires no weakening of the generator's modularity. Changes in the plan structure are invisible to the linguistic component, and the change in languages is invisible to the planner. In addition, since IGEN explicitly models and reasons about the effects of its linguistic choices, it can gracefully handle situations where the available time or linguistic resources are limited.

References

Appelt, Douglas E. 1983. TELEGRAM: A grammar formalism for language planning. In *Proceedings of the 21st Annual Meeting*, pages 74–78, Cambridge, MA, June. Association for Computational Linguistics.

Appelt, Douglas E. 1985. *Planning English Sentences*. Studies in Natural Language Processing. Cambridge University Press.

Dale, Robert. 1989. *Generating Referring Expressions in a Domain of Objects and Processes*. Ph.D. thesis, Centre for Cognitive Science, University of Edinburgh.

Danlos, Laurence. 1987. *The Linguistic Basis of Text Generation*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, England, English translation edition. Translated by Dominique Debize and Colin Henderson.

Delin, Judy, Anthony Hartley, Cécile Paris, Donia Scott, and Keith Vander Linden. 1994. Expressing procedural relationships in multilingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 61–70, Kennebunkport, ME, June.

Elhadad, Michael and Jacques Robin. 1992. Controlling content realization with functional unification grammars. In

Robert Dale, Eduard Hovy, Dietmar Rösner, and Oliviero Stock, editors, *Aspects of Automated Natural Language Generation*. Springer-Verlag, pages 89–104.

Fisher, Steven K. and Kathleen R. McKeown. 1990. Coordinating text and graphics in explanation generation. In *Proceedings of the Eigth National Conference on Artificial Intelligence*, pages 442–449, Boston, MA. American Association for Artificial Intelligence.

Grosz, Barbara J. and Candace L. Sidner. 1985. Discourse structure and the proper treatment of interruptions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 832–839, Los Angeles, CA, August. Morgan Kaufmann.

Grosz, Barbara J. and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Hovy, Eduard H. 1988a. *Generating Natural Language Under Pragmatic Constraints*. Lawrence Erlbaum, Hillsdale, NJ.

Hovy, Eduard H. 1988b. Planning coherent multisentential text. In *Proceedings of the 26th Annual Meeting*, pages 163–169, Buffalo, NY, June. Association for Computational Linguistics.

Hovy, Eduard H. 1988c. Two types of planning in language generation. In *Proceedings of the 26th Annual Meeting*, pages 179–186, Buffalo, NY, June. Association for Computational Linguistics.

Kantrowitz, Mark and Joseph Bates. 1992. Integrated natural language generation systems. In Robert Dale, Eduard Hovy, Dietmar Rösner, and Oliviero Stock, editors, *Aspects of Automated Natural Language Generation*. Springer-Verlag.

Mann, William C. 1983. An overview of the nigel text generation grammar. In *Proceedings of the 21st Annual Meeting*, pages 79–84. Association for Computational Linguistics.

Mann, William C. and Sandra A. Thompson. 1987. Rhetorical structure theory: Description and construction of text structures. In Gerard Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*. Martinus Nijhoff Publishers, Boston, MA, chapter 7, pages 85–96.

McDonald, David D. 1983. Natural language generation as a computational problem. In M. Brady and Robert Berwick, editors, *Computational Models of Discourse*. MIT Press, pages 209–265.

McDonald, David D. 1988. Modularity in langauge generation: Methodological issues. In *Proceedings of the AAAI-88 Workshop on Text Planning and Generation*, St. Paul, MN, August.

McDonald, David D. 1991. On the place of words in the generation process. In Cécile Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, Boston, MA, chapter 9, pages 229–247.

McKeown, Kathleen R. 1985. *TEXT GENERATION: Using Discourse Strategies and Focus Constraints to Generate Natural Language*. Cambridge University Press.

Meteer, Marie W. 1989. *The "Generation Gap": The Problem of Expressibility in Text Planning*. Ph.D. thesis, University of Massachusetts at Amherst, Amherst, MA, December.

Meteer, Marie W. 1994. Generating event descriptions with sage: A simulation and generation environment. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 99–107, Kennebunkport, ME, June.

Meteer, Marie W., David D. McDonald, Scott D. Anderson, David Forster, Linda S. Gay, Alison K. Huettner, and Penelope Sibun. 1987. Mumble-86: Design and implementation. Technical Report 87-87, Computer and Information Science, University of Massachusetts at Amherst, Amherst, MA.

Moore, Johanna D. and Cécile Paris. 1989. Planning text for advisory dialogues. In *Proceedings of the 27th Annual Meeting*, pages 203–211, Vancouver, BC, June. Association for Computational Linguistics.

Nii, H. Penny. 1986a. Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective. *AI Magazine*, 7(3):82–106.

Nii, H. Penny. 1986b. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53.

Nilsson, Nils J. 1980. *Principles of Artificial Intelligence*. Tioga Publishing Co., Palo Alto, CA.

Nirenburg, Sergei, Rita McCardell, Eric Nyberg, Philip Werner, Scott Huffman, Edward Kenschaft, and Irene Nirenburg. 1988. Diogenes-88. Technical Report CMU-CMT-88-107, Center for Machine Translation at Carnegie-Mellon University, Pittsburgh, PA, June.

Nogier, Jean-François. 1989. A natural language production system based on conceptual graphs. Technical Report F.146,

Centre Scientifique IBM France, Paris.
Panaget, Franck. 1994. Using a textual representation level component in the context of discourse and dialogue generation. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 127–136, Kennebunkport, ME, June.
Paris, Cécile L. and Donia R. Scott. 1994. Intentions, structure and expression in multi-lingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 45–52, Kennebunkport, ME, June.
PenMan Project, The. 1989. The PenMan documentation. Technical Report, USC/Information Sciences Institute, Marina del Rey, CA.
Reiter, Ehud. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, ME, June.
Reithinger, Norbert. 1990. Popel—A parallel and incremental natural language generation system. In Cécile Paris, William Swartout, and William Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, Boston, MA, pages 179–200.
Rubinoff, Robert. 1986. Adapting mumble: Experience with natural language generation. In *Proceedings of AAAI-86*, pages 1063–1068, Philadelphia, PA, August. American Association for Artificial Intelligence, Morgan Kauffman.
Rubinoff, Robert. 1992. *Negotiation, Feedback, and Perspective Within Natural Language Generation*. Ph.D. thesis, CIS Department, University of Pennsylvania, Philadephia, PA, December. Available as Technical Report MS-CIS-92-91.
Sibun, Penelope. 1991. *Locally Organized Text Generation*. Ph.D. thesis, University of Massachusetts at Amherst, Amherst, MA. Available as COINS Technical Report 91-73.
Sondheimer, Norman K. and Bernhard Nebel. 1986. A logical-form and knowledge-base design for natural language generation. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 612–618, Philadelphia, PA, August. American Association for Artificial Intelligence.
Thompson, Henry. 1977. Strategy and tactics: A model for language production. In *Papers from the Thirteenth Regional Meeting*, pages 651–668. Chicago Linguistics Society.
Wanner, Leo. 1994. Building another bridge over the generation gap. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 137–144, Kennebunkport, ME, June.