




Automating the Generation of Coordinated Multimedia Explanations

Steven K. Feiner and Kathleen R. McKeown
Columbia University



We developed an explanation system to overcome the disadvantages of conventional authoring in multimedia applications. COMET not only determines what to say, but how to say it.

Sometimes a picture is worth the proverbial thousand words; sometimes a few well-chosen words are far more effective than a picture. Pictures often describe objects or diagram physical actions more clearly than words do. In contrast, language often conveys information about abstract objects, properties, and relations more effectively than pictures can. Pictures and language used together can complement and reinforce each other to enable more effective communication than can either medium alone. In this sense, multimedia information systems may greatly increase effective communications.

Fortunately, technical advances are beginning to reduce the cost of hardware for computer-based multimedia and hypermedia. First-generation authoring facilities let users create presentations that include text, graphics, animation, and video. Regardless of the basic functionality or interface provided, however, multimedia authoring systems require authors to possess even more skills than do single-medium authoring systems. Not only must authors be skilled in the conventions of each medium, but they must also be able to coordinate multiple media in a coherent presentation, determining where and when to use different media, and referencing material in one medium from another. Furthermore, since the presentation must be authored in advance, the ways in which it can be customized for an individual user or situation are limited to those built in by the author.

To overcome the disadvantages of this predesigned authoring, we have developed an experimental test bed for the automated generation of multimedia explanations. COMET (Coordinated Multimedia Explanation Testbed)¹ has as its goal the coordinated, interactive generation of explanations that combine text and three-dimensional graphics, all of which is generated on the fly.

In response to a user request for an explanation, COMET dynamically determines the explanation's content using constraints based on the type of request, the information available in a set of underlying knowledge bases, and information about the user's background and goals. Having determined *what* to say, COMET also determines *how* to express it at the time of generation. The pictures and text that it uses are not "canned": COMET does not select from a database of conventionally authored text, preprogrammed graphics, or prerecorded video.

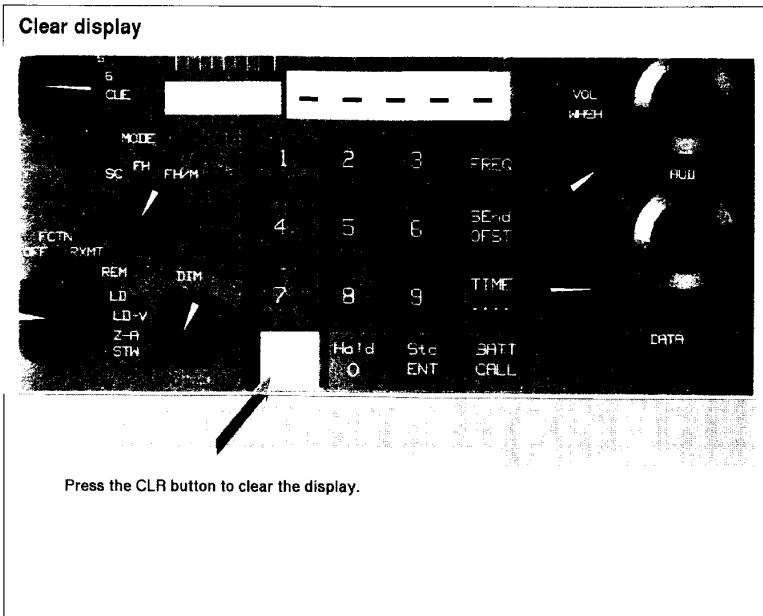


Figure 1. COMET's directions for clearing the radio display.

Instead, COMET decides which information should be expressed in each medium, which words and syntactic structures best express the portion to be conveyed textually, and which graphical objects, graphical style, and picture structure best express the portion to be conveyed graphically. COMET's text and graphics are created by separate *media generators*, each of which can communicate with the other.

We first provide a brief overview of COMET's domain and architecture. Then we focus on the specific ways in which COMET can coordinate its text and graphics. *Coordination* begins with the choice of media in which specific information is communicated. For example, an object's complex shape may be shown in a picture, rather than described in text, while a sentence may describe a causal relation between several actions involving the object. Coordination also means applying knowledge about what information is expressed in text to influence the generation of graphics, and vice versa. Thus, the graphics generator may use the fact that a causal relation is being communicated in text to determine how it depicts other information, even though the relation itself is not depicted. Finally, coordination means using knowledge about how information is expressed in other media in deci-

sion-making. For example, if the graphics generator shows the location of an object by highlighting it, the text generator can refer to "the highlighted object."

Overview

Much of our work on COMET is being done in a field maintenance and repair domain for a military radio receiver-transmitter. When provided with a set of symptoms, COMET generates multimedia explanations that instruct the user in how to carry out diagnostic tests. The user interacts with COMET by means of a simple menu and can initially choose to request instructions for a specific procedure or to invoke the diagnostic component. In the latter case, an underlying expert system is called to determine the problems that the radio is experiencing and to identify their causes.

The user selects symptoms from a menu. If the expert system decides that a set of diagnostic tests must be run to determine the cause of the problem, it calls the explanation component to tell the user how to carry out these tests. Explanations consist of one or more steps that are presented in a series of displays. Although our emphasis thus far has been on generating explanations, rather than on navigating through them, COMET's menu interface also provides rudimentary facilities for exploring the explanation by paging for-

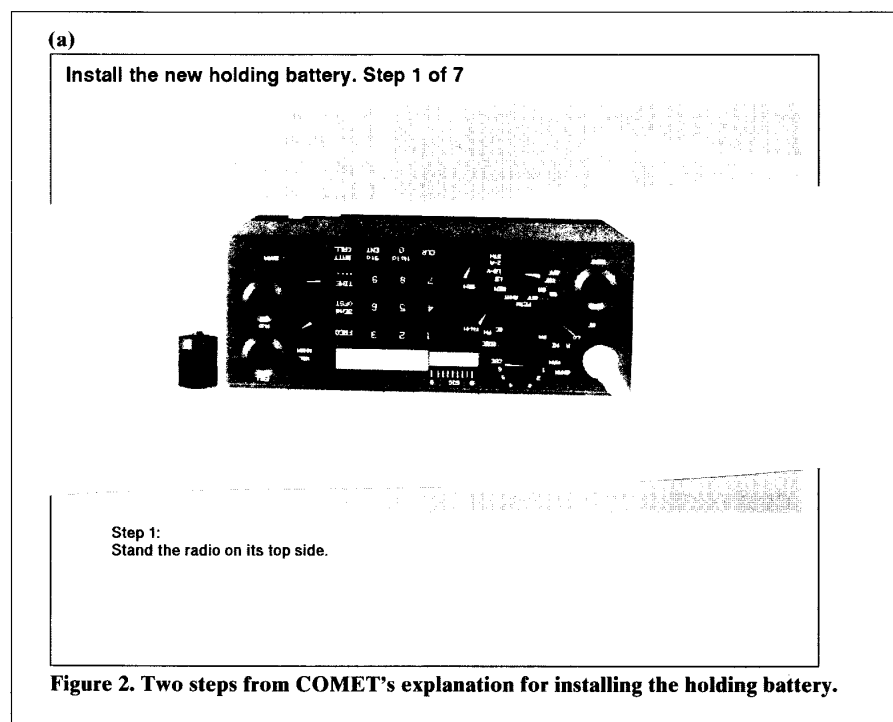


Figure 2. Two steps from COMET's explanation for installing the holding battery.

ward and backward through its steps. It can also access steps by name.

Figure 1 shows COMET's explanation for clearing the radio's display. Figure 2 shows the beginning of COMET's multistep explanation for installing a new "holding battery." (The holding battery provides power for the radio's memory when the main battery has been removed.) In these first two steps, the user is instructed to turn the radio upside down and remove the cover plate from the battery compartment. Replacing the holding battery is the first of a series of actions that COMET instructs the user to perform in the course of troubleshooting loss of radio memory, a symptom that the user selected from COMET's menu.

System organization. COMET consists of a set of parallel processes that cooperate in the design of an explanation, as shown in Figure 3. On receiving a request for an explanation, the content planner uses text plans, or schemas,² to determine which information from the underlying knowledge sources should be included in the explanation. COMET uses three different domain knowledge sources: a static representation of domain objects and actions encoded in the Loom knowledge repre-

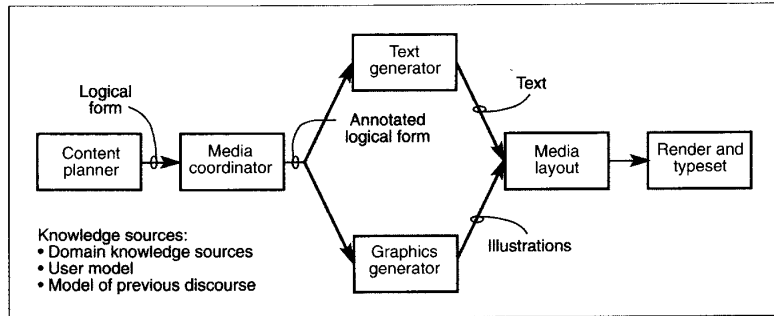


Figure 3. System architecture.

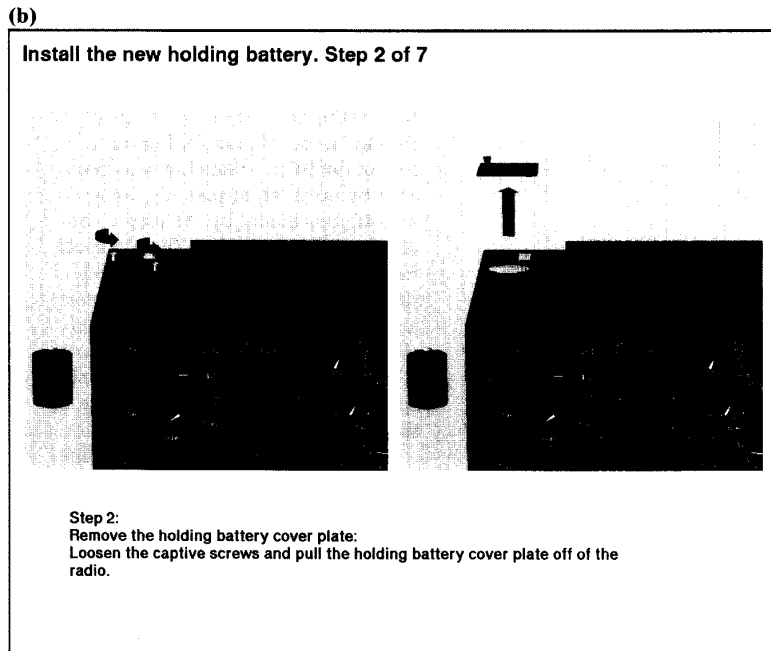
sentation language,³ a diagnostic rule base, and a detailed geometric knowledge base needed for graphics generation. It also maintains a user model and a model of the previous discourse. These knowledge sources are used by all system components to construct the explanation, not just by the content planner. Consequently, they are shown separately at the bottom of the figure without arrows to each module.

The content planner produces the full content for the explanation, represented as a hierarchy of logical forms⁴ (LFs), as explained in the sidebar titled "Unification in COMET." The LFs then are

passed to the media coordinator. This component refines each LF by annotating it with directives that indicate which portions are to be produced by each of the media-specific generation systems. The text generator and graphics generator each process the same LFs, producing fragments of text and graphics that are keyed to the LFs they instantiate. The output from both generators is processed by the media-layout component, which formats the final presentation for the low-level rendering-and-typesetting software.

COMET's major components run in parallel on up to five networked workstations. Text and menus are displayed through the X Window System, while 3D shaded graphics are rendered by Hewlett-Packard's Starbase graphics package. Each example shown in the figures takes approximately 10-20 seconds to generate and display following the initial user request.

One main feature of COMET's architecture is the use of the LF as a type of blackboard facility. (A blackboard⁵ is a central repository in which a system component can record its intermediate decisions and examine those of other components.) Each component reads and annotates its LF, continually enriching it with further decisions and finer specifications until the explanation is complete. Annotations include directives (like the media coordinator's choice of medium) and details about how a piece of information will be realized in text or graphics (like the proper verb to convey an action). While the annotated LF serves as a blueprint for the final explanation, it also allows for communication between media-specific components. For example, when deciding which expressive possibilities best con-



Unification in COMET

COMET uses FUF, an efficient extended version of Functional Unification Grammar,⁶ for media coordination. FUF also performs two text-generation tasks (selecting words and generating syntactic structure) and part of graphics generation (mapping the LFs to a communicative goal language supported by the graphics generator). Each component has its own "grammar" that is unified non-deterministically with the LF to annotate it with directives or further specifications. The result is a cascaded series of FUF grammars, each handling a separate task.

In FUF, both the input LF and the task grammar are represented using the same formalism, a set of attribute-value pairs. A value can be an atomic symbol or, recursively, a set of attribute-value pairs. For example, consider the following fragment of an input LF:

```
(substeps
  [((process-type action)
    (process-concept c-push)
    (roles ( . . . ) . . . )])
```

This fragment contains a single attribute, *substeps*, whose value is a set of attribute-value pairs. It contains three subattributes. The first, *process-type*, specifies the type of substep process as an action. The second, *process-concept*, indicates that the specific action is a knowledge-base concept called *c-push* that represents the action of pushing. The third, *roles*, specifies the actors and objects that participate in the action. The value of the *roles* attribute is a set (not shown here). Additional attribute-value pairs occur in a full LF.

Annotation is accomplished by *unifying* the task grammar with the input and is controlled by the grammar. For each attribute in the grammar that has an atomic value, any corresponding input attribute must have the same value. (Technically, it must have a compatible⁶ value. As one example, when the grammar attribute has the value *Any*, the input attribute can have any value.) When the values are different, unification fails. When the attributes match and both values are sets, unification is applied recursively to the values, and the result replaces the input value. When the input LF does not contain the grammar attribute, the attribute and its value are added to the input. Any attributes that occur in the input but not in the grammar remain in the input after unification. Thus, unification matches only the relevant subsections of the input. Note that unification is similar to set union, since enriched attribute-value pairs from both input and grammar are merged.

A fragment of the media coordinator's grammar states that an action should be realized in both graphics and text:

```
((process-type action) ;; If process is an action
 (media-graphics yes) ;; Use graphics
 (media-text yes) ;; Use text
 . . .))
```

On unifying this fragment of the grammar with the value of the *substeps* fragment of the input LF, FUF first checks whether the attribute *process-type* occurs in the input. Since it does and the value following it is *action*, FUF now checks whether the attribute *media-graphics* occurs in the input. Since it does not, FUF adds to the input the attribute-value pair (*media-graphics yes*), a directive indicating that the action is to be realized in graphics. Similarly, it adds the directive (*media-text yes*), specifying that the action is also to be realized in text. Thus, the first attribute-value pair is used as a test for this grammar portion. When it matches the input (the input LF describes an action), the input is annotated with the remaining attribute-value pairs. If the input LF had contained a different *process-type* (like *abstract*), unification with this portion of the grammar would have failed and FUF would have attempted unification with a new portion.

In most uses of the media coordinator grammar, a fragment is recursively applied to small nested segments of the input LF at least once. For example, the input LF can contain a causal relation, with two roles, both of which are actions. In this case, the grammar fragment as shown matches each role. A different fragment matches the causal relation (one that annotates it to be realized in text only). The roles of the actions are annotated recursively, as are their modifiers.

vey the specified content, a media generator can examine decisions made in other media by reading their annotated LFs and use that information to influence its own choices. COMET uses a single mechanism, FUF (Functional Unification Formalism), to make annotations throughout the system. This allows for additional bidirectional interaction between COMET's components through the use of unification, as described in the sidebar. Before describing coordination in more detail, we briefly discuss some of COMET's key components, focusing on their individual capabilities.

Media coordinator. This component performs a fine-grained analysis of an input LF to decide whether each portion should be realized in either or both media. After conducting a series of informal experiments and a survey of literature on media effectiveness, we distinguished six different types of information that can appear in an LF. We have categorized each type as to whether it is more appropriately presented in text or graphics. We use graphics alone for location and physical attributes, and text alone for communicating abstract actions and expressing connectives that indicate relationships among actions, such as causality. Both text and graphics represent simple and compound actions. The media coordinator has a FUF grammar that maps these information types to media.

Figure 4 shows a representative portion of the annotated LF produced by the media coordinator for Figure 1. The part of the LF in roman font was generated by the content planner, while the annotations added by the media coordinator are in boldface. This LF specifies a single substep and its effect, where the substep is a simple action (*c-push*) and the effect is also a simple action (*c-clear*). The *c-push* substep has one role (the medium, *c-button-clr*), and it also specifies that the location and size of the button should be included.

Figures 1 and 4 illustrate the fine-grained division of information among media. For example, location information is portrayed in graphics only, while the actions are realized in both text and graphics. In contrast, other information in the LF is communicated only in text, such as the causal relation between pushing the button and clearing the display.

Text generator. COMET's text generator² realizes the LF segments it has been assigned in text. It must determine both the number of sentences needed to realize the segments and their type (compound, simple, declarative, or imperative). It must select verbs to express LF actions, and nouns and modifiers to refer to LF objects. Finally, it must construct the syntactic structure for each sentence and linearize the resulting tree as a sentence.

The text generator divides into two modules that carry out these functions: the *Lexical Chooser* and the *Sentence Generator*. The Lexical Chooser selects the overall sentence type and the words, while the Sentence Generator produces individual sentences. Both modules are implemented using FUF.

The text generator can select words based on a variety of underlying constraints. This enables it to use a number of different words for the same LF concepts, depending on the context. The result is a wider variety of more appropriate output. COMET can use constraints from the underlying knowledge base, from previous discourse, from a user model, and from syntax. For example, for the knowledge-base concept c-install, the text generator can use "install," "reinstall," or "return." It makes a choice based on previous discourse (that is, what it has already told the user). Thus, after the user has installed the new holding battery, COMET instructs the user to remove the radio's main battery to check the new holding battery's functionality. At this point, COMET uses the previous discourse to select the verbs "reinstall" and "return" when instructing the user to put the main battery back in the radio. If the user had not been previously instructed to remove the battery, COMET would have selected the verb "install."

COMET can also avoid words that the user does not know. For example, it

```

((cat lf)
 (directive-act substeps)
 (substeps
  [((process-type action)
   (process-concept c-push)
   (mood non-finite)
   (speech-act directive)
   (function ((type substeps)
              (media-text yes)
              (media-graphics no)))
   (roles
    ((medium
     ((object-concept c-button-clr)
      (roles
       ((location ((object-concept c-location)
                  (media-graphics yes)
                  (media-text no)))
        (size ((object-concept c-size)
              (media-graphics yes)
              (media-text no))))))
      ...
     ))
    (cat lf)
    (media-graphics yes)
    (media-text yes))]
 (effects
  [((process-type action)
   (process-concept c-clear)
   (mood non-finite)
   (function ((type effects)
              (media-text yes)
              (media-graphics no)))
   (speech-act assertive)
   (roles
    ((agent
     ((object-concept c-display)
      (roles
       ((location ((object-concept c-location)
                  (media-graphics yes)
                  (media-text no)))
        (size ((object-concept c-size)
              (media-graphics yes)
              (media-text no))))))
      ...
     ))
    (cat lf)
    (media-text yes)
    (media-graphics yes)))]))

```

Figure 4. Logical form for Figure 1.

generates "Make sure the plus lines up with the plus" instead of "Check the polarity" when describing battery installation to a user not familiar with the word "polarity." The novel use of FUF to represent the lexicon efficiently provides a variety of different interacting constraints on word choice.

Graphics generator. IBIS (Intent-Based Illustration System⁷) generates illustrations designed to satisfy the communicative goals specified in the annotated LFs that it receives as input. The communicative goals that IBIS currently supports include showing absolute and relative locations of objects, physical properties (such as size, shape, material, and color), state (such as a knob setting), change of state (such as the change in a knob setting), and a variety of actions (such as pushing, pulling, turning, and lifting). In designing an illustration, IBIS controls all aspects of the picture-making process: the objects included and their visual attributes, the lighting specification, the graphical style used in rendering the objects, the viewing specification, and the structure of the picture itself.

IBIS uses a generate-and-test approach. The IBIS rule base contains at least one design rule for each communicative goal that can appear in an input LF. Each design rule invokes a set of stylistic strategies that specify high-level visual effects, such as highlighting an object. These strategies are in turn accomplished by still lower level rules that realize the strategies. The lower level rules create and manipulate the graphical depictions of objects included in the illustration and modify the illustration's lighting specification, viewing specification, and rendering information.

For example, IBIS uses a combination of techniques to portray the location of the button in Figure 1, as requested in the LF of Figure 4. It selects a viewing specification that (1) locates the button panel centrally in the illustration, (2) makes additional, surrounding context visible, and (3) ensures that both the object and context are recognizable. It highlights

the button by modifying the intensity of the lights that illuminate the objects in the illustration.

IBIS rules evaluate the success of each task that it performs. This is important because of the complex interactions that can occur in an illustration. Consider object visibility. Each object may be obscured by or obscure other objects. IBIS must determine whether visibility constraints are violated and address these by modifying the illustration. If an IBIS strategy doesn't succeed, it can backtrack and try another one. For example, if illuminating an object doesn't make it brighter than surrounding objects, IBIS can try to decrease the intensity of the lights illuminating the surrounding objects.

Media coordination

A multimedia explanation system must coordinate the use of different media in a single explanation. It must determine how to divide explanation content between different media such as pictures and text. Moreover, once the content has been divided, the system must determine how material can be generated in each medium to complement that of the other media.

A few researchers are addressing the automated generation of coordinated multimedia explanations with emphasis on how the media can complement each other. Integrated Interfaces⁸ produces US Navy briefing charts, using rules to map objects in the application domain (a database of information about ships) into objects in the presentation. Sage⁹ explains how and why quantitative models change over time, while Wip¹⁰ explains physical actions like those of COMET's domain.

Integrated Interfaces and Sage operate in a two-dimensional world of charts and graphs, and do not address the problems of describing objects and actions in 3D. Integrated Interfaces also uses many design rules specific to the particular kind of briefing chart it produces. Although Wip also emphasizes tight media coordination in application to 3D domains, its content planner takes an incremental approach. Each piece of information to be communicated is assigned sequentially to its generators. Evaluations of potential success are returned to the planner to help determine media assignments even before enough

information is provided for an entire sentence or illustration.

In contrast, COMET provides its media generators with more information at a time in a common LF that describes what the other generators have been assigned. The LF can also be enriched with accomplishments of the other generators. Thus COMET gives its media generators more context from which to work in making their initial decisions, while still allowing feedback.

Here we focus on two aspects of media coordination in COMET. First, we show how the use of a common content-description language allows for more flexible interaction between media, making it possible for each generator to query and reference other generators. By passing the same annotated description of what is to be described to each generator, we permit each generator to use information about what the other generators present to influence its own presentation. Then, we show how bidirectional interaction between the media-specific generators is necessary for certain kinds of coordination. Bidirectional interaction allows COMET to generate explanations that are structurally coordinated and that contain cross-references between media.

Common content description

All components in COMET following the content planner share a common description of what is to be communicated. Just as modules accept input in the same formalism, they can also annotate the description as they carry out its directives. This design has the following ramifications:

- It lets text and graphics influence each other.
- Communicative goals are separated from the resources used to carry them out.
- It provides a mechanism for text and graphics generators to communicate.

Mutual influence. Since both the text generator and graphics generator receive the same annotated content description as input, each knows which goals are to be expressed in text, in graphics, or both. Even when a media-specific generator does not communi-

cate a piece of information, it knows that the information is to be conveyed to the user; thus, it can use this knowledge to influence its presentation. Consider a portion of the explanation that COMET generates to instruct the user in how to install the holding battery. The second step of the explanation (Figure 2b) was generated from a complex LF that consists of one goal (to remove the holding battery cover plate) and two complex substeps that carry out that goal. As Figure 2b illustrates, the media coordinator determines that the goal is to be generated only in text ("Remove the holding battery cover plate:") and that the substeps are to be shown in both media.

Although IBIS depicts only the substeps of the LF, it receives the entire annotated LF as input. Since it receives the full LF, and not just the pieces assigned to graphics, IBIS knows that the actions to be depicted are steps that achieve a higher level goal. Although this goal is not itself realized in graphics, IBIS uses this information to create a composite illustration.⁷ This type of illustration consists of an integrated set of pictures that work together to achieve a common set of goals that cannot be accomplished in a single "simple" illustration. In this case, IBIS rules do not include any satisfactory way to show the radio with its cover plate and captive screws in different positions in one illustration.

If IBIS were to receive only the substeps, it would have no way of knowing that the substeps are being described in relation to a higher level goal. It may end up producing two separate illustrations, just as it does for each simple LF, such as that shown in Figure 2a. Thus, information conveyed in the explanation as a whole, but not in graphics, influences how IBIS depicts other information.

Separation of goals from resources. Because we are using a common content-description language, content must be specified at a level that is appropriate for all generators. We have found that by expressing content as a combination of communicative goals and the information needed to achieve these goals, each generator can select the resources it has at hand for accomplishing its assigned goals. In text generation, this means the selection of specific syntactic or lexical resources (using passive

voice to indicate focus, for example). In graphics generation, it means the selection of a conjunction of visual resources (modifying an object's material and the lights that illuminate it to highlight it, for example).

Consider again the explanation shown in Figure 1 and its associated annotated LF in Figure 4. The main goal of the first part of the LF is to describe an action (c-push) and its role (medium). Subgoals include referencing an object (for example, c-button-clr, the clear button) and conveying its location and size. IBIS and the text generator use different resources to achieve these goals. For example, the text generator selects a lexical item, the verb "press," to describe the action. "Press" can be used instead of other verbs because of the characteristics of the medium, c-button-clr. If the medium were a slider, a verb such as "push" or "move" would be required. In contrast, IBIS uses a *metaobject*, an object that does not itself represent any of the objects in the world being depicted. In this case, the metaobject is an arrow that IBIS generates to depict the action of pushing the button. To refer to the clear button itself, the Sentence Generator uses a definite noun phrase, whereas IBIS highlights the object in the picture.

A mechanism for communication. Since both generators understand the same formalism, they can provide more information to each other about the resources they have selected simply by annotating the content description. Thus, the content description serves as a blackboard to which all processes can write messages. We use this facility for coordinating the internal text structure with pictures.

Bidirectional interaction

Certain types of coordination between media can only be provided by incorporating interacting constraints between text and graphics. Two-way communication between the media-specific generators may be required as they carry out their individual realizations. Furthermore, coordination may only be possible once partial decisions have been made by the media-specific generators. For example, the text generator needs

to know how the graphics generator has depicted an object before it can refer to the object's visual properties in the illustration. Here we discuss two types of coordination that require bidirectional interaction: coordination of sentence breaks with picture breaks, and cross-referencing text and graphics.

Coordinating sentence breaks with picture breaks. In addition to revealing several dimensions along which to assign information to media, our media coordination experiments also demonstrated a strong preference for tight structural coordination between text and graphics. Our subjects much preferred sentence breaks that coincided with picture breaks. While multiple sentences accompanying one picture were found satisfactory, subjects strongly objected to a single sentence that ran across several pictures.

Including this type of coordination in COMET requires two-way interaction between text and graphics. Both text and graphics have hard and fast constraints that must be taken into account to achieve sentence-picture coordination. IBIS uses a variety of constraints to determine picture size and composition, including how much information can easily fit into one picture, the size of the objects being represented, and the position of the objects and their relationship to each other. Some of these constraints cannot be overridden. For example, if too many objects are depicted in one picture, individual objects may be too small for clarity.

This situation suggests that constraints from graphics be used to determine sentence size and thereby achieve coordination between picture and sentence breaks. However, some grammatical constraints on sentence size cannot be overridden without creating ungrammatical — or at least very awkward — text. Each verb takes a required set of inherent roles. For example, "put" takes a medium and to-location. Thus, "John put." and "John put the book." are both ungrammatical. Once a verb is selected for a sentence, this can in turn constrain minimal picture size; the LF portion containing information for all required verb roles should not be split across two pictures. Consequently, constraints from text must also be taken into account.

In COMET we incorporate this interaction by maintaining two separate tasks that run independently, each annotat-

ing its own copy of the LF when a decision is made and querying the other when a choice about sentence or picture structure must be made. Once a verb is selected for a sentence, the text generator annotates its copy of the LF by noting the roles that must be included to make a complete sentence. At the same time, the graphics generator annotates its LF with the mapping from the pieces of information to be communicated by graphics to the identifiers of the illustrations in which it intends to communicate the information.

When different sentence structures are possible, the text generator uses the graphics generator's annotations to make a choice by unifying the graphics generator's LF with its own. Consider the example of clearing the display shown in Figure 1. IBIS generates one picture showing the action and its effect; the text generator produces one sentence incorporating the effect as a purpose role of the action ("... to clear the display"). IBIS could depict this action in many ways, depending on the situation. For example, a pair of "before" and "after" pictures can be used, the first showing the action "push" about to occur and the second showing the cleared display. This picture structure can be especially useful in locating the objects participating in the action by showing their appearance prior to this event. Figure 5 (on the next page) shows what happens when IBIS's style rule base is modified so that a composite "before" and "after" pair is preferred. After consulting the annotated logical form, the text generator produces two separate sentences.

Cross-references. One important goal of media coordination is to allow material in one medium to cross-reference material in another. COMET provides for two kinds of cross-referencing: structural and content. The former refers to the coarse structure of the material being referenced. For example, a sentence could refer to an action by mentioning that it is depicted in one of the two pictures on the display. In contrast, a content cross-reference refers to the material's content, such as an object's position in a picture or the way in which the object is highlighted. Structural cross-references require only high-level knowledge of how the material being referenced is structured, whereas content cross-references require low-level

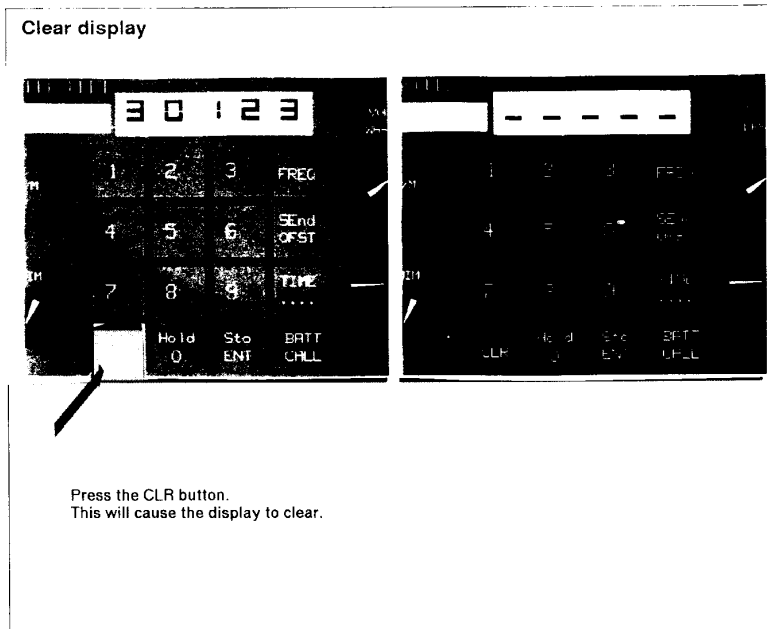


Figure 5. An alternative explanation for clearing the radio display. A composite illustration containing two pictures is generated with coordinated text (compare with Figure 1).

knowledge of how the material's communicative goals are realized.

COMET's text generator can make both structural and content cross-references to IBIS illustrations. IBIS constructs a representation of each illustration it generates that is indexed by the LF. This representation contains information about the illustration's hierarchical structure, the identity and position of the objects that it depicts, and the kinds of illustrative effects used in constructing the illustration (like highlighting or cut-away views). The text generator queries this representation to generate cross-references. For example, it can refer to information that is communicated "... in the left picture" (structural cross-reference) or mention "... the old holding battery shown in the cut-away view" (content cross-reference).

Current status and limitations

COMET can provide instructions for maintenance and repair procedures in two different contexts. A user can directly request instructions for a specific

procedure through a menu interface (essentially, asking "How do I do x ?"). Alternatively, during symptom diagnosis, the user can request an explanation for any diagnostic procedure the system specifies must be carried out. COMET can explain over 40 complex procedures represented in the knowledge base. A variety of explanations are provided for a single procedure, depending on user background, explanation context, or previous discourse. For example, COMET can vary the vocabulary (like not using the word "polarity" if it isn't in the user's vocabulary), the illustration design (like attempting to reuse the preceding illustration's viewing specification for the next illustration to avoid confusing camera motion), or the information communicated (like omitting an explanation for a procedural step if it was explained recently). COMET also has preliminary facilities for answering follow-up questions of the form: "What is an x ?", "Where is the x ?", or "Why should I do x ?"

COMET was designed to be as domain independent as possible. It can be adapted with minimum effort to a new task-based domain when tasks and actions are encoded using a standard plan-based representation and objects are

represented using frames. The content planner can produce the content for explanations of tasks or actions represented as plans, as well as for follow-up questions. Because many artificial intelligence systems use plans and frame-based representations, COMET's explanation facilities can be used in a wide range of applications. Similarly, the Sentence Generator grammar, the Lexical Chooser rules, and IBIS rules apply to any domain, task-based or not. In fact, these components have already been used in a number of applications under development at Columbia University. The media coordination rules we developed also apply to any domain that includes actions, physical objects, and abstract relations.

To handle a new domain, it would be necessary to augment the lexicon (adding new vocabulary), the Loom knowledge base (adding new plans and objects), and the graphics knowledge base (adding the new objects' detailed geometry and physical properties). These are currently substantial tasks, but so is the effort required to create conventionally authored explanations for a new domain. Also, note that some domain-dependent information, such as the graphics knowledge base, would ideally be available in CAD/CAM databases created when the objects to be documented were designed. COMET's rule bases were designed for domains involving physical objects. Therefore, handling domains that stress abstract relations among abstract concepts (like statistical analyses of numeric variables) would require major changes. The changes include different content-planning strategies, a different media coordinator grammar, and new graphics generation approaches (for example, adding a method for designing quantitative data displays^{9,12}).

The COMET testbed has allowed us to explore many ways to coordinate the generation of text and graphics. Our present and future work on COMET and its components includes the development of additional generators that support the temporal media of speech and animation. (IBIS already allows for direct, dynamic user control of the viewing specification.⁷) Our work also includes the design of a browsing/navigation facility

for COMET's explanations. We are developing a media layout component that will rely on the media generators' annotations to determine the relationships among pieces of text and graphics. This will allow COMET to group related items spatially.

We plan to allow feedback from the media generators to affect assignments made by the media coordinator and the selection of communicative goals made by the content planner. Our use of unification has the potential to make such feedback possible. Currently, we use an overall control structure for efficiency, calling the unifier separately for each grammar. Instead, we could call the unifier once for the combined series of grammars, thus allowing complete interaction through unification among the types of constraints. In this scenario, a decision made at a later stage of processing can propagate back to undo an earlier one. For example, information about the syntactic form selected can propagate back to the lexical chooser to influence verb choice.

While COMET is a research prototype, we believe that far more powerful systems will someday generate high-quality multimedia explanations for users in a variety of domains. Potential applications include education (explaining scientific phenomena) or even basic home repairs (coaching the user through troubleshooting a broken appliance). Although the hardware needed to run our current system is beyond the reach of most users, rapid improvements in the price-performance ratio will soon make high-performance real-time 3D graphics a fundamental capability of any computer system. In our work on COMET, we have attempted to lay some of the groundwork for the kinds of knowledge-based user interfaces that technological advances will soon make feasible. ■

Acknowledgments

Research on COMET is partly supported by Defense Advanced Research Projects Agency Contract N00039-84-C-0165, the Hewlett-Packard Company AI University Grants Program, National Science Foundation Grant IRT-84-51438, New York State Center for Advanced Technology Contract NYSSTF-CAT(88)-5, and Office of Naval Research Contracts N00014-82-K-0256, N00014-89-J-1782, and N00014-91-J-1872. COMET's development is an ongoing group

effort and has benefited from the contributions of Michael Elhadad (FUF), Dorée Seligmann (IBIS), Andrea Danyluk (diagnostic rule base), Yumiko Fukumoto (media coordinator), Jong Lim (static knowledge base and content planner), Christine Lombardi (media coordinator), Jacques Robin (Lexical Chooser), Michael Tanenblatt (knowledge base), Michelle Baker, Cliff Beshers, David Fox, Laura Gabbe, Frank Smadja, and Tony Weida.

References

1. S. Feiner and K. McKeown, "Coordinating Text and Graphics in Explanation Generation," *Proc. Eighth Nat'l Conf. Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park, Calif., 1990, pp. 442-449.
2. K.R. McKeown et al., "Language Generation in COMET," in *Current Research in Language Generation*, C. Mellish, R. Dale, and M. Zock, eds., Academic Press, London, 1990.
3. R. MacGregor and R. Bates, "The Loom Knowledge Representation Language," Tech. Report ISI/RS-87-188, Information Sciences Institute, University of Southern California, 1987.
4. J. Allen, *Natural Language Understanding*, Benjamin Cummings Publishing Company Inc., Menlo Park, Calif., 1987.
5. R. Reddy et al., "The Hearsay Speech Understanding System: An Example of the Recognition Process," *Proc. IJCAI 73*, Morgan Kaufmann Publishers, Palo Alto, Calif., 1973, pp. 185-193.
6. M. Kay, "Functional Grammar," *Proc. Fifth Mtg. Berkeley Linguistics Soc.*, Berkeley Linguistics Society, Berkeley, Calif., 1979.
7. D. Seligmann and S. Feiner, "Automated Generation of Intent-Based 3D Illustrations," *Computer Graphics (Proc. SIGGRAPH 91)*, Vol. 25, No. 4, July 1991, pp. 123-132.
8. Y. Arens, L. Miller, and N. Sondheimer, "Presentation Design Using an Integrated Knowledge Base," in *Intelligent User Interfaces*, J. Sullivan and S. Tyler, eds., Addison-Wesley, Reading, Mass., 1991, pp. 241-258.
9. S. Roth, J. Mattis, and X. Mesnard, "Graphics and Natural Language as Components of Automatic Explanation," in *Intelligent User Interfaces*, J. Sullivan and S. Tyler, eds., Addison-Wesley, Reading, Mass., 1991, pp. 207-239.
10. W. Wahlster et al., "Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation," *Proc. European Chapter Assoc. Computational Linguistics*, 1991, pp. 8-14.
11. E. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.
12. J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Trans. Graphics*, Vol. 5, No. 2, Apr. 1986, pp. 110-141.



Steven K. Feiner is an associate professor of computer science at Columbia University. His research interests include image synthesis, applications of artificial intelligence to computer graphics, user interfaces, animation, hypermedia, virtual worlds, and visualization.

Feiner received a PhD in computer science from Brown University in 1987. He serves on the editorial boards of *Electronic Publishing* and *ACM Transactions on Information Systems*, and has coauthored the book *Computer Graphics: Principles and Practice*, 2nd ed. Feiner received an Office of Naval Research Young Investigator Award in 1991 and is a member of the IEEE Computer Society and ACM.



Kathleen R. McKeown is an associate professor of computer science at Columbia University. Her research interests include natural language generation, user modeling, expert system explanation, and natural language interfaces.

McKeown received a PhD in computer science from the University of Pennsylvania in 1982. She was elected president of the Association of Computational Linguistics for a term beginning in January, 1992, and currently serves as its vice president. McKeown has served on the executive council of the Association for Artificial Intelligence and was program cochair of its annual conference in 1991. She received a National Science Foundation Presidential Young Investigator Award in 1985.

Readers may contact Steven K. Feiner and Kathleen R. McKeown at the Department of Computer Science, Columbia University, 500 W. 120 St., New York, NY 10027.