The Role of Underlying Structure in Text Generation

Robert Alan Granville
BBN Systems and Technologies Corporation
33 Moulton Street
Cambridge, Massachusetts 02138

Abstract

While a coherent organization is necessary for the generation of a multisentential paragraph, this organization itself conveys information, such as what knowledge is primary and what is secondary, and which of the various possible relationships between the pieces of knowledge the speaker wishes to make explicit. The organization of the message is an integral part of the message. Given this, it is wrong to assign the task of text structuring to any component other than the one that desires to convey the message in the first place. This paper describes *George*, a proposed system that organizes its text before the task of generation is begun. Particular attention is given to a new knowledge base paradigm called *functional hierarchy*, which is designed to facilitate explanation generation, and how it can be used to build Rhetorical Structure Theory representations that specify text organization before actual generation is begun.

Introduction

It is a well known fact that just as we can't string together a collection of words in an arbitrary order and achieve a well-formed sentence, we can't string sentences together in an arbitrary fashion and achieve a meaningful paragraph. The underlying structure of text and how this contributes to coherence is the subject of active research, most notably by Hovy [Hov88]. However this research seems to ignore that different organizations of sentences yield paragraphs with different meanings, that the *structure* of a paragraph is an integral part of what the paragraph conveys. Given this, it is wrong to assign the task of text structuring to any component other than the one that desires to convey the message in the first place.

This paper describes *George*, a proposed system that organizes its text before the task of generation is begun. *George* is an extension of the MACH-III intelligent tutoring system [KGM90] that uses the expert system knowledge base to choose the appropriate underlying text structure along with the knowledge to be presented in the text, rather than have the text generator try to build the underlying structure when presented with only the pieces of knowledge to be conveyed.

After a brief review of Hovy's approach toward text structuring in the next section, we explore the role of text structure and how it affects the total message being conveyed in the third section. The fourth section describes *George*, paying particular attention to the intelligent tutoring system knowledge base and its unique organization into a *functional hierarchy*, a new knowledge base paradigm designed to facilitate explanation generation. The fifth section discusses how we can use the functional hierarchy knowledge base to help direct the selection of the message structure, using examples from MACH-III which demonstrate again why this selection process must occur before generation. We conclude this section with a brief discussion of some of the open issues to be resolved as part of this research effort.

Hovy's Approach

In [Hov88] Hovy correctly states that arbitrary orderings of facts will not lead to satisfactory text. The problem as he sees it then is to determine which orderings will lead to acceptable text, and with his text structuring system he attempts to address this problem.

Hovy takes his input to be an unordered set of chunks of knowledge of equal weight (importance) generated by an expert system. Since an arbitrary ordering won't work, and there are far too many possible combinations for an exhaustive search ($n!$ for n sentences), the structure must somehow be built, and this building must be guided by the sentences themselves.

Hovy uses Rhetorical Structure Theory [MT87], or RST, as the basis for his structure. RST is a formalism for specifying the relationships between pieces of text. Hovy's text structuring system builds a communicative plan using RST relations as its operators. The constraints of the RST relations become the operator preconditions, and the effect or goal fields of the relations become the effects of the operators. These operators are used in a standard plan generation system with the resulting plan being an RST structure representing the organization of a coherent paragraph conveying all the input facts.

Structure as Message

An underlying assumption in Hovy's approach is that the expert system intends merely to convey a set of facts or "chunks of knowledge". This is an overly simplifying assumption. A coherent paragraph communicates more than the simple chunks of knowledge stated in its sentences. It also communicates the *relationships* between these chunks, the very relationships RST represents. When a speaker has a communicative goal, the goal is not to present a set of facts in any coherent fashion, but rather to present them in an organization that conveys in addition to the facts how the speaker perceives them to be related. The underlying structure of the message is as much a *part* of the message as is the contents of the message.

Consider the following two example paragraphs:

[1] *You must start the ignition by turning the car key. Otherwise you don't start the motor, and can't drive the car.*

[2] *You must turn the car key. Otherwise you can't drive the car since you didn't start the motor by starting the ignition.*

Paragraphs [1] and [2] contain the same four simple facts, but they use different organizations and have different effects. One difference is the points each paragraph emphasizes. In [A], starting the ignition is most central, with starting the motor as the second most important clause. In contrast, [B] emphasizes turning the car key strongest, followed next by being able to drive the car. Another difference is that Paragraph A] explicitly shows a relationship between starting the ignition and turning the car key, while no such relationship is stated in Paragraph B. Even though the two paragraphs contain exactly the same four simple facts, the messages the convey are *not* equivalent.

Now let us return to Hovy, and consider his example in [Hov88] of a coherent paragraph with the RST structure shown in Figure 1:

(1) *The system asks the user to tell it the characteristic of the program to be enhanced.* (2) *Then the system applies transformations to the program.* (3) *In particular, the system scans the program* (4) *in order to find opportunities to apply transformations to the program.* (5) *Then the system resolves conflicts.* (6) *It confirms the enhancement with the user.* (7) *Finally, it performs the enhancement.*

He contrasts this with the following, which presents the same information, he says, but in an incoherent fashion:

(1) *The system performs the enhancement.* (2) *Before that, the system resolves conflicts.* (3) *First, the system asks the user to tell it the characteristic of the program to be enhanced.* (4) *The system applies the transformations to the program.* (5) *It confirms the enhancement with the user.* (6) *It scans the program* (7) *in order to find opportunities to apply transformations to the program.*

But we can, of course, find an RST for this second paragraph, as in Figure 2, so the problem with this example is not that it doesn't have an underlying structure. Rather, the underlying structure doesn't represent the relationships between the facts as we know them to be. These facts simply do not make sense as seven unrelated steps performed in the order presented in this paragraph. That is, part of the complete message (it's organization) is incorrect.

An Alternative Approach

Rather than attempt to build an appropriate RST that will fit isolated, unstructured input from an expert system, *George* takes both the facts to be presented and the relational structure of these facts as its input. This is possible because in the case of MACH-III the knowledge base of the expert system component is organized as a functional hierarchy.

Functional Hierarchy

Functional Hierarchy is a new paradigm for organizing expert system knowledge bases, based on the procedural abstraction principles of Liskov and Guttag [LG86]. Functional hierarchy differs greatly from production rules (the customary basis of an expert system) in that functional hierarchy rules define the actions a system can take, rather than the conditions under which actions *may* take place. The concept of "action" is expanded to include *all* actions the system takes, including control decisions, rather than just changes to the database, thereby eliminating the need for a separate control structure. These rules are arranged into a hierarchy, where the action of a rule is defined as a combination of other actions.

There are several advantages to using a functional hierarchy over a production rule system. All the benefits that accrue from the use of procedural abstraction are manifest:

- easier to understand
- easier to write
- easier to maintain and modify

All the knowledge in the system can be explicitly represented in the system since there is no need for external mechanisms (such as conflict resolution procedures) where knowledge can be hidden. This allows enhanced explanation capabilities. Finally, the knowledge can be easily organized into a desirable framework.

Functional hierarchy was used very successfully in MACH-III (Maintenance Aid Computer for HAWK – Intelligent Institutional Instructor), an intelligent simulation and tutoring system to help train novice mechanics to troubleshoot the HAWK HIPIR radar (ANPQ-57) developed by BBN Systems and Technologies Corporation for the Army Research Institute. The system is currently being used as part of the curriculum of the US Army's training school. For a complete description of MACH-III and the use of functional hierarchy, see [KGM90], [KGM89], and [MTG89].

An example from the functional hierarchy used in MACH-III (called the troubleshooting tree) is shown in

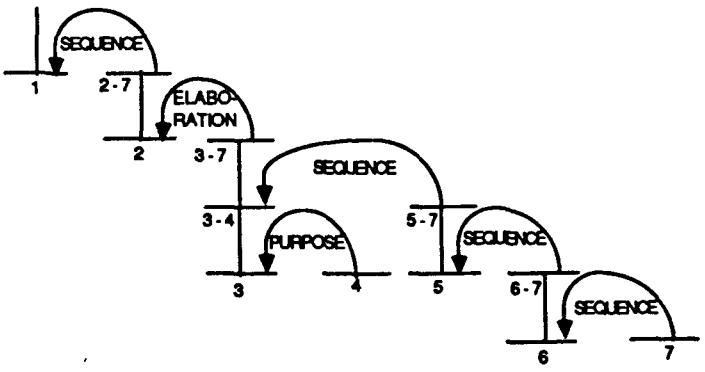

Figure 1: RST for Hovy's Coherent Paragraph

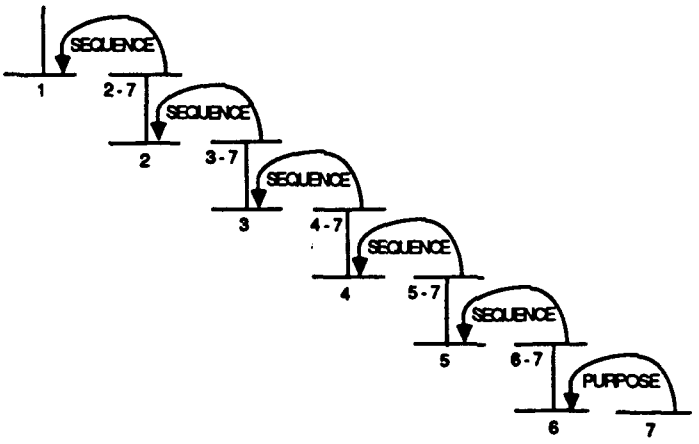

Figure 2: RST for Hovy's Incoherent Paragraph

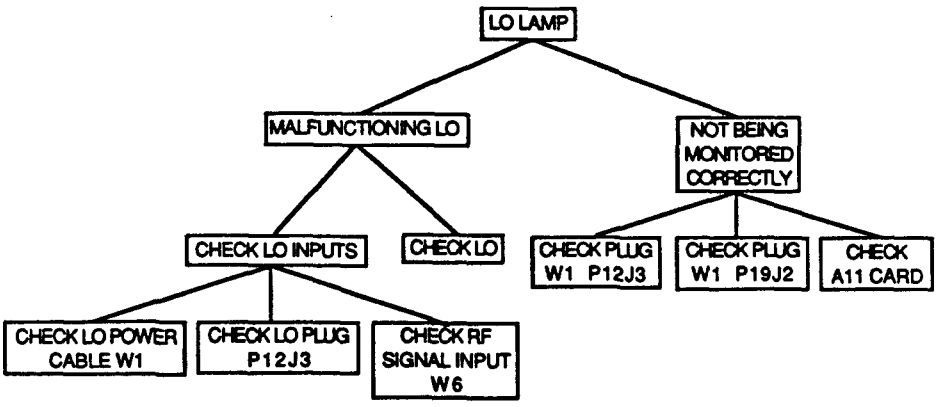

Figure 3: LO Fault Example

Figure 3. The root node of this tree, labelled **LO LAMP**, defines how to determine why the LO Lamp is lit, indicating a "local oscillator fault. In other words, this node defines how to isolate an LO problem. According to the tree, isolating the LO fault is done by checking for an LO malfunction (the left child of the **LO LAMP** node), and checking to see if the LO function is being monitored correctly (the right child of the **LO LAMP** node). It is important to note that at this root node *neither* subaction is defined. That is, the rule defined by this node contains calls to the procedures to check for an LO malfunction and to check to see if the LO function is being monitored correctly, but does *not* contain definitions for these procedures.

Continuing our example, checking for an LO malfunction (the **MALFUNCTIONING LO** node in the tree) consists of checking LO inputs (the left child) and checking the LO itself (the right child). Since checking individual devices, such as the LO, is a primitive action in the MACH-III simulation that needs no further definition, the **CHECK LO** node has no children. The **CHECK LO INPUTS** node, defining how to check LO inputs, has three children, and so checking LO inputs consists of three steps: checking the LO power cable W1, checking the LO plug P12J3, and checking the RF signal input cable W6. Since these actions are again primitive in the simulation, they need no further definition, and their corresponding nodes have no children.

It should be noted that from a strictly functional point of view the intermediate rules are not necessary. We could have simply defined the action of isolating an LO fault as performing the seven primitive actions, resulting in the tree in Figure 4, and the resulting system would still be able to correctly solve the problem. However, the system would not be able to explain *why* these steps were being taken. In contrast, the system of Figure 3 knows, for example, that checking LO plug P12J3 is necessary to determine whether the LO inputs are functioning properly, which in turn must be known to determine whether the local oscillator is malfunctioning. The point is that the functional hierarchy concisely specifies the knowledge the system explicitly has in its knowledge base, and exactly how the pieces of knowledge are interrelated. Thus the hierarchy defines the level of explanation of which the system is capable.

Explanations from a Functional Hierarchy

Since the troubleshooting expert system knowledge base is organized as a tree, troubleshooting the radar consists of moving correctly through that tree from node to node. This simplifies the task of evaluating the appropriateness of a student's actions, since each action she takes "moves" the student from one node (the current node) to another. Each move can be one of four types. The simplest case is when the node moved to represents an action already done. (In the troubleshooting procedures for the HAWK, there is never any need to repeat steps.) The next case is when the node moved to is a child of the current node, in which case the system judges the action as good. The third case is when the action moves the student to a node that is not a descendant of the current node, but is part of the troubleshooting tree for the current problem. The system evaluates such moves to be skipping around from task to task before they are completed. The final case is when the node isn't in the current problem's troubleshooting tree, and the system determines that the action is inappropriate for the problem at hand.

The current MACH-III system takes these evaluations coupled with the knowledge of the nodes involved to create messages in English for the student using a scheme based on templates with slots. Each of the four cases has templates with slots to be filled by information contained in the nodes, such as which components of the radar are involved and what tests are performed on them by the action represented.

For example, the following is a typical message actually generated by MACH-III to report an action judged to be "skipping around":

> *Checking the* **W4 P1 P2J3** *is a reasonable step since it is part of checking RCVR RF inputs. However, this leaves investigating whether noise is being introduced during modulation undone.*

If the student asks for more information, the system generates the following:

> *Checking the* **W4 P1 P2J5** *is part of checking RCVR RF inputs, which is part of investigating whether feedthrough is not being correctly removed, which is part of considering the RCVR Noise Lamp. On the other hand, investigating whether noise is being introduced during modulation consists of checking the* **Spin Motor** *power input, the* **W2,** *checking the* **Nutating Scanner Assembly,** *and checking the* **Scan Driver Assembly.** *Therefore, checking the* **W4 P1 P2J5** *is not part of investigating whether noise is being introduced during modulation.*
>
> *Investigating whether noise is being introduced during modulation isn't finished yet since checking the* **Scan Driver Assembly,** *which is part of investigating whether noise is being introduced during modulation, isn't done.*

Functional Hierarchy with RST

George is an extension of MACH-III that dynamically generates the critiques for the expert troubleshooting system. Instead of using the knowledge resident in the nodes of the troubleshooting tree to instantiate slots in a prewritten template, the system organizes this knowledge into an RST structure representing those relationships of the facts the troubleshooter wishes to convey. The instantiated structures are then traversed in left to right order to build input messages for the Mumble-86 text generation system [MMA*87].

Preliminary research has shown that subtle differences in what is being stressed in a text, what is made secondary, and what is being assumed can be manifested by variations in the underlying RST representation. By definition, the

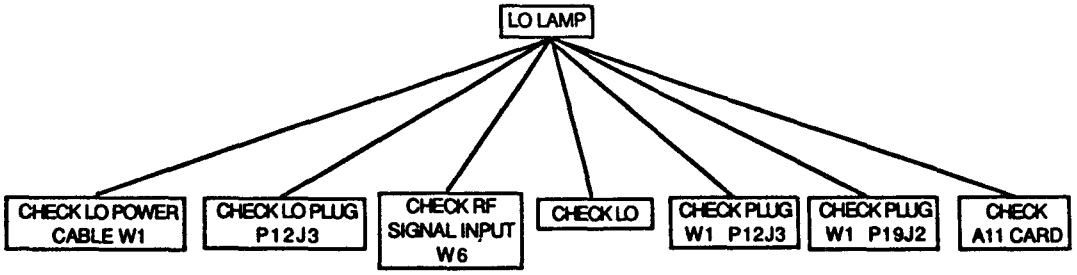

Figure 4: LO Fault Revisited

nuclei of an RST structure are more central to the text than are the satellites. Additionally, the relationships explicitly represented in the structure are conveyed in the text (either explicitly or through the use of "clue words"), while relationships not represented are left tacit.

As an example, consider the ten paragraphs shown in Figure 5. Paragraphs **B** through **J** are variations of an actual text produced by MACH-III, Paragraph **A**. All ten paragraphs contain the same "chunks" of knowledge, but each presents this knowledge differently. The main thrust of Paragraph **A** is that you should execute a Lamp Test in order to know whether the BITE Test indications are correct. The synopsis of Paragraph **B** is that if you don't push the Lamp Test Switch, you won't know how to proceed. The other two facts are merely used to support this contention.

These two paragraphs also express different relationships between the four facts. In Paragraph **A**, we are told explicitly that a Lamp Test is executed by pushing the Lamp Test Switch. In contrast, Paragraph **B** has no direct relation between executing a Lamp Test and pushing the Lamp Test Switch. Figures 6 and 7 show the differing RST structures for these two paragraphs. (Note that we have not labelled the relationships in these two figures for clarity's sake, since for this example we have only discussed the existence or nonexistence of relationships, and not what those relationships are.)

With ten different ways of stating the same four facts (by no means an exhaustive list), each of which presents them in a different light, and each of which makes explicit and tacit different relationships between these facts, it would be impossible for a text generating system to determine which was the "best" variation given only the four facts. Consequently, the component that decides these four facts are the correct ones to present must also be the component that decides what is the primary information, what is the secondary or supporting information, and what relationships between these pieces of information must be stated. That is, this component must also decide the underlying RST structure for itself.

George is capable of deciding the RST structures for its text through fairly straightforward mechanisms. It follows from the particular functional hierarchy knowledge base in the troubleshooting expert system, there are basically four types of messages that have to be communicated, and selection among these four types is dictated by the functional hierarchy and the student's position in this structure. Then instead of choosing from among a relatively small set of templates with slots instantiated by the appropriate troubleshooting tree nodes as was done in MACH-III, we can choose from among a relatively small set of RST structures (essentially the RST structures for the templates) with the same troubleshooting tree nodes providing the pieces of knowledge the RST structures are organizing. This gives the resulting system greater flexibility in generation over the template and slot generator of MACH-III, where individual paragraphs (the granularity of the MACH-III templates) read well enough, but multiparagraph texts (as were sometimes required) were predictably poor since each paragraph was generated in isolation.

There are of course two important issues that remain to be discussed, because they remain to be resolved. The first issue concerns the choice of a specific RST for a message. While the MACH-III functional hierarchy contributes a great deal to the selection process, it is by itself not sufficient. While some relationships and emphases can be determined from the contents of the message, there are many subtle variations, such as those shown in the paragraphs of Figure 5, which make the selection of an RST harder. Exactly how changes in the RST affect the resulting surface structure and what influences contribute to decisions about these changes remain open problems.

The second issue is the problem of multiparagraph text. Our experience with MACH-III showed us that there is more to multiparagraph text than stringing together isolated well-formed paragraphs, not surprising since the same is true of multisentential paragraphs and multiword sentences. The underlying structure of the entire text, depicting interparagraph relationships and emphases, must also be determined for successful generation. Fortunately, RST is capable of representing interparagraph structure as well as intraparagraph structure, giving us the framework for exploring how paragraph structure and total text structure interact and how these structures affect the surface text.

A.
(2) Executing a Lamp Test by (1) pushing the Lamp Test Switch should be the first thing you should do. (3) Otherwise you have no way of knowing whether the BITE Test indications are correct, (4) and thus no way of knowing how to proceed.

B.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (4) Otherwise you have no way of knowing how to proceed, (3) since you didn't determine whether the BITE Test indications are correct (2) by executing a Lamp Test.

C.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (2) Otherwise you don't execute a Lamp Test, (3) which means you have no way of knowing whether the BITE Test indications are correct, (4) and thus no way of knowing how to proceed.

D.
(1) Pushing the Lamp Test Switch (2) in order to execute a Lamp Test should be the first thing you should do. (3) Otherwise you have no way of knowing whether the BITE Test indications are correct, (4) and thus no way of knowing how to proceed.

E.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (2) Otherwise you don't execute a Lamp Test, (4) which means you have no way of knowing how to proceed, (3) since you have no way of knowing whether the BITE Test indications are correct.

F.
(2) Executing a Lamp Test (1) by pushing the Lamp Test Switch should be the first thing you should do. (4) Otherwise you have no way of knowing how to proceed, (3) since you have no way of knowing whether the BITE Test indications are correct.

G.
(1) Pushing the Lamp Test Switch (2) in order to execute a Lamp Test should be the first thing you should do. (4) Otherwise you have no way of knowing how to proceed, (3) since you have no way of knowing whether the BITE Test indications are correct.

H.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (2) Otherwise you don't execute a Lamp Test (3) in order to determine whether the BITE Test indications are correct, (4) and thus you have no way of knowing how to proceed.

I.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (3) Otherwise you have no way of knowing whether the BITE Test indications are correct (2) since you didn't execute a Lamp Test, (4) and thus you have no way of knowing how to proceed.

J.
(1) Pushing the Lamp Test Switch should be the first thing you should do. (4) Otherwise you have no way of knowing how to proceed, (2) since you don't execute a Lamp Test (3) in order to determine whether the BITE Test indications are correct.

Figure 5: 10 Paragraph Variations

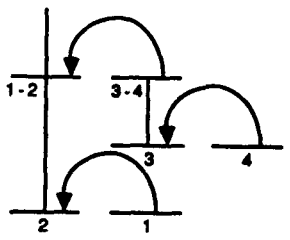

Figure 6: RST for Paragraph A

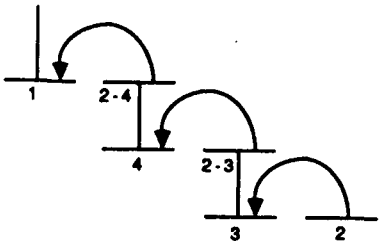

Figure 7: RST for Paragraph B

Conclusions

While a coherent organization is necessary for the generation of a multisentential paragraph, this organization itself conveys information, such as what knowledge is primary and what is secondary, and which of the various possible relationships between the pieces of knowledge the speaker wishes to make explicit. The organization of the message is an integral part of the message. Given this, it is wrong to assign the task of text structuring to any component other than the one that desires to convey the message in the first place.

George is a proposed extension of MACH-III that accepts this fact. Use of the functional hierarchy troubleshooting tree developed for MACH-III directs the choice of the correct RST structure from a small set of candidates, with information resident in appropriate troubleshooting tree nodes instantiating the structure. This instantiated RST structure can then be used to build an input message for the MUMBLE-86 generator. Thus text generation in *George* is more flexible than in MACH-III, and better flowing text is produced.

There are still several important issues to be resolved, most notably how the selection of a message's underlying structure is controlled and how multiparagraph texts can be properly organized and how this organization affects the final text. The combined use of functional hierarchy and RST as described in this paper gives a solid framework in which these issues can be explored.

Acknowledgements

Thanks to David McDonald, without whose encouragement and timely reviews of drafts this paper would never have been written.

References

[Hov88] Eduard H. Hovy. Planning coherent multisentential text. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 1988.

[KGM89] Laura C. Kurland, Robert Granville, and Dawn MacLaughlin. *HAWK MACH-III Explanations of the Receiver Troubleshooting Tree.* Technical Report, BBN Systems and Technologies Corporation, 1989.

[KGM90] Laura C. Kurland, Robert Alan Granville, and Dawn M. MacLaughlin. Design, Development and Implementation of an Intelligent Tutoring System (ITS) for Training Radar Mechanics to Troubleshoot. In *Journal of Machine-Mediated Learning*, Taylor & Francis, New York, 1990. in publication.

[LG86] Barbara Liskov and John Guttag. *Abstraction and Specification in Program Development.* MIT Press, Cambridge, Massachusetts, 1986.

[MMA*87] Marie W. Meteer, David D. McDonald, Scott D. Anderson, David Forster, Linda S. Gay, Alison K. Huettner, and Penelope Sibun. *Mumble-86: Design and Implementation.* Technical Report COINS Technical Report 87-87, University of Massachusetts at Amherst, Amherst, Massachusetts, 1987.

[MT87] William C. Mann and Sandra A. Thompson. *Rhetorical Structure Theory: A Theory of Text Organization.* Technical Report ISI/RS-87-190, Information Sciences Institute, Marina del Rey, California, 1987.

[MTG89] Dawn MacLaughlin, Yvette Tenney, and Robert Granville. *HAWK MACH-III Explanations of the Transmitter Troubleshooting Tree.* Technical Report, BBN Systems and Technologies Corporation, 1989.