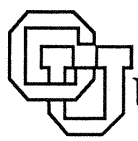
The Expression of Local Rhetorical Relations in Instructional Text

Keith Vander Linden
Susanna Cumming
James Martin

CU-CS-585-92 March 1992



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

The Expression of Local Rhetorical Relations in Instructional Text[1]

Keith Vander Linden, Susanna Cumming, James Martin

CU-CS-585-92 March 1992

University of Colorado at Boulder

Technical Report CU-CS-585-92
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309

NOTE: This is the complete version of a paper appearing in the Proceedings of the the 6th International Workshop on Natural Language Generation (Vander Linden et al., 1992).

The Expression of Local Rhetorical Relations in Instructional Text[1]

Keith Vander Linden[2,4], Susanna Cumming[3,4], James Martin[2,4]

[2]Department of Computer Science
[3]Department of Linguistics
[4]Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430

email: linden@cs.colorado.edu,
scumming@clipr.colorado.edu,
martin@cs.colorado.edu

March 1992

Abstract

We are currently engaged in a project to study the generation of instructional texts for common consumer devices. Our initial efforts have focused on an exhaustive corpus-based analysis of instruction booklets for cordless telephones. In this paper, we present our analysis of the way the various processes used in this domain are expressed. Our emphasis here is on the relationship between the surface grammatical coding of these processes and the underlying rhetorical structure of the text. This analysis has been formalized using a systemic-functional framework, with the resulting system networks forming the basis for the IMAGENE text generation system.

[1]This work was supported in part by NSF Grant IRI-9109859, awarded to the third author.

Contents

List of Figures

List of Tables

1 Introduction

We are engaged in a project to study the generation of a kind of text we call *instructional text.* By instructional text we have in mind the kind of booklet you find when you purchase a common household appliance such as a telephone, alarm clock or VCR. The overriding goal of this project is to produce a text generation system that is capable of taking the output of a planner and producing appropriate instructional text. Thus, our goals overlap with those of other recent projects concerning instructions (Mellish and Evans, 1989; McKeown et al., 1990; Dale, 1990). These projects, however, have focused on issues such as the nature of the planner output, the coordination of figures and text, and the representation objects and actions. The focus of our work has been to discern the mapping from function to form in these texts by looking carefully at instructional texts actually produced by human writers. Therefore, our contribution to this area is the development and implementation of a formal characterization of what makes a good instructional text, based on a detailed corpus-based analysis of existing instructional texts.

The particular results presented in this report concern the surface grammatical coding of a variety of local rhetorical relations common in this domain. To make this discussion more concrete, consider the following alternatives for expressing a volume adjustment in the telephone domain.

(1) Adjust the volume to your preferred level *with the VOL LO/HI switch.*

(2) *Use the VOL LO/HI switch* to adjust volume to your preferred level. (code) [2]

Given that a process such as volume adjustment can be represented as an action frame with an instrument slot, it would seem that (1) would be the most congruent alternative to choose. However, our analysis of the corpus indicates that instrument clauses such as the one used in (2) are the preferred form. The alternation hinges on whether the switch itself is the local topic, in which case the clause form is used. The clause form naturally appears most often since these texts are typically organized around the buttons and switches that control the device. The alternative form appears most often in initial installation instructions before the controls have been fully explained. We have encountered many of these sorts of alternation during our exploration of this domain.

A generator that could identify and deal with these complex issues of grammatical choice fits together nicely with a number of ongoing research projects in the area of planning action sequences (Alterman et al., 1991; Gruber, 1990; Foley et al., 1991; Allen et al., 1991). These projects have focussed considerable attention on the planning of actions and structure of the output of these planners, which would provide a natural point at which to pick up the problem of natural language generation. Some of these projects have actually identified language as an important aspect of their work, but have not pursued it extensively.

Following a brief motivation of the theoretical tools we have used and a definition of instructional text, this paper will describe our methodology for finding, analyzing and formalizing these distinctions of form. It will then present the results of the analysis and how they are implemented in the IMAGENE (Instruction MAnual GENErator) text generation system.

1.1 Systemic-Functional Grammar

We have used the Penman implementation of the system network from systemic-functional grammar (Mann, 1985; Halliday, 1976; Hudson, 1971) to formalize the results of our study. Both theoretical and practical considerations have converged in dictating this choice for the current project. From a theoretical point of view, we are attracted to the system as an implementation of a functional approach to language. The basic research problem in text generation is identical to that of functional and cognitive linguistics: to discover the principles which allow speakers/writers to navigate the lexicogrammatical resources of

[2] Our convention will be to add a reference to the end of all examples that have come from the corpus, indicating which manual they came from. (code) and (exc) will stand for examples from the Code-a-Phone and Excursion manuals respectively (Code-a-phone, 1989; Excursion, 1989). All other examples are contrived.

their language and produce utterances/texts which satisfy simultaneous goals at a variety of levels, both cognitive and social in nature. As an implementation of the functional approach to language, Halliday's systemic theory is particularly useful, because the systemic formalism provides a convenient and consistent architecture for the description and implementation of the links between communicative goals and linguistic forms.

1.2 Rhetorical Structure Theory

We have used Rhetorical Structure Theory (or RST) (Mann and Thompson, 1988) as a descriptive tool in our corpus analysis and as a constructive tool in IMAGENE (for a discussion of the descriptive/constructive distinction, see Mann and Thompson, 1987b). It was developed at USC/ISI as a means of representing text structure — viewed in terms of the semantic and pragmatic relations that hold between text units at all levels — in a computational context. It was developed jointly by linguists and computer scientists, and is valuable in that it provides a means of incorporating well-established approaches from linguistics into a uniform notation.

While its suitability for representing interactive discourse is questionable, it has been successfully applied to a variety of written genres (Hovy, 1989; Hovy and McCoy, 1989; Moore and Paris, 1988; Scott and Souza, 1990; Cawsey, 1990). Generally a new domain has dictated modifications to the inventory of relations, but this very adaptability is one of its most useful features.

Earlier work on RST has focused on the higher levels of text structure: while a few researchers mentioned the desirability of extending rhetorical analysis below the clause level, this possibility has not yet been exploited. In our corpus, however, we found that most of the rhetorical complexity was located at the clausal and subclausal levels. This required the extension of RST into the clause in order to provide a uniform treatment for clauses and subclausal relations such as prepositional phrases. Consider the following examples of purpose relations, some at the clause level and some at the phrase level:

(3a) *To end a previous call*, hold down FLASH [6] for about two seconds, then release it. (code)

(3b) Follow the steps in the illustration below, *for desk installation.* (code)

(3c) The OFF position is primarily used *for charging the batteries.* (code)

Such examples, which are very prevalent in our data, challenge the widely held assumption that the principles governing inter-clausal relations are very different in kind from those governing intra-clausal relations. According to this view, the former can be characterized as simple (a single type of structural relation and a small finite set of relation types), highly productive, universal, and uniform (a single type of representation is appropriate for all levels from text to paragraph to sentence). Conversely, the latter is characterized as complex (requiring a large number of rules), highly constrained, language-specific (and thus learned), and diverse (different principles are involved in the organization of the noun phrase than the sentence; and within the noun phrase, different kinds of relations obtain between the head noun and e.g. its determiners and its adjectives). While this contrast does indeed hold at the level of form, at the levels of meaning and of discourse function we must recognize some degree of interpenetration between these two types of organization – specifically, phrasal constituents (such as the "for" phrases in the above examples) can have semantic content like that of the typical clause (they refer to actions or states) and relations with other constituents similar to interclausal relations, such as "purpose" (Cumming, 1991). Consequences of these observations for our implementation will be discussed below.

1.3 Instructional Text

In order to define what we mean by the term "instructional text" we collected around 50 text examples from various sources and in various languages that could be considered instructional. This body of text examples includes user manuals, recipes, text books, assembly instructions, and explanatory text. It became clear, however, that it did not display a sufficiently cohesive set of characteristics to allow study and implementation. We, therefore, have focused our attention on texts similar to the prototype text

mentioned above, that of a booklet which accompanies a common consumer device which is intended to instruct a non-specialist user on how to use the device. To be more precise, we have narrowed our consideration to texts that have the following three characteristics:

1. The text must describe the direct physical manipulation of a device by the reader. This rules out instructions concerning productive processes like recipes[3]. It also rules out text books, a text occasionally called "instructional" but which involves no physical manipulation.

2. The instructions and the devices must be intended for a wide non-specialist audience. This rules out many highly specialized, one of a kind, devices found in industry and the military. The text in these manuals tends to employ a highly prescribed form which we feel is unnatural. Another benefit of this restriction has been to guarantee the availability of a large quantity of texts in a wide variety of languages.

3. The device must have what we call *medium functionality*. This rules out tools that are either too simple, such as hammers, or too complex, such as large programming environments. The simple devices do not typically act as agents in the processes described, which is a phenomenon we are interested in studying. The complex devices may tend to give rise to altered writing mechanisms that don't match those employed for simpler devices. These texts, for example, are tailored for readers who are expected to neither understand the whole process in one reading nor remember this process from one use to the next.

These restrictions have allowed us to start with a coherent, if perhaps overly restrictive, body of texts having similarities of grammatical and rhetorical structure. Our initial study, described in this report, employs a corpus made up of such text from two cordless telephone manuals, comprising around 1700 words. It has allowed us to develop the tools and architecture to a level that will support a larger study. The larger study will involve the expansion of the corpus to include other examples of instructions that meet the three criteria listed above. Given that these criteria may rule out texts of compatible type, we are also ready to consider relaxing any of them, provided the new texts reflect similar characteristics.

2 Corpus Analysis: Using a Database Tool

In order to design a generator for instructional text, it was necessary to first do an analysis of the rhetorical and lexicogrammatical characteristics of "natural" instructional text. The basic problem we face is to identify systematic covariation between forms and functional environments; for this study, we focus on the grammatical realization (clause or phrase type) of sets of actions and states bearing various rhetorical relations to each other. Various aspects of this problem have of course been addressed in the functional syntax literature, which mention a variety of relevant factors (among more recent work, see e.g. Cumming, 1984; Haiman and Thompson, 1989; Ford, 1988, the papers by Noonan, Thompson and Longacre in Shopen (1985), and various analyses in the RST framework, including Ford, 1986; Thompson and Mann, 1987; Mann and Thompson, 1987a; Matthiessen and Thompson, 1987; Sanders et al., 1990). These works have proved fruitful as sources of hypotheses; however, they have all focussed on conversation, narrative, or expository prose, and we could not assume that the results would transfer to a new genre: in fact, clause combining strategies vary notoriously across genres (see e.g. Chafe, 1982; Beaman, 1984; Biber, 1988).

In conducting our analysis, we adopted a methodology of doing a comprehensive analysis of each of a small number of texts. This approach is motivated by the need in functional analysis for the identification of multiple instances of both target linguistic forms and functional environments across a single text so that recurring correlations can be identified. A full grammatical and rhetorical analysis of the entire text is necessary, since it is not possible to determine beforehand what aspects of the context may be relevant.

[3]This restriction has allows us to avoid the problem of reference to constantly changing objects, one that has been studied in detail by Dale (1988).

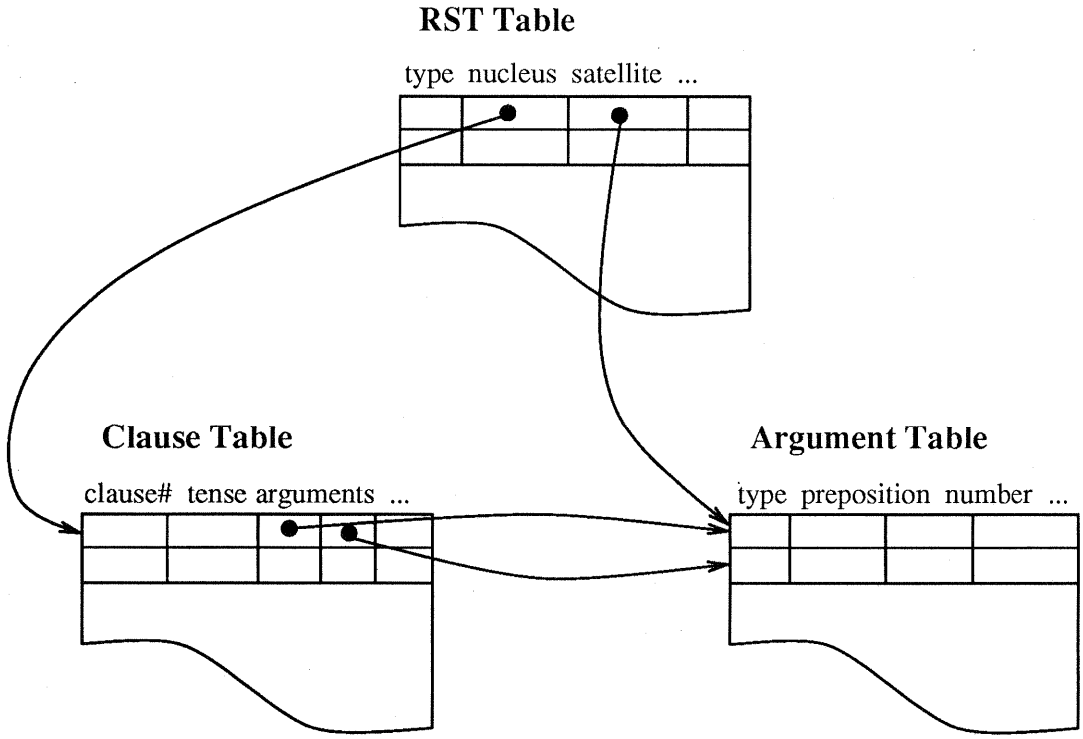


Figure 1: The Structure of the Instructional Text Database

The analysis was carried out by hand, since any automatic parser inevitably embodies assumptions about the answers to some of the very questions we were trying to address. Even with so small a corpus, however, the difficulty of forming hypotheses and testing them against the data required some automated help. To solve this problem, we created a database containing three tables relating to different levels of structure: prepositional and noun phrases, clauses, and nodes in the rhetorical structure[4]. The structure of this database is shown in Fig. 1. Hierarchical relationships in the text, both syntactic and rhetorical, were indicated via pointers between these files. Clause and phrase records contain, in addition to the text of the unit, fields for coding information such as determiner, number, and modification (for NPs) or linker, verb form, and argument structure (for clauses). The rhetorical structure file contains fields for indicating the nucleus and type of relation for each satellite, and the parent for each nucleus; all nodes contain pointers to the associated clause or phrase. In this way information about the core relations between rhetorical functions and morphosyntactic forms could easily be obtained, and it was a simple matter to test hypotheses against the entire corpus as they emerged. This tool is a partial implementation of the tool discussed in Cumming (1990).

3 Results of the Analysis: The System Networks

To formalize the results of this study, we have built system networks for the most important local rhetorical relations we found in our corpus: procedural sequences, purposes, preconditions, and results. These relations make up about 55% of the relations we identified in the corpus. Of the other types of rhetorical relations, joint and functional lists, which we address in the architecture but have not studied,

[4]The actual RST and grammatical analyses of the text were carried out by one of the authors and the examples crucial to the formalization of the results were reviewed by all the authors.

make up about 20%, elaboration and title relations, tending to be at higher levels, another 20%, and seven other less common relations, 5%.

Our findings indicate that, not surprisingly, sequences of actions are expressed as sequences of imperative clauses. Purposes and preconditions are expressed as subordinate clauses fronting the actions they describe and results as separate declarative sentences following the actions they describe. There are, however, numerous conditions that can alter this basic pattern. Actions, under certain circumstances, may be expressed as phrases attached to the end of the immediately following action. We call this process *rhetorical demotion* because the action, for rhetorical reasons, is not being stated as a separate imperative clause. Similarly, circumstances may dictate that relations normally expressed as subordinate clauses may be expressed as separate sentences. We call this *rhetorical promotion*. Here are some examples of these processes:

(4) Lift the handset and set the OFF/STBY/TALK [8] switch to TALK. Return the OFF/STBY/TALK switch to STBY *after your call.* (code)

(5) TONE—For touchtone systems. *Tone signalling service (from your phone company) is required for TONE features.* (code)

In (4), the "after your call" is coding the action of calling someone, but the writer considered this to be obvious, and thus rhetorically demoted it. This is appropriate in a cordless phone manual where the use of a normal phone is assumed. In (5), the writer considered having a particular service from the phone company important enough to rhetorically promote it. Occasionally, problems in understanding an instructional text arise from the inappropriate use of these promotions and demotions.

We have chosen the terms "promotion" and "demotion" because they employ a metaphor which implies a change from a more "basic" status. It is an undisputed universal of human languages that main, finite clauses function primarily to code new, independent actions and events, while noun phrases function to code time-stable entities. Non-finite clauses and nominalizations code actions and events which are either not new or dependent in some way (such as time, participants, or causality) on another event. These are the "prototypical" (in the sense of Hopper and Thompson, 1984) or "congruent" (in the sense of Halliday, 1985) functions of the various coding types. This claim has been justified both by means of cross-linguistic studies of the functions of similar forms in various languages (as in e.g. Hopper and Thompson, 1984) and by means of frequency counts in a single language (as in e.g. Thompson, 1987): the "congruent" mapping occurs more often. In our data, for instance, only 5% of the 65 reader actions were coded as demoted phrases, the rest were coded as imperative clauses. Similarly, only 8% of the 61 precondition expressions were coded as separate sentences.

We will now, after a short section on how to interpret system networks, discuss in some detail the functional distinctions made by the purpose, precondition, and result system networks and how they use text-based realization statements to construct a rhetorical structure.

3.1 System Network Notation

System networks from systemic functional grammar are basically decision networks read from left to right. We have employed four basic notations for these networks which are shown in Fig. 2. The notation for a system is shown in Fig. 2A. The name of the system is given on the left and available output features are given on the right. Only one of the of the features may be chosen. Figure 2B shows a notation indicating that both systems A and B are to be entered. Figure 2C and 2D show complex entry conditions for systems. The former indicates an OR-gate which is passed when either X or Y is true (or both), and the latter an AND-gate which is passed only when X and Y are true. A more detailed discussion of these formalisms can be found in Winograd (1972).

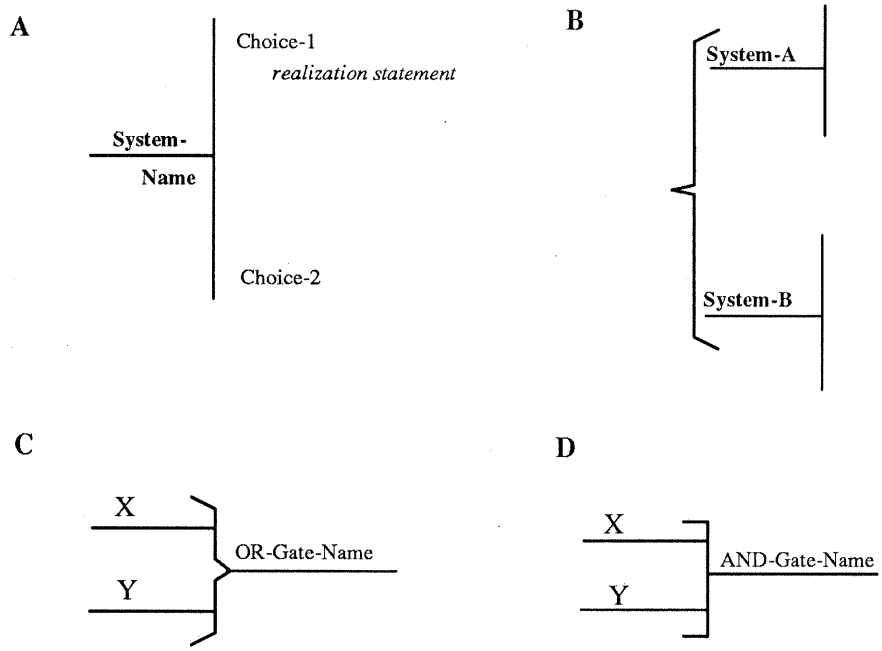


Figure 2: Notational Forms for System Networks

3.2 Purpose Relations in Instructional Text

Instructional text is action oriented and very often includes some statement of purpose for the actions it prescribes. Our study of these purpose expressions has resulted in the network shown in Fig. 3. Here are some representative examples of the purpose clauses/phrases we found in the corpus:

(6a) *To end a previous call*, hold down FLASH [6] for about two seconds, then release it. (code)

(6b) Follow the steps in the illustration below, *for desk installation.* (code)

(6c) The OFF position is primarily used *for charging the batteries.* (code)

(6d) *For frequently busy numbers*, you'll want to use REDIAL [7], and the pause will have to be in Redial memory. (code)

We can see two issues of choice at the rhetorical level in these examples. First, the purpose clauses/phrases can occur either before or after the actions which they motivate. Second, there are four grammatical forms to choose from. In (6a) we see a "to" infinitive form (tnf), in (6b) a "for" prepositional phrase with a nominalization ("installation") as the complement, in (6c) a "for" preposition with a gerund phrase as a complement, and in (6d) a "for" preposition with a noun phrase as a complement that is the goal of the corresponding action. The frequency with which these slots and forms appeared in the 33 purpose expressions in our corpus is given in Table 1.

We'll now give a brief discussion of how the systems in the **Purpose-Slot** and **Purpose-Form** sub-networks address these two issues. The **Purpose-Slot** sub-network formalizes Thompson's notion of the "vastly different functions" for initial and final purpose clauses (Thompson, 1985). In keeping with her analysis, **Purpose-Form** represents the factors determining the form of the purpose expression separately from those determining the slot. Taken together, they generate a greater range of purpose expressions than is typical in generation systems (see e.g. Hovy, 1988) and identify the functional reasons for choosing one form over the other.

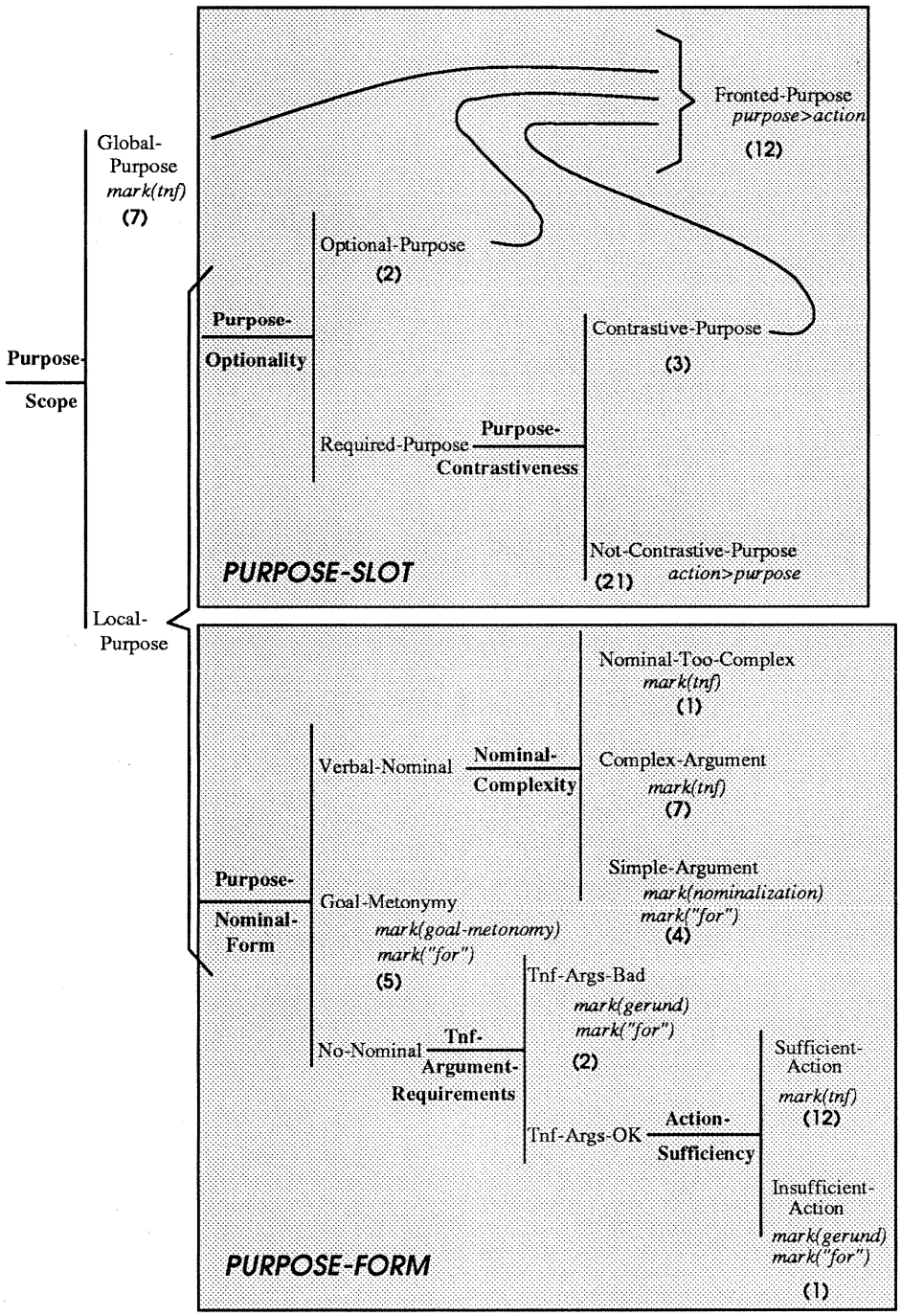


Figure 3: The Purpose System Network

Table 1: The frequency of various form and slot combinations of purposes in instructional text

	Initial	Final
To-Infinitive	8	12
For-Nominalization	0	4
For-Gerund	0	3
For-Goal-Metonymy	3	2
Other	1	0

3.2.1 Purpose Slot

The **Purpose-Slot** sub-network places the purpose clause/phrase in the final position in most cases. This matches the data given in Table 1, where the majority of the purpose clause/phrases are in the final position. The exceptions to this are when the scope of the purpose is multiple, the purpose is considered optional, or the purpose is considered contrastive.

The first exception, handled by the **Purpose-scope** system, concerns the number of actions the purpose pertains to. We could not, for instance, restate (6a) as "?? Hold down FLASH [6] for about two seconds, then release it *to end a previous call.*" As Thompson points out (Thompson, 1985), the purpose clause is often used, as it is here, to state a context in which the prescribed actions are to be interpreted, and thus should be fronted. The restatement also implies, incorrectly, that the purpose applies to the last action alone rather than to the sequence of actions.

The remaining exceptions occur when the purpose is considered optional or contrastive and are handled by the **Purpose-optionality** and **Purpose-contrastiveness** systems respectively. Here are examples of them from the corpus:

(7) *For more information and wall installation instructions*, see the Installation Notes on page 3. (code)

(8) If your battery is not holding a charge and you believe it has developed a "MEMORY" condition, drain it and recharge according to the outlined instructions. *In order to prevent "MEMORY" from occurring*, keep your handset out of the base for at least 2 hours a day. (exc)

In (7), the purpose is optional, that is, the reader may or may not want more information at this point in the text. The purpose is, therefore, stated first to set the appropriate context for the prescribed action. In (8), the purpose of preventing "MEMORY" is stated in contrast to the state of affairs in the previous sentence. It is thus fronted to set the appropriate context for the prescribed action.

3.2.2 Purpose Form

The **Purpose-Form** sub-network, which determines the grammatical form of purpose expressions, is more complicated. As illustrated in the diagram, the form and slot sub-networks share the **purpose-scope** system. For a purpose whose scope includes more than one action, the "to" infinitive form is used. Example (6a) illustrated this. These sorts of context-setting purposes are not demoted to phrase status.

The remainder of the **Purpose-Form** sub-network is based on the observation that instructional text is oriented toward reader actions. Because purpose clauses/phrases are not reader actions, they tend to be demoted to phrase status whenever possible. Thus, the **purpose-nominal-Form** system will realize a prepositional phrase with a nominalization as the complement whenever possible[5] This use

[5]The **purpose-Form** sub-network currently represents three discrete points along the continuum from fully nominal to fully verbal forms expressing the same action (Quirk et al., 1985), namely the nominalization, the gerund, and "to" infinitive.

of phrases with nominalizations in rhetorical relations is common in instructional text and elsewhere (Cumming, 1991).

Even if a nominalization exists, however, it still may not be used depending upon the determination of the **nominal-complexity** system. This system, based on the examples in our corpus, restricts nominalizations to a single, non-complex argument. Consider the following examples, each of which corresponds to one of the possible features of this system:

(9a) Use the VOL LO/HI [2] switch *to adjust volume to your preferred listing level.* (code)

(9b) ?? Use the VOL LO/HI [2] switch *for volume adjustment to your preferred listening level.*

(10a) FLASH uses proper timing *to avoid an accidental hangup.* (code)

(10b) ?? FLASH uses proper timing *for accidental hangup avoidance.*

In cases (9a) and (10a), taken from the corpus, there was a nominalization available, "adjustment" and "avoidance", but neither was used. The "adjustment" nominalization in (9b) was not used because it required more than one argument. The "avoidance" nominalization in (10b) was rejected because the argument "accidental hangup" was itself a nominalization and thus too complex. In both cases, the "to" infinitive form was preferred.

Goal Metonymy occurs in purposes when the goal of the purpose clause is more important than its action. This occurs in:

(11) *For frequently busy numbers*, you'll want to use REDIAL [7], and the pause will have to be in Redial memory. (code)

Here we have an ellipsis where the full purpose would be something like "to deal with frequently busy numbers" or "for dealing with frequently busy numbers". The object of the verb, in this case, is considered more important than the verb itself, allowing the metonymy.

If no nominalization is available, the **tnf-argument-requirements** and **action-sufficiency** systems will typically produce the "to" infinitive except when the infinitive form requires unwanted arguments or the action in the main clause is not sufficient to achieve the purpose. Here are examples of these cases:

(12a) The BATT LOW Light [9] comes ON when the battery is weak. The handset must be returned to the base *for recharging.* (code)

(12b) ?? The BATT LOW Light [9] comes ON when the battery is weak. The handset must be returned to the base *to recharge (the battery?).*

(13a) OFF is used primarily *for recharging the battery.* (code)

(13b) ?? OFF is used primarily *to recharge the battery.*

Examples (12a) and (13a) were found in the corpus, while the alternate "to" infinitive expressions, (12b) and (13b) seem suspect; (12b), because we would be required to restate "the battery" which is unacceptable in this context, (13b), because the "to" infinitive clause implies that putting the handset in OFF position is all that is required for recharging the battery (which is not the case with this telephone).

3.2.3 Coverage of the Purpose Expressions in the Corpus

The parenthesized numbers in the purpose system network diagram represents the quantity of examples in the corpus that each system handled correctly. So, for example, in the purpose network, Fig. 3, there were 12 purpose that were fronted for one reason or another, represented by the "(12)" in the diagram near the "Fronted-Purpose" or-gate. We've listed the statistics for each terminal choice, that is right-most choice in the network, and also for all choices leading into and-gates and or-gates. This convention will be maintained in the precondition and result system networks as well.

The purpose network, as described, explains 29 of the 33 purpose expressions in the corpus (88%). Of the 4 exceptions, 2 involve the use of "in order to" rather than just "to" in the "to" infinitive. We

have not been able to identify any functional difference between the two and have, thus, stayed with the simpler form (in conformity with the more concise style advocated by Strunk, 1979). The form and slot of these two examples are handled correctly.

The remaining two exceptions expose deeper problems that we will deal with when we have more examples on which to base a solution. They are as follows:

(14a) *Your call may be ended in any of the following manners:* 1. ...2. ... (exc)

(14b) To speak to a caller or *to place a call,* the OFF/STBY/TALK SWITCH should be set to Talk. (exc)

Example (14a) is the only expression of a purpose for either-or actions. Our current network would produce something like "*To end your call:* 1. ...2. ..." which incorrectly implies that steps 1 and 2 are to be taken in sequence.

Our purpose network would not have produced the form of the second purpose in (14b), but rather would have produced "for call placement." It seems that being expressed in conjunction with another "to" infinitive form forced this purpose to use the same form. It is also possible that the phrase "call placement" is not, but some measure, and "good" nominalization for "placing a call." We will wait for more data of this type before formulating a solution to this problem.

3.3 Precondition Relations in Instructional Text

Instructional text often includes statements of the preconditions for the actions it prescribes. It may also state these actions themselves as preconditions. Our study of these precondition expressions has resulted in the network shown in Fig. 4. Here are some representative examples:

(15a) *When the 7010 is installed* and *the battery has charged for twelve hours,* move the OFF/STBY/TALK [8] switch to STBY. (code)

(15b) *If you leave the OFF/STBY/TALK [8] switch in TALK,* move the switch to PULSE, ... (code)

(15c) Return the OFF/STBY/TALK switch to STBY *after your call.* (code)

The forms of precondition expressions, as you can see, are much more diverse than either purpose or result expressions. Each precondition is based on an action that would normally have been expressed with an imperative, active clause, but for various reasons has been rhetorically demoted to a precondition status. As a precondition, any action can be stated as either the action itself or the relevant condition brought about by that action. Example (15a) has one of each. "When the 7010 is installed" is an expression of the relevant condition brought about by the action "installing the 7010." We call this a *Terminating Condition* because it states the condition which is true following the termination of the action. Terminating conditions are always expressed as present tense relational clauses. "The battery has charged for 12 hours", from (15a), is a statement of the action of charging the battery. Action statements such as this can either be expressed as present-perfect clauses, as in this case, or present tense clauses as in (15b) depending upon the type of action being expressed. Example (15c) contains an action expressed as a precondition which is neither fronted nor in clause form.

These examples illustrate the three major issues of choice in expressing preconditions, the choices of slot, linker, and form. The frequency with which these slots and forms appeared in the precondition expressions in our corpus is given in Table 2. The following sections detail the system networks designed to automate these choices.

3.3.1 Precondition Slot

The Precondition-Slot sub-network, included in Fig. 4, determines whether or not to front a particular precondition expression. In the corpus, preconditions are typically fronted, and therefore the sub-network will default to fronting. There are four types of exceptions to this default which are illustrated here.

(16a) The BATTERY LOW INDICATOR will light *when the battery is the handset is low.* (exc)

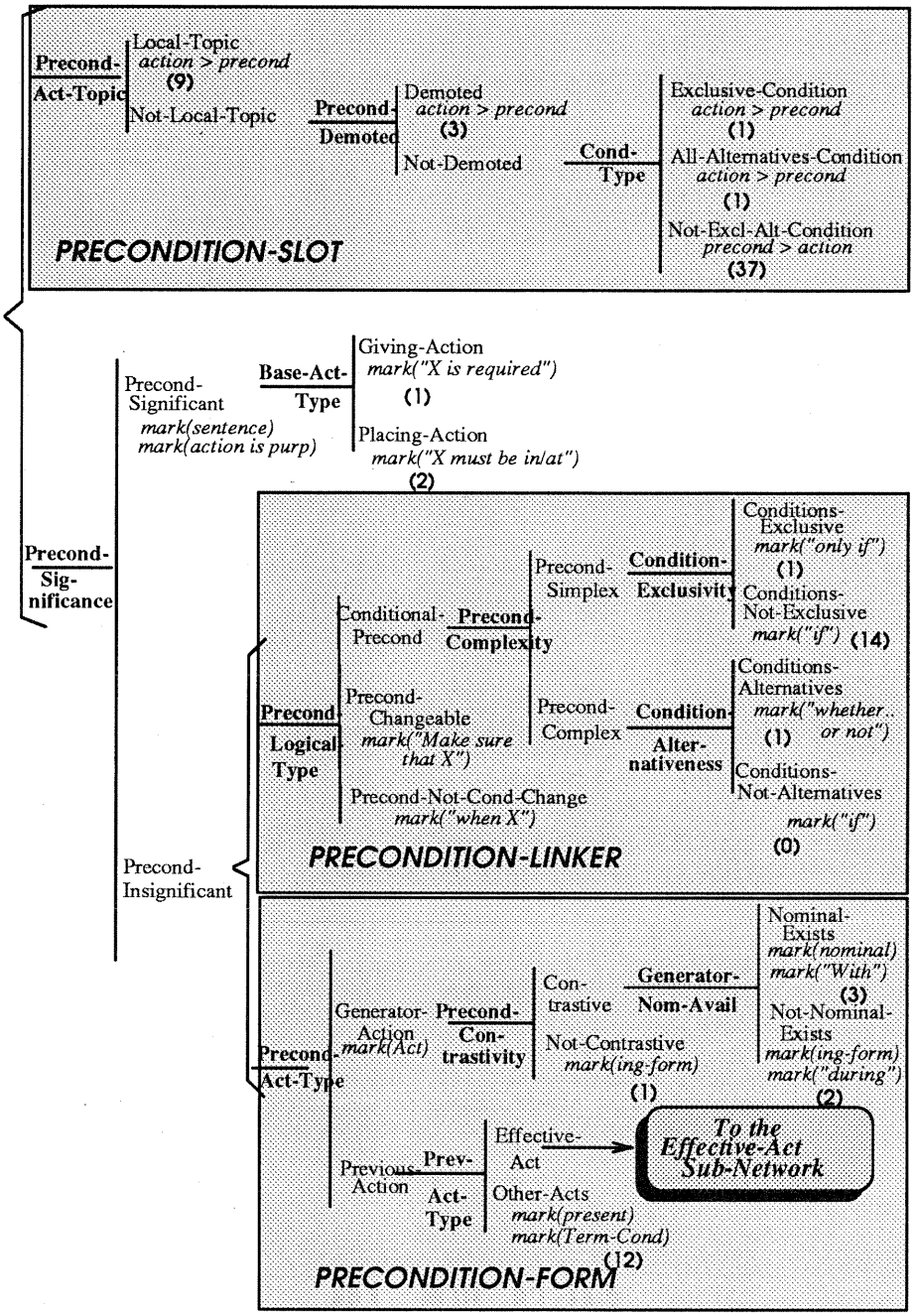


Figure 4: The Precondition System Network

Table 2: The frequency of various form and slot combinations of Preconditions in instructional text

	Initial	Final
Separate Sentence	0	3
Present Condition	9	9
Future-Condition	0	1
Present-Act	17	0
Present-Perfect Act	1	4
Future-Act	1	1
ing-Form	4	1
Nominalization	4	2
Relative Clause	0	1
Prepositional Phrase	1	0

(16b) Return the OFF/STBY/TALK switch to STBY *after your call.* (code)

(16c) The phone will ring *only if the handset is on the base.* (code)

(16d) In the STBY (standby) position, the phone will ring *whether the handset is on the base or in another location.* (code)

Example (16a) could go either way, except that it is the first sentence in a section titled "Battery Low Indicator", making the discussion of the indicator the local topic of conversation, and thus the appropriate theme of the sentence. Example (16b) is the example of rhetorical demotion we have discussed before. The action is considered obvious and this is demoted to phrase status and put at the end of the following action. Examples (16c) and (16d) show preconditions that are not fronted because of the logical nature of the precondition. In (16c), the condition is an exclusive condition which is never fronted. Neither is the condition in (16d) which lists all the possible alternatives, making itself automatically true.

The remainder of the precondition system network is divided among two tasks, those of determining the appropriate linker and the appropriate form. The linker is chosen by the **Precondition-Linker** sub-network, included in Fig. 4, and the form by the **Precondition-Form** sub-network, to be discussed below. Before either of these two sub-networks are reached, the **Precond-Significance** system is entered. Highly significant preconditions are usually expressed as separate sentences to give them special status. Consider the following examples:

(17a) TONE — For touchtone systems. *Tone signaling service (from your phone company) is required for TONE features.* (code)

(17b) ?? TONE — For touchtone systems *if you have tone signaling service from your phone company.*

(18a) The PAGE [11] button lets you signal from the base to the handset. *The OFF/STBY/TALK [8] switch, on the handset, must be in STBY position for paging.* (code)

(18b) ?? The PAGE [11] button lets you signal from the base to the handset *if the OFF/STBY/TALK [8] switch, on the handset, is in STBY position.*

In (17a), the precondition that tone signalling service be available is significant enough to warrant a separate clause. It would have been possible to express this information as in (17b), but this does not highlight the precondition as much. Note that this precondition is not fronted because of the contrastiveness of the concept "TONE" which is an element in a list of possible positions for the PULSE/TONE switch. Similarly, in (18a), the precondition is given a higher status by its expression as a sentence. It

would be possible to express the information as in (18b), but not with as high a significance. The slot on this example as well must be after the action, again because of contrastiveness.

If the precondition is not deemed significant, then the **Precondition Linker** and **Precondition Form** sub-networks are entered. In general, determine the linker for the precondition and the form of the precondition respectively although the **Precondition Form** sub-network does override the choice of the linker made by **Precondition Linker**.

3.3.2 Precondition Linker

The **Precondition-Linker** sub-network, also included in Fig. 4, determines what linker to use for the precondition. The **Precond-Logical-Form** system determines the logical nature of the preconditions and sets the linker accordingly. The three possible forms are conditional, changeable, and neither of the two. If the precondition is conditional, "if", "only if", or "whether ...or ..." are chosen as appropriate. Here are some examples of these:

(19a) The phone will ring *only if the handset is in the base.* (code)

(19b) *If you have touch-tone service,* move the TONE/PULSE SWITCH to the Tone position. (exc)

(19c) In the STBY (standby) position, the telephone will ring *whether the handset is on the base or in another location.* (code)

Example (19a) is an exclusive condition, that is, the phone will ring if the precondition is true, and will not ring if it is false. Note that the **Precondition-Slot** sub-network makes this same inquiry. The slot of such preconditions is always after their actions. Example (19b) contains the standard "if" condition. Example (19c) contains a complex precondition where the multiple alternatives cover all the logically possible values of the condition. This type of precondition is always true and is never fronted by the **Precondition-Slot** sub-network.

Preconditions which the user can effectively change, which we call "changeable" precondition, come at the beginning of sections that are not part of a sequence of prescribed actions. Here is an example:

(20) *1. Make sure the handset and base antennas are fully extended.* 2. Set the OFF/STBY/TALK SWITCH to Talk. (exc)

The reader is expected to be able to determine if the precondition is true, and if it is not, to perform an action to ensure that it is. In (20), the reader will check the antennas and extend them if they are not already extended. This type of precondition is expressed as a "Make sure" imperative clause.

If the precondition is neither conditional nor changeable, it is linked with "when." This commonly occurs when the precondition is part of an ongoing sequence of actions prescribed by the instructions and constitutes the bulk of the preconditions in the corpus. Here is an example:

(21) Lift the handset and set the OFF/STBY/TALK [8] switch to TALK. *When you hear dial tone,* dial the number on the Dialpad [4]. (code)

The precondition in (21) occurs in a sequence of actions and thus is not conditional or changeable.

3.3.3 Precondition Form

As we noted above, preconditions can be expressed as either a terminating condition or an action. The choice between the two is made by the **Precondition Form** sub-network, the higher level part of which is included in Fig. 4, and the part dealing with effective actions is shown in Fig. 6. This choice depends largely upon the type of action on which the precondition is based. Our corpus has lead us to a taxonomy of action types that engender different forms of expression. This taxonomy is shown in Fig. 5. The point of using this taxonomy is that the form of the expression of a preconditions is dependent upon the type of act being expressed, which can be determined from the representation of the process. We'll now talk about each of these action types in turn.

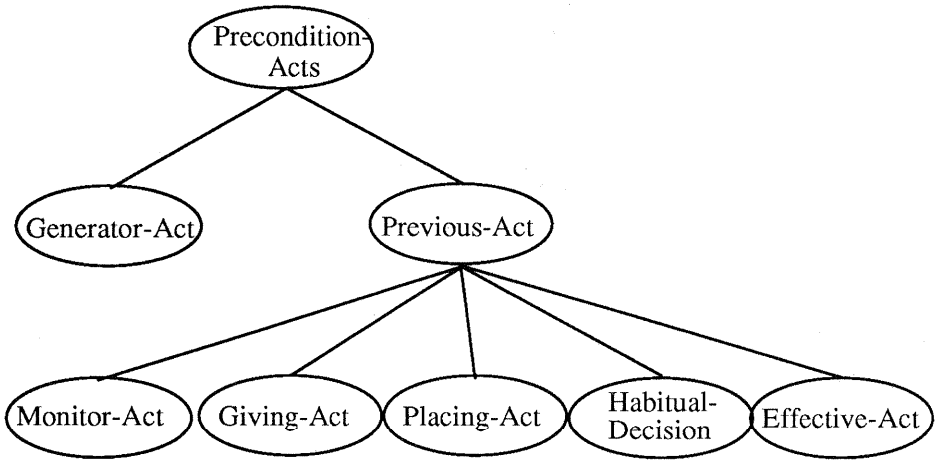


Figure 5: The Taxonomy of Action Types

Generator Actions. A generator action states a condition that is concurrent with the expressed action. We've named this type after the "generation" relation identified by Goldman (1970) and discussed by others (Di Eugenio, 1990; Balkanski, 1991). This type of action is typically expressed, in our corpus, in action form rather than in terminating condition form because it is, in some sense, concurrent with the action it is a precondition for. Here are some examples:

(22a) *When receiving a call,* the handset will ring. (exc)

(22b) *With TALK,* the 7010 works like an ordinary phone — no need to move the switch to answer or make calls. (code)

(22c) You can store the pause into Redial memory *during manual dialing.* (code)

Example (22a) is a non-contrastive precondition and is thus expressed using a participial form. Here, the act of receiving a call is in some sense the same as the act of the phone ringing, very much like Goldman's example of the light going on being the same as flipping the light switch. The linker "when" is chosen by the **Precondition Linker** sub-network.

If the generator precondition is contrastive with previous topics in the text, it is demoted to phrase status. If a nominalization exists, as in (22b), the form "with <nominalization>" will be used. If no nominalization exists, as in (22c), the form "during <ing-form>" will be used.

Previous Actions. There are several types of previous actions, that is actions which are neither concurrent nor generators, which we break into two groups: Effective actions, which will be discussed immediately, and the set of non-effective actions, which will be discussed below.

The bulk of the preconditions in our corpus are based on effective actions, that is some previously performed material action. The **Effective-Act** sub-system, shown in Fig. 6, is designed to determine the appropriate form for them. Here are some examples:

(23a) *1. Make sure the handset and base antennas are fully extended.* 2. Set the OFF/STBY/TALK SWITCH to Talk. (exc)

(23b) Lift the handset and set the OFF/STBY/TALK [8] switch to TALK. Return the OFF/STBY/TALK switch to STBY *after your call.* (code)

(23c) *When the 7010 is installed* and *the battery has charged for twelve hours,* move the OFF/STBY/TALK [8] switch to STBY. (code)

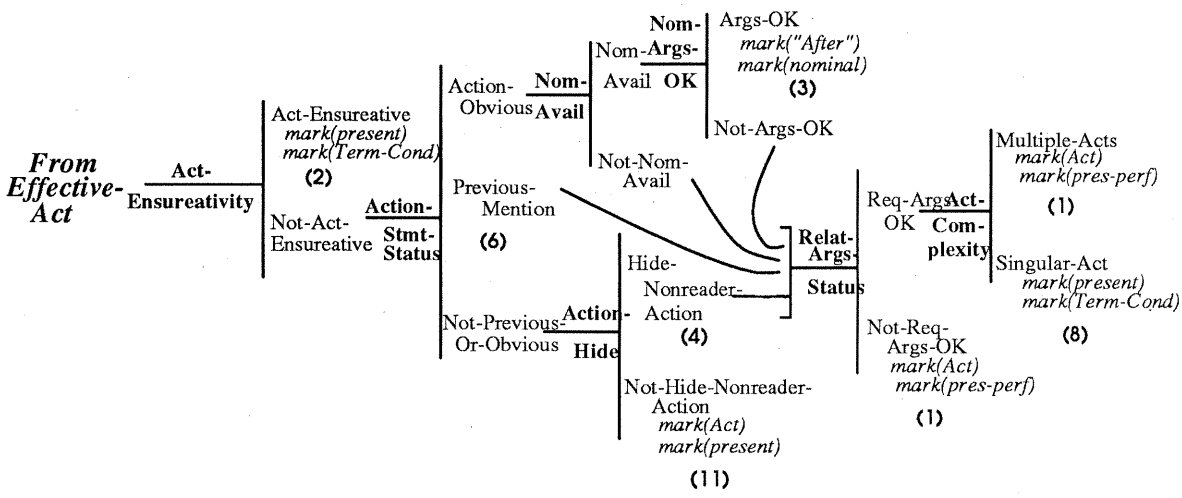


Figure 6: The Effective Action Portion of the Precondition-Form Sub-Network

(23d) *If you make a dialing error,* or want to make another call immediately, FLASH gives you new dial tone without moving the OFF/STBY/TALK switch. (code)

(23e) *After you have discharged and recharged the battery:* 1. Plug one end of telephone line cord into the base. 2. Plug the other end into the modular wall jack. (exc)

(23f) The BATTERY LOW INDICATOR will light *when the battery is the handset is low.* (exc)

The **Act-Ensurativity** system determines if the action is of the form "if the condition is true, then fine, otherwise do something to make it true." In (23a) the reader is being instructed to check the antennas, and if they are extended then go on, otherwise extend them and go on. The **Precondition Linker** sub-network will choose the linker "make sure that" and the **act-ensurativity** system will choose the terminating condition form of the action in present tense.

If the action is not ensurative, then the **Action-Stmt-Status** is entered. Here, if the act is considered obvious, the nominalized form is used if it exists. In (23b), the call itself is considered obvious and is thus rhetorically demoted. This is a case where the **Precondition Form** sub-network overrides the linker chosen by the **Precondition Linker** sub-network.

If the action has been mentioned previously in the text, then the **Action-Stmt-Status** system chooses the terminating condition form of the action in present tense. Example (23c) shows such a case.

If the action is neither obvious nor previously mentioned, the **Action-Hide** system is entered. This system determines if the agent of the action in something other than the reader and if that action should be hidden from the reader. This is common with non-reader actions that are too complex to explain. If this is not the case, the system chooses the act form of the action in present tense as in (23d).

If the action is a non-reader action that should be hidden, or if there was some problem with the nominalized form of an obvious action, the **Relat-Args-Status** system is entered. These cases all require that the action be hidden from the reader, making the terminating condition form of the action most appropriate. This form only works, however, if the arguments to the action that we must express can be stated in the relational form. If this is not the case, the present-perfect act form of the action is chosen, as in the second precondition in (23c). In this example, we would have preferred to say "?? When the 7010 is installed and the battery is charged, ..." but then would not have been able to mention the the 12 hours, a piece of information which was deemed important.

If the required arguments are possible with the relational form, then the **Act-Complexity** system is entered. If the precondition concerns two actions performed on the same object, the act form in present

perfect tense is used, as in (23e). The alternate form "When the battery is discharged and recharged." seems to imply that both states can be true and once which is not the case here. Another form "When the battery is discharged and the battery is recharged." seems to imply the same thing. If the action is singular, then the terminating condition form is used in present tense as in (23f).

There are several other types of actions, lumped together under Other-Acts in the **Prev-Act-Type** system. They are the monitor, giving, placing, actions and habitual decisions. Here are examples of them:

(24a) *When you hear dial tone,* dial the number on the Dialpad [4]. (code)

(24b) *If you have touch-tone service,* move the TONE/PULSE SWITCH to the Tone position. (exc)

(24c) The phone will ring *only if the handset is on the base.* (code)

(24d) *If you leave the OFF/STBY/TALK [8] switch in TALK,* move the switch to PULSE, and tap FLASH [6] the next time you lift the handset, to return to PULSE dialing mode. (code)

Monitor actions, as shown in (24a), concern the monitoring of some conditions in the environment. Our corpus only contained listening actions whose terminating condition form is like that of (24a). This type of action was never expressed, in our corpus, in action form as the termination is all that is relevant. Action form would be something like "?? *When you have listened for the dial tone,* do ..." which seems to imply that simply listening was all that was important. Monitor actions, if expressed as actions, take the form of imperative commands such as "Listen for the dial tone."

Giving and placing actions, similarly, are expressed as terminating conditions as in (24b) and (24c). It would be unusual to see something like "?? *If the phone company has given you touch-tone service,* do ..." or "?? Do ... *if you have placed the handset on the base.*" We have not seen these forms in our corpus.

Habitual decisions is our term for the decision to make an habitual practice of some action. When stated as preconditions, they take the form in (24d). This expression refers not to a singular action of leaving the OFF/STBY/TALK switch in TALK position, but rather to the decision to habitually leave it so. The singular event would be expressed as "*If you have left the OFF/STBY/TALK switch in TALK,* do ..." which means something quite different from the expression in (24d) which is stated in present tense.

Because monitor, giving, and placing actions, as well as habitual decisions are stated, in our corpus, as present tense, terminating conditions, we have lumped them together under the choice Other-Previous-Act in the **Precond-Act-Type** system. The particular details of their expression should be derivable from the representation of the process itself.

3.3.4 Coverage of the Precondition Expressions in the Corpus

The precondition network, as described, explains 45 of the 61 precondition expressions in the corpus (74%). Because preconditions are more complex than either purposes or results it is not surprising that there are more complex problems with the precondition network. We'll discuss the problems with slot, linker, and then form.

Slot Problems. The one problem with slot determination comes in the following example:

(25) 1. Return the handset to the cradle with the handset set to Stby or Talk. Your EXCURSION will disconnect *when the handset is in the base.* (exc)

In this case, our network would have fronted the precondition. We suspect the fact that it has not been fronted in the corpus has to do with its relationship with a result. Perhaps being a precondition for a result has forced it to the end of the sentence. We will need more confirming examples before reflecting this in the system network.

Linker Problems. There are four relatively minor problems with determining the linker for preconditions. Two of them relate to the use in the corpus of the unusual linkers "By the time" rather than "When" and "Should" rather than "If." We will not incorporate them into the networks unless we see more examples of each and can determine a functional distinction. There are also two uses of "After" where we would produce "When." Typically the choice between the two is based on the possible concurrency of the precondition and the related action. If there is possible concurrency, "After" is preferred, otherwise, "When" is used. There is no possible concurrency in either case in the corpus because both are stating terminating conditions. This claim is related to a clause combining claim concerning"and" and "then." "Do action A *and* do action B" is used when A and B cannot be done concurrently (or are supposed to be done concurrently), while "Do action A *then* do action B" is used to forcibly order possibly concurrent actions in time.

Form Problems. There are nine problems related to the form of precondition expressions. We will not discuss each one, but will mention three that seem of particular interest. Consider these three examples of preconditions from the corpus:

(26a) 1. Return the handset to the cradle *with the handset set to Stby or Talk.* (exc)

(26b) This will disconnect the line and enable your telephone to ring *when an incoming call is received.* (exc)

(26c) *If you keep both calls active,* your phone line is busy. (code)

Preconditions can be stated as relative clauses, as in (26a). Unfortunately, the current corpus contains only one such case leaving us unable to make any concrete statement about the functional motivation for using such a construction. We will address this if we find more occurrences of it.

There were two examples of passive constructions in preconditions, as in (26b). Clearly the passive allows the writer to focus on an argument that is not the subject, or perhaps avoid the mention of the subject, but it is unclear in (26b) that this is what is happening. The subject "you" is not strictly avoided in this manual and is, in fact, mentioned earlier in the paragraph. We will need more than two examples to base any concrete statement as to when these constructions are used.

Finally, the generator precondition in (26c) is not in the form the system network is designed to handle. The current system network either produces "With <nominalization>" or "During <ing-form>", neither of which would work in this context. It seems that the nature of the generator condition requires a more flexible construction such as the clause form in (26c), but, again, we will need more examples on which to base a solution.

3.4 Result Relations in Instructional Text

Instructional text deals with actions and very often includes some statement of the results of the actions it prescribes. Our study of these clauses has resulted in the network shown in Fig. 7. Here are some representative examples:

(27a) When the 7010 is installed and the battery has charged for twelve hours, move the OFF/STBY/TALK [8] switch to STBY. *The 7010 is now ready to use.* (code)

(27b) 3. Place the handset in the base. *The BATTERY CHARGE INDICATOR will light.* (exc)

Result relations are typically expressed after the action they pertain to as seen in (27) so the result network exclusively addresses the form of the result. The frequency with which these slots and forms appeared in the 11 result clauses in our corpus is given in Table 3.

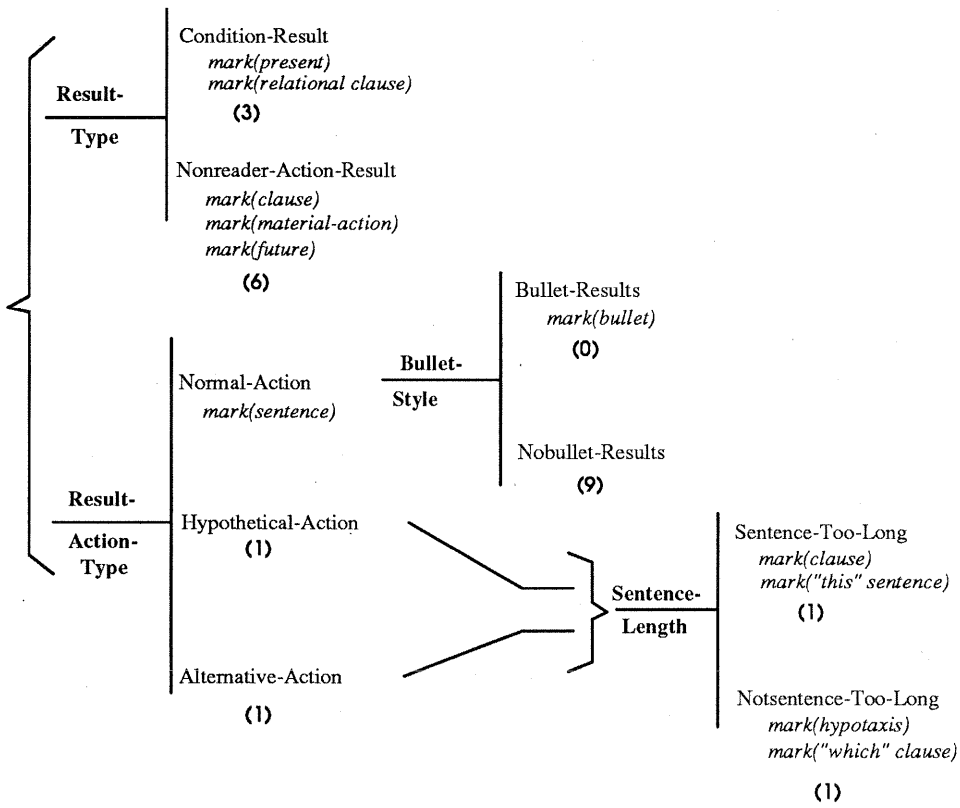


Figure 7: The Result System Network

Table 3: The frequency of various form and slot combinations of results in instructional text

	Initial	Final
Present/Relational	0	3
Future/Action	0	7
Present/Action	0	1

3.4.1 Result Form

The form tends to be either a present, relational clause as in (27a) or a future, material action clause as in (27b), corresponding to results which are conditions and results which are non-reader actions respectively. The non-reader actions are marked as irrealis by the future tense presumably to keep readers from thinking they must perform the action.

The **Result-Action-Type** system determines what type of action the result pertains to. "Normal" actions are those actions which must always be performed by the reader in the course of the execution of the instructions. These are common enough that some manuals mark them explicitly with bullets or other typographical marks. This issue of style is handled by **Bullet-Style**. The current corpus contains no manual that chooses this style.

Hypothetical actions, those that may or may not happen to the reader, and Alternative actions, those that the reader may or may not choose to execute, are handled differently. Because they are not results that the reader can assume to be the case, they are not stated independently, but rather are tied grammatically to the actions that bring them about. Two examples of this from the corpus are:

(28a) If you do not wish to return the handset to the base, position the OFF/STBY/TALK SWITCH to Stby. *This will disconnect the line and enable your telephone to ring when an incoming call is received.* (exc)

(28b) When the handset is away from the base and not in use, the OFF/STBY TALK SWITCH should never be lift in the Talk position. Instead, it should be set to Stby, *which will allow the handset to ring.* (exc)

In these examples, the specified results are contingent upon "non-normal" actions, and are grammatically tied to the expression of those actions using "which" or "this" to refer to the action. This, evidently, makes it less likely that the reader will incorrectly expect them to be the case when the action has not been performed. The use of hypotaxis or separate clauses, determined by **Sentence-Length**, is based on the complexity of the clauses; in our corpus, the separate clause form was used if the action clause was already combined with more than one clause. For example, the action in (28a) is coupled with a precondition and thus cannot be combined with the result.

3.4.2 Coverage of the Result Expressions in the Corpus

The result system network, as described, explains 8 of the 11 result clauses in the corpus (72%). Of the three exceptions in the corpus, two were not separate sentences as the network would predict and one was not in future tense as predicted.

(29a) 5. After 24 hours, connect the telephone line cord *and your EXCURSION is ready to use.* (exc)

(29b) 3. Tap the REDIAL [7] button. *The last number you dialed (up to 32 digits) is automatically redialed.* (code)

Example (29a) is one of the two results that where not expressed as separate sentences. It is unclear what is gained with this expression over the form we would produce, "5. After 24 hours, connect the telephone line cord. *Your EXCURSION is now ready to use.*"

Example (29b) is the only non-reader action result expression that is not in future tense. It is similarly unclear what advantage this expression has over the one we would produce, "3. Tap the REDIAL [7] button. *The last number you dialed (up to 32 digits) will automatically be redialed.*" unless there is something related to the clause being passive.

3.5 Realization Statements for Text

The system networks we have developed employ realization statements for text. These statements are similar to Nigel's realization statements except that they operate on rhetorical structures. We have used four basic types: `insert`, `order`, `mark`, and `remove`.

`Iterative-Add` adds a particular text function to the rhetorical structure tree. This statement is similar to Nigel's `insert` except that it is iterative and the functions it inserts are subscripted. By iterative, we mean that it adds an arbitrary number of whatever function is indicated. We can use it to add an arbitrarily long list of procedural actions as children of a text node, or to add a list of preconditions to a particular action. By subscripted functions, we mean that it is capable of adding any number of nodes and subscripting them. This way we can indicate, for example, which purpose nodes go with which action nodes. We also include `Add`, a non-iterative version of this realization statement.

`Order` is very similar to Nigel's `order` realization statement. It allows us to textually order two functions in the rhetorical structure. The **Purpose-Slot** system, for example, uses the `order` realization statement (denoted by < and > in Fig. 3) to specify the slot of the purpose clause/phrase.

`Mark` adds an annotation to a node in the tree. These annotations are used to determine which Nigel inquiries should be preset before the call to Nigel for a particular clause/phrase. The **Purpose-Form** system, for example, marks purposes as "to" infinitives (tnf), gerunds, etc.

`Remove` allows us to remove a particular node or link from the rhetorical structure. In a rhetorical demotion, for example, we `remove` the rhetorical link signifying that a node is an action, and `insert` the link for the same node indicating that it is a precondition satellite of another action.

3.6 Information Sources for Inquiries

Responding to the inquiries for the purpose system network, as with all the other networks we have developed, requires consulting four different sources of information:

1. The Process Representation — This source represents the process that the writer wants to talk about. It can be used, for example, to respond to the **Purpose-scope** system in the purpose network which inquires about the number of actions the purpose in question pertains to.

2. The User Model — This source represents the information the user is expected to know and is used to respond to the inquiries from the structure building tools concerning the level of detail of the instructions.

3. The Lexicogrammar — This source represents the lexical and grammatical tools available to the system for expression. It can be used, for example, to respond to the **Purpose-nominal-Form** system in the purpose network which inquires about the availability of a nominalization for a particular action. This approach uses the lexicogrammar as a resource, as Bateman has proposed in (Bateman, 1991).

4. The Generated Text — This source represents the information related in the previous text. It can be used, for example, by the precondition network system **Action-Stmt-Status** to determine if an action related to a precondition has or has not been mentioned before. It is possible that the process representation itself would contain this information, but it is information of a different type.

5. The Style Parameters — This set of parameters represents the writing and formatting style of a particular technical writing shop. The result network's **bullet-style** system, for example, consults this set to determine whether result relations be explicitly bulleted or not. The identification of such parameters for instructional text should aid in the ongoing effort to model register variation (Paris and Bateman, 1990).

Currently, we are defining the inquiry interface and have not built the tools to run them in what Nigel refers to as *implemented* mode (Nigel, 1988). We feel that building this interface is an important first step in determining what a text generator needs from these sources; from this information we can go on to design the required knowledgebases in the appropriate way as discussed in (Paris and Maier, 1991).

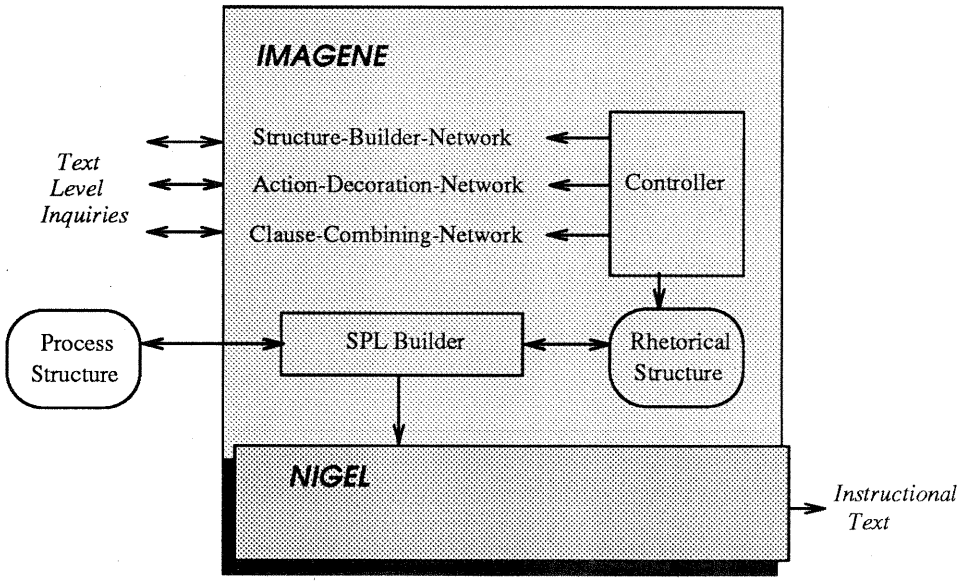


Figure 8: IMAGENE's Architecture

4 Implementation of the Results: IMAGENE

The results of the analysis just described, are being implemented in IMAGENE. IMAGENE presents an inquiry interface for generating texts, much like Penman does for generating sentences. Its architecture is depicted in Fig. 8.

IMAGENE's controller executes the three system networks depicted to produce a rhetorical structure[6]. The resulting rhetorical structure is represented using an implementation of RST in LOOM (MacGregor, 1989) which includes those aspects of RST necessary for instructional text including purpose, precondition, and result relations and facilities for recursive depth of procedural and functional lists. This structure is then traversed and an SPL (Penman, 1989) statement for for each expressible sentence is constructed and passed to Penman.

This architecture provides the flexibility to build a rhetorical structure and then pass back over it as many times as necessary making modifications. It also has the flexibility to use system networks or any other form of control structure on each pass depending upon the requirements of the task. Currently, only system networks are used, but it would be possible, for example, to include a spreading activation account of clause combining should that prove to be a more useful paradigm.

The remainder of this section outlines a sample run of the program, detailing the construction of the rhetorical structure, the attachment of local rhetorical relations to each of the actions in that structure (called *action decoration*), and the invocation of Penman for each collection of nodes constituting a sentence. The example text includes two actions, moving a switch and extending the antenna, and both a precondition and a result. We then conclude with more examples of IMAGENE's output.

4.1 The Structure Builder

The structure-building tool starts with a preliminary rhetorical structure of one root node, determines what type of children that node will have, and then iteratively adds those children to the structure using the `iterative-add` realization statement. It then recursively calls itself for each of the children using

[6] Although the architecture calls for a system network for clause combining, this task is currently done with a simple traversal mechanism. A more detailed analysis of clause combining is yet to come.

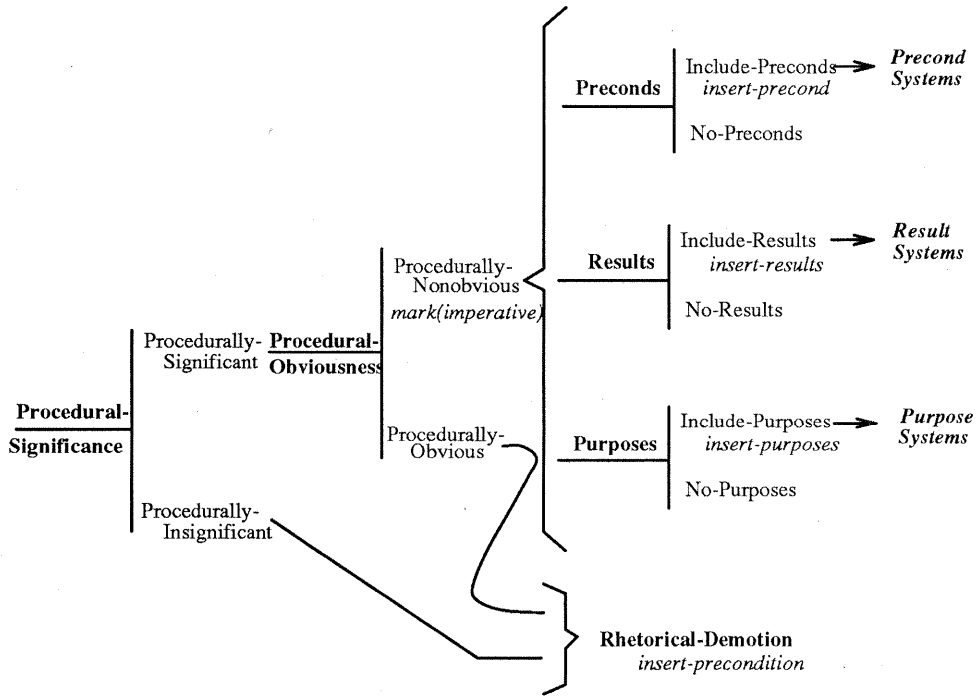


Figure 9: The Action Decorator

Nigel's `preselect` statement. This process results in a partial RST tree of arbitrary depth containing all temporally sequenced and temporally unsequenced children, called *procedural* and *functional* children respectively, of each node[7]. No nodes for preconditions, purposes, or results are included.

4.2 Action Decoration

IMAGENE then traverses the partial RST tree, calling the action-decoration network for each action at the leaf level. This network attaches various local rhetorical relations to the actions and performs some restructuring. Figure 9 shows the root of the action-decoration network.

The systems **procedural-significance** and **procedural-obviousness** specify that actions become imperative commands unless they are deemed procedurally insignificant or procedurally obvious, in which case they are rhetorically demoted and attached to the following action by the **rhetorical-demotion** gate. For actions that are significant and non-obvious, this network calls decoration sub-networks, such as the purpose network discussed above, to realize any desired preconditions, results, and purposes. Currently, the **Preconds**, **Results**, and **Purposes** systems simply inquire whether or not to include their respective relations rather than encoding the functional criteria for including such a relation. To date we have been more interested in the question of how to express a local rhetorical relation, rather than on when they should be expressed.

The result of this pass is a decorated rhetorical structure. This is the partial rhetorical structure built by the structure-builder, extended with preconditions, results, and purposes and modified by any rhetorical promotions and demotions. The structure for our example is depicted in Fig. 10. It includes a root node called `text-root1` which has two procedural children, `action1` and `action2`. `Action1` has both a precondition and a result. In this example the environment/user has mapped the functions `action1` and `action2` to the abstract actions of moving a switch and extending the antenna respectively,

[7] Currently, only procedural children are handled by IMAGENE.

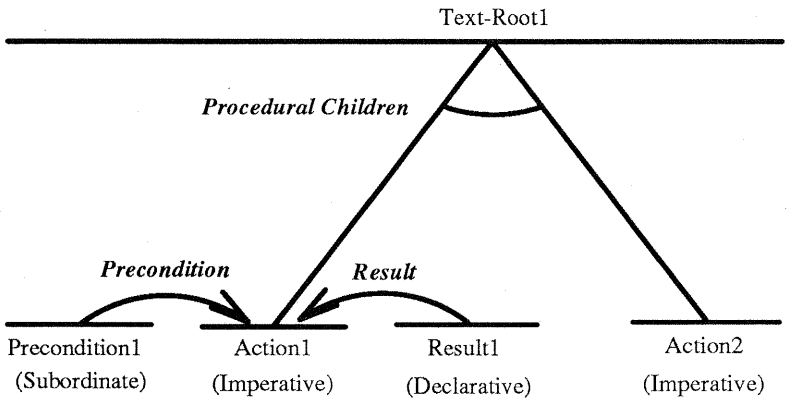


Figure 10: A Sample IMAGENE Rhetorical Structure

`precondition1` to having the battery charged and `result1` to having the phone ready. This structure with these mappings is reflected in the resulting text shown in the next section.

4.3 Calling Nigel

When the decorated rhetorical structure has been produced, IMAGENE traverses the tree again, calling Nigel with an SPL statement constructed for each set of leaf nodes that constitute a sentence. The construction of this SPL statement requires access to information from two sources, the rhetorical structure and the process structure, as can be seen in Fig. 8. The rhetorical structure, constructed by IMAGENE, indicates the form of the clauses/phrases (imperative, declarative, nominalizations, etc.), determines the order of the clauses/phrases, and chooses the conjunctions and prepositions. The process structure, constructed by hand, includes the name of the domain concepts corresponding to the predication involved and the arguments to this predication. This process structure is presumed to be similar to the output of an automated planner responsible for planning the process being described. This simplification allows us to concentrate our study on the expression of instructions, rather than on the execution of a real planner; see Mellish and Evans (1989) for a study of the problems involved in generating from the output of a planner. IMAGENE's output for our running example is:

> When the battery has been charged for eight hours, move the OFF/STBY/TALK switch to STBY. The phone is now ready to use. Extend the base antenna.

This includes the two actions "move" and "extend", the precondition that the battery has been charged, and the result the phone is ready to use. We did not specify any functional information that would dictate any rhetorical promotions or demotions.

4.4 More Examples

Examples of rhetorical demotion and promotion will now be discussed. As a context for these examples, consider IMAGENE's expression of the two actions, charging and moving:

> Charge the battery for eight hours. Move the OFF/STBY/TALK switch to STBY.

Rhetorical demotion, as discussed above, is the process of demoting an action to precondition status and tacking it on the end of the following action. This is typically done when an action is considered obvious to the reader. If, in the given example, IMAGENE is told that the charging of the battery will be obvious to the reader, it produces:

Move the OFF/STBY/TALK switch to STBY, after the battery has been charged for eight hours[8].

If, on the other hand, IMAGENE is told that the charging of the battery is a precondition and that this precondition is considered especially significant, it produces:

Make sure that the battery has been charged for eight hours. Move the OFF/STBY/TALK switch to STBY.

These examples show the sorts of detailed variation of expression of local rhetorical relations that IMAGENE provides.

5 Conclusion

We have discussed IMAGENE, a text generator that uses system networks to translate process structures into grammatically annotated rhetorical structures. We have shown how these system networks are capable of capturing the functional distinctions we found in our analysis of a corpus of instructional texts and have noted the requirements imposed by this analysis on the knowledgebases supporting text generation. Our plans for this work include: the inclusion of other forms of instructions in the corpus; the development of a process representation and user model which include the information we have identified as important; an investigation of the high level organization of instructional text; an analysis of lexical selection in instructional text including the use of non-literal language and referring expressions; and a study of when and why instructional texts include the local rhetorical relations we have been working with.

[8] When the system network for preconditions is fully implemented, IMAGENE will use a nominalization, if one exists, for this situation.

References

Allen, J. F., Kautz, H. A., Pelavin, R. N., and Tenenberg, J. D. (1991). *Reasoning About Plans.* Morgan Kaufmann, San Mateo, CA.

Alterman, R., Zito-Wolf, R., Carpenter, T., Goodman, M., and Waterman, S. (1991). Readings from the FLOABN project. Technical Report CS-91-157, Brandeis University. Volume I.

Balkanski, C. T. (1991). Logical form of complex sentences in task-oriented dialogues. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics - Student Session,* pages 331–332. Berkeley, CA.

Bateman, J. A. (1991). Decision making in text generation: Towards a negative definition. In Meteer, M. and Zukerman, I., editors, *Proceedings of the IJCAI-91 Workshop on Decision Making Throughout the Generation Process,* August 24–25, Darling Harbor, Sydney, Australia.

Beaman, K. (1984). Coordination and subordination revisited: Syntactic complexity in spoken and written narrative discourse. In Tannen, D., editor, *Coherence in Spoken and Written Discourse,* pages 45–80. Ablex, Norwood, NJ.

Biber, D. (1988). *Variation Across Speech and Writing.* Cambridge University Press, Cambridge.

Cawsey, A. (1990). Generating explanatory discourse. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation,* chapter 4. Academic Press. Selected readings from the 2nd European NLG Workshop, April, 1989, Edinburgh.

Chafe, W. L. (1982). Integration and involvement in speaking, writing and oral literature. In Tannen, D., editor, *Spoken and Written Language,* pages 35–53. Ablex, Norwood, NJ.

Code-a-phone (1989). *Code-A-Phone Owner's Guide.* Code-A-Phone Corporation, P.O. Box 5678, Portland, OR 97228.

Cumming, S. (1984). Local cohesion in Chinese and English: an approach to clause combining. In *Proceedings of the Tenth Annual Meeting of the Berkeley Linguistics Society,* pages 465–471.

Cumming, S. (1990). Natural discourse hypothesis engine. In McKeown, K. and an Sergei Nirenburg, J. M., editors, *Proceedings of the Fifth International Workshop on Natural Language Generation,* Junc 3–6, Dawson, PA.

Cumming, S. (1991). Nominalization in English and the organization of grammars. In *Proceedings of the IJCAI-91 Workshop on Decision Making Throughout the Generation Process,* August 24–25, Darling Harbor, Sydney, Australia.

Dale, R. (1988). The generation of subsequent referring expressions in structured discourses. In Zock, M. and Sabah, G., editors, *Advances in Natural Language Generation - An Interdisciplinary Perspective,* volume 2, chapter 4. Ablex.

Dale, R. (1990). Generating recipes: An overview of Epicure. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation,* chapter 9. Academic Press. Selected readings from the 2nd European NLG Workshop, April, 1989, Edinburgh.

Di Eugenio, B. (August 2, 1990). A language for representing action descriptions. Preliminary Thesis Proposal, University of Pennsylvania.

Excursion (1989). *Excursion 3100.* Northwestern Bell Phones, A USWest Company.

Foley, J., Kim, W., Kovacevic, S., and Murray, K. (1991). UIDE — An intelligent user interface design environment. In Sullivan, J. W. and Tyler, S. W., editors, *Intelligent User Interfaces,* chapter 15. Addison-Wesley.

Ford, C. (1986). Overlapping relations in text structure. In *Proceedings of the Second Annual Meeting of the Pacific Linguistics Society.*

Ford, C. (1988). *Grammar in ordinary interaction: the pragmatics of adverbial clauses in conversational English.* PhD thesis, UCLA.

Goldman, A. I. (1970). *A Theory of Human Action.* Prentice Hall.

Gruber, T. R. (1990). Model-based explanation of design rationale. Technical Report KSL 90-33, Knowledge Systems Laboratory, Stanford University.

Haiman, J. and Thompson, S., editors (1989). *Clause combining in grammar and discourse.* Benjamins, Amsterdam.

Halliday, M. A. K. (1976). *System and Function in Language.* Oxford University Press, London. Ed. G. R. Kress.

Halliday, M. A. K. (1985). *An Introduction to Functional Grammar.* Edward Arnold, London.

Hopper, P. J. and Thompson, S. A. (1984). The discourse basis for lexical categories in universal grammar. *Language*, 60(4):703–752.

Hovy, E. H. (1988). Planning coherent multisentential text. In *Proceedings of the Twenty-Sixth Annual Meeting of the Association for Computational Linguistics.* State University of New York, Buffalo, NY, USA, June 7-10.

Hovy, E. H. (1989). Approaches to the planning of coherent text. Technical Report ISI/RR-89-245, USC Information Sciences Institute.

Hovy, E. H. and McCoy, K. F. (1989). Focusing your RST: A step toward generating coherent multisentential text. In *Program of the Eleventh Annual Conference of the Cognitive Science Society,* August 16–17, Ann Arbor, MI, pages 667–674. Lawrence Erlbaum Association, Publishers.

Hudson, R. A. (1971). *English Complex Sentences – An introduction to systemic grammar.* North–Holland, Amsterdam.

MacGregor, R. (1989). *The LOOM Users Manual.* USC Information Sciences Institute. Draft.

Mann, W. C. (1985). An introduction to the Nigel text generation grammar. In Benson, J. D., Freedle, R. O., and Greaves, W. S., editors, *Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th International Systemic Workshop,* volume 1, pages 84–95. Ablex.

Mann, W. C. and Thompson, S. A. (1987a). Rhetorical structure theory: a framework for the analysis of texts. *IPRA Papers in Pragmatics*, 1. Also published as USC tech report ISI/RS-87-185.

Mann, W. C. and Thompson, S. A. (1987b). Rhetorical structure theory: Description and construction of text structures. In Kempen, G., editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics.* NATO Scientific Affairs Division, Martinus Nijhoff. Proceedings of the NATO Advanced Research Workshop on Natural Language Generation, Nijmegen, The Netherlands, August 19-23, 1986.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: A theory of text organization. In Polanyi, L., editor, *The Structure of Discourse.* Ablex.

Matthiessen, C. M. I. M. and Thompson, S. A. (1987). The structure of discourse and "subordination". In Haiman, J. and Thompson, S., editors, *Clause Combining in Discourse and Grammar.* Benjamins, Amsterdam. Also available as ISI tech. report ISI/RS-87-183.

McKeown, K. R., Elhadad, M., Fukumoto, Y., Lim, J., Lombardi, C., Robin, J., and Smadja, F. (1990). Natural language generation in COMET. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, chapter 5. Academic Press.

Mellish, C. and Evans, R. (1989). Natural language generation from plans. *Computational Linguistics*, 15(4):233–249.

Moore, J. D. and Paris, C. L. (1988). Constructing coherent text using rhetorical relations. Submitted to the Tenth Annual Conference of the Cognitive Science Society, August 17–19, Montreal, Quebec.

Nigel (1988). *The Nigel Manual.* USC Information Sciences Institute, Penman Natural Language Group. Draft.

Paris, C. L. and Bateman, J. A. (1990). User modeling and register theory: A congruence of concerns. Draft – Submitted to Journal of User Modeling.

Paris, C. L. and Maier, E. (1991). Knowledge resources or decisions. In *Proceedings of the IJCAI-91 Workshop on Decision Making Throughout the Generation Process,* August 24–25, Darling Harbor, Sydney, Australia.

Penman (1989). *The Penman User Guide.* USC Information Sciences Institute, Penman Natural Language Group, 2nd edition.

Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A Comprehensive Grammar of the English Language.* Longman.

Sanders, T. J. M., Spooren, W. P. M. S., and Noordman, L. G. M. (1990). Towards a taxonomy of coherence relations. Technical report, Discourse Studies Group, Department of Language and Literature, Tilburg University, The Netherlands.

Scott, D. R. and Souza, C. (1990). Getting the message across in RST-based text generation. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, chapter 3. Academic Press.

Shopen, T., editor (1985). *Language Typology and Syntactic Description.* Cambridge University Press, Cambridge, England.

Strunk Jr., W. and White, E. B. (1979). *The Elements of Style.* McMillan, New York, third edition.

Thompson, S. A. (1985). Grammar and written discourse: Initial and final purpose clauses in English. *Text*, 5(1–2):55–84.

Thompson, S. A. (1987). "Subordination" and narrative event structure. In Tomlin, R. S., editor, *Coherence and Grounding in Discourse*, pages 435–454. John Benjamins Publishing Co. Outcome of a Symposium, Eugene, Oregon, June, 1984, and published as vol. 11 of the series Typological Studies in Language.

Thompson, S. A. and Mann, W. C. (1987). Antithesis: a study of clause combining and discourse structure. In Steele, R. and Threadgold, T., editors, *Language Topics: Essays in honour of Michael Halliday*, volume II, pages 359–381. Benjamins, Amsterdam. Also published as USC tech report ISI/RS-87-177.

Vander Linden, K., Cumming, S., and Martin, J. (1992). Using system networks to build rhetorical structures. In Dale, R., editor, *Collected Papers of Proceedings of the Sixth International Workshop on Natural Language Generation,* Trento, Italy, 5–7 April 1992. Springer Verlag. To appear.

Winograd, T. (1972). *Understanding Natural Language.* Academic Press, New York.