A Relational Database Tool for Detailed Corpus Analysis

Keith Vander Linden , Susanna Cumming

CU-CS-599-92 June 1992

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGMENTS SECTION.

A Relational Database Tool for Detailed Corpus Analysis[1]

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

[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

June 1992

Abstract

A relational database tool for the representation and analysis of linguistic corpora is described. It has the capability of representing syntactic and semantic information at the clause level, the argument level, and the rhetorical level. The categories that it encodes, how they are encoded, and some examples of how they are used are given. The platforms on which it has been implemented are also discussed.

[1] This work was supported in part by NSF Grant IRI-9109859.

Contents

List of Figures

List of Tables

1 Introduction

The range of grammatical coverage offered by current tactical text generators allows us considerable flexibility in the forms we may choose to express a given concept. This flexibility, however, forces us to face the problem of choice in generation. Which of the many different forms that express the same basic idea should we use to express this concept in this context? Nowhere is this problem more evident than in discourse generation, where we face the choice problem at the discourse level as well as at the sentence level.

A typical approach to this problem is to let our intuitions do the talking, that is, continue to adjust the text generator until the output sounds good to us. The resulting text is good, but only to a first approximation. Discerning the nature of effective expression requires that we move beyond our intuitions and look at real text. The forms of expression that are used in real text are typically the ones found most useful in expressing a concept. Employing this corpus-based method requires that we gather a set of examples of the text type we are interested in and carefully read through them, looking for consistent patterns of use. This approach is becoming more common and lends a credible reality check to the text we generate. Anyone who has done this, however, will realize that mapping functions to forms in a large corpus is a daunting task. Some level of automation is required to ease the pain of this type of research.

An effective method of automation that has come out of functional linguistic research is to encode a small text corpus in a relational style database, encoding the various categories of information that we find useful in text analysis. To date, these systems have received little if any attention in the computational linguistics literature. This needs to change. If some standard of representation could be established, the arduous task of encoding information on a particular text type in the first place could be performed once and the finished product distributed for the benefit of all. Clearly a definitive standard of representation is unattainable due to the variety of possible research questions, but the relational database model provides considerable flexibility for taking a baseline representation and tailoring it to suit specific research goals. This report is a first attempt to define such a baseline.

The type of corpus analysis advocated here and for which our database tools have been built differs from the large scale text corpora work done on data available from the ACL data collection initiative (DCI) and from the even larger scale statistical analyses. We are advocating the detailed analysis of a relatively small corpus where the analyst has total control of the analysis process, as advocated in Cumming (1990).

2 Levels of Representation

The database we have built represents information on three different levels, the rhetorical, clausal, and argument levels. Figure 1 shows the relations that are maintained between the three levels. The Clause table maintains a record for each clause in the database which, among other information, references records in the Argument table containing information on its subject, direct object, and indirect object. The Argument table also includes NP modifiers such as possessors and prepositional phrases. The RST table contains a record for each node in an RST analysis of the text, containing fields referring to the nucleus and the satellite of the relation.

The following sections detail the nature of each of the fields in the tables, specifying the valid range of values. In may cases prespecifying such a range of values is important for effective querying of the database.

2.1 The Clause

The Clause table stores both syntactic and semantic information concerning each clause in the text. One record is stored for each clause.

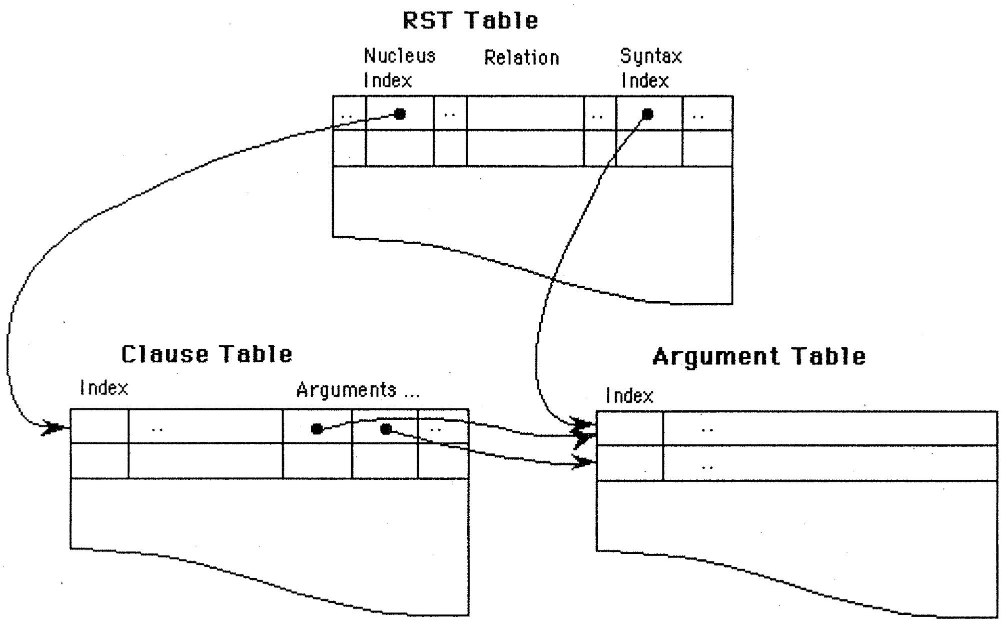


Figure 1: The relations between the three levels

2.1.1 The Fields

This section details the contents and use of each field. The primary key for the table is the combination of SOURCE and INDEX. A quick reference to all the fields in the Clause table can be found in table 1. A more detailed discussion of each field and its possible values follows directly.

SOURCE is a 3 character string coding the particular source that the clause is from. Including it allows INDEX to be an index to clauses, starting from 1 for each source. This allows the text from the various sources to be removed and edited without affecting the clause numberings for the other sources.

INDEX is the number of the particular clause. This number must allow a fractional component in order to allow the addition of new clauses in a database that has already been built. It is fairly common to come back to a clause and decide that it is really two separate clauses.

PAGE is simply the page number in the source text on which the clause in question starts. It is helpful in finding the clause in the original text and potentially also in identifying page breaks in the text.

LINKER is the lexical item used in the clause to link with another clause in another record. For coordinating clause linkages, the linker is stored with the second clause as the first lexical item in the raw text, and as its value for LINKER. Subordinating linkers are stored with the subordinate clause in the same way.

ADVERB simply lists any adverb that is employed in the text.

TENSE represents the tense used by the clause. The range of possible values is represented in table 2. The value "-" is used when there are incomplete sentences in the source that must be represented somewhere. We don't represent these incomplete sentences as noun phrases or prepositional phrases because they are not arguments to any predication or NP, and we want to be able to determine the order of the text from the Clause table alone.

Clause Quick Reference Table

Field Name	*Field Type*	*Field Size*	*Description*
SOURCE	Character	003	A code indicating the source text of the clause.
INDEX	Number	008	The clause number within the source text.
PAGE	Number	005	The page number where the example can be found.
LINKER	Character	012	Any clause linker used for this clause.
ADVERB	Character	012	Any adverb used.
TENSE	Character	003	The Tense of the clause.
MODAL	Character	010	Any modal used.
HAVE_EN	Character	001	Whether or not the predicate has the Have–En form.
BE_ING	Character	001	Same for Be–Ing.
BE_EN	Character	001	Same for Be–En
VERBSTEM	Character	012	The stem of the main verb.
POLARITY	Character	001	The polarity of the predicate.
SEMCLASS	Character	006	The semantic class of the predication.
REFERENCE	Character	012	The predication referred to by the predicate (if any).
LOOKBACK	Number	005	The distance looked back by the reference (if any).
NOUNHEAD	Number	005	The index of the head NP for relative clauses.
CLAUSEHEAD	Number	005	The index of the head clause for hypo and parataxis.
ROLE	Character	006	The semantic role played by the predication.
SLOT	Character	001	The slot of the clause (if a relative or conjoined clause).
ARG_1	Number	005	The argument number of the subject of the clause.
ARG_2	Number	005	The argument number of the direct object of the clause.
ARG_3	Number	005	The argument number of the indirect object of the clause.
TAG	Character	001	A field used to mark questions or problems.
TEXT	Character	254	The raw text of the clause.
ABSNUM	Number	005	A utility field used for lookback computations.

Table 1: Quick reference of Clause table field types

Clause Tense Table

Possible Value	*Description*
ing	-ing form
ppt	past participle
prs	present participle
pre	simple present
pst	simple past
fut	simple future
tnf	to infinitive
imp	imperative
nom	nominalization
-	no verb, no nominalization

Table 2: The possible values for TENSE in the Clause table

Clause Semantic Class Table

Possible Value	*Description*
MAT	Material (physical) action
SPEECH	Verbal action
MENT	Mental action - cognition etc.
RELAT	Relational – copulas etc.
EXIST	Existential – there be
?	Don't know

Table 3: The possible values for SEMCLASS in the Clause table

MODAL represents any modal verbal elements used in the clause.

HAVE_EN, BE_ING, and BE_EN are fields having values of "y" or "n". They code the aspect and voice systems of English. HAVE_EN indicates perfective aspect, BE_ING, progressive aspect, and BE_EN indicates the passive. These three issues can occur singly or in groups in the English verb sequence and are thus represented separately.

VERBSTEM represents the base form of the main verb used in the clause.

POLARITY indicates the polarity of the clause. The range of possible values is "A" for affirmative, "N" for negative, and "?" for unknown cases.

SEMCLASS represents the semantic class of the predication. The range of possible values is represented in table 3.

REFERENCE and LOOKBACK are used for referential distance computations. REFERENCE contains a unique string identifying the referent of the predication. In the Clause table, this referent is some event mentioned elsewhere in the text. LOOKBACK is a utility field in which a utility program that computes referential distances will load its result. These distances are typically measured in clauses or words.

NOUNHEAD is used in the Clause table entry for a relative clause to reference the noun phrase being modified. This is an index into the Argument table. Notice that both the SOURCE and NOUNHEAD values are needed to find the correct Argument entry. It is also used for other modifying participial clauses and complements.

CLAUSEHEAD is used to represent any kind of coordination or subordination of clauses. Subordinate clauses refer to their head clauses using CLAUSEHEAD and the second clause in a coordinate construction refers to the first in the same way.

ROLE indicates the syntactic role played by a subordinate or relative clause. The range of possible values is listed in table 4. The notation "-" is used for all clauses that are not relative or conjoined.

SLOT indicates the position of the relative or subordinate clause relative to its head and is left empty for other types of clauses. The range of possible values is "B" for before, "A" for after, and " " for unknown.

Clause Role Table

Possible Value	*Description*
-	not conjoined
ADV	Adverbial
CMP	Complement
CONJ	Conjunction
REL	Relative
?	Unknown

Table 4: The possible values for ROLE in the Clause table

ARG_1, ARG_2, and ARG_3 contain the indexes of entries in the Argument table corresponding to the subject, direct object, and indirect object respectively. If no such syntactic element exists, the field is left blank. Note again that the SOURCE field is needed to find the correct entry in the Argument table.

TAG contains some arbitrary character indicating any problems that need to addressed later. It does not represent any syntactic or semantic information concerning the clause.

TEXT contains the raw text of the clause directly from the source. It will contain the textual representations of all the related noun phrases and prepositional phrases, which will also appear in the TEXT field of the Argument table.

ABSNUM is used as a utility field to record a linear count for the clauses of each manual. The INDEX field could be used were it not for the fact that, occasionally, text spans originally identified as single clauses, are broken into two clauses using fractional INDEX numbers, and, likewise, text spans originally identified as multiple clauses, are consolidated into a single clause, thus skipping an INDEX value.

2.1.2 Examples

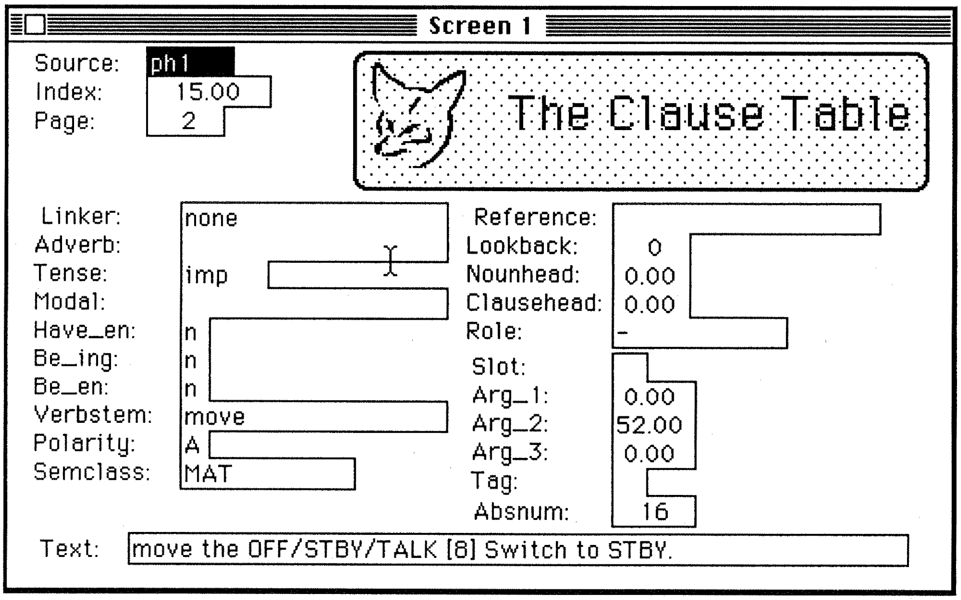
Figure 2 is an example record from the Clause table for an instructional text database. In this example, a simple imperative mood clause with the verb stem "move" is represented. It is from the source "ph1" indicating the first of the phone manuals. Notice that the ARG_2 slot indicates that the argument record number 52 is the direct object. This record will be discussed later and is represented in figure 3.

2.2 Arguments to the Clauses

The Argument table stores both syntactic and semantic information concerning noun phrases and prepositional phrases in the database, one record being stored for each. The table represents NPs and PPs that serve as arguments to predications by referencing the head clause and represents possessors and modifying PPs by referencing the head NP.

2.2.1 The Fields

This section details the contents and use of each field. The primary key for the table is the combination of SOURCE and INDEX. A quick reference to all the fields in the Clause table can be found in table 5 while a more detailed discussion of each field and its possible values follows directly.



Source: ph1
Index: 15.00
Page: 2

The Clause Table

Linker: none
Adverb:
Tense: imp
Modal:
Have_en: n
Be_ing: n
Be_en: n
Verbstem: move
Polarity: A
Semclass: MAT

Reference:
Lookback: 0
Nounhead: 0.00
Clausehead: 0.00
Role: -
Slot:
Arg_1: 0.00
Arg_2: 52.00
Arg_3: 0.00
Tag:
Absnum: 16

Text: move the OFF/STBY/TALK [8] Switch to STBY.

Figure 2: An example record from the Clause table

Argument Quick Reference Table

Field Name	*Field Type*	*Field Size*	*Description*
SOURCE	Character	003	A code indicating the source text of the argument.
INDEX	Number	008	The argument number within the source text.
SEMCLASS	Character	006	The semantic class of the argument.
TYPE	Character	006	The type of the embedded noun phrase.
NUMBER	Character	001	The number of the embedded NP (singular or plural).
PREP	Character	012	The preposition used (if any).
DET	Character	005	Any determiner used.
QUANT	Character	006	Any quantifiers used.
ADJ	Character	012	Any adjectives used.
REL_PP	Character	001	Whether the argument has a relative clause or PP modifier.
REFERENCE	Character	014	The referent of the noun phrase.
LOOKBACK	Number	005	The distance looked back by the reference (if any).
CLAUSE_HD	Number	005	The clause to which this is an argument.
NP_HD	Number	005	The noun head adjectival NPs and PPs.
ROLE	Character	006	The role of the argument.
SLOT	Character	001	The slot of the argument.
TAG	Character	001	A field used to mark questions or problems.
TEXT	Character	100	The raw text of the argument.

Table 5: Quick reference of Argument table field types

Argument Semantic Class Table

Possible Value	*Description*
HUM	Human
INAN	Inanimate
ORG	Organization
NATION	Nation
BODY	Body part
TIME	Time expression
PLACE	Place expression (location)
PROP	Proposition (verbal/mental)
PART	Part of (a number of...)
DUM	Dummy IT (Subj. or Obj.)
ABS	Abstract entity
EVENT	Event (nominalization etc.)
?	Don't know

Table 6: The possible values for SEMCLASS in the Argument table

SOURCE, as in the Clause table, is a 3 character string coding the particular manual that the argument is from. It allows INDEX to be an index to arguments in each text that is included in the database, starting from 1 for each source. This allows the text from the various sources to be removed and edited without affecting the argument numberings for the other sources.

INDEX, as in the Clause table, is the number of the particular argument. Taken together, SOURCE and INDEX form the primary key on the Argument table. Although this number allows a fractional component as in the Clause table, it is not used. The numbering of the arguments does not correspond in any way with the numbering of the clauses in which they are included. Thus, new arguments, added to the table at later times, are simply added at the end of the table. The linear order of the source text must, therefore, be determined from the Clause table, not the Argument table.

SEMCLASS represents the semantic class of the referent of the argument. The range of possible values is represented in table 6.

TYPE specifies the type of the embedded noun phrase, giving both syntactic and semantic information. The range of possible values are given in table 7.

NUMBER gives the number of the noun phrase embedded in the argument. The range of possible values is given in table 8.

PREP, DET, QUANT, ADJ indicate the lexical preposition, determiner, quantifier, and adjective(s) used in the argument, if any. Currently, if these fields are filled the remaining information is not stored, except in the raw text field.

REL_PP indicates whether the argument has a relative clause or modifying prepositional phrase. The range of possible values are given in table 9.

Argument Type Table

Possible Value	*Description*
?	Can't tell
ADJ	Referring adjective
LEX NP	with lexical head
NAME	Name of person, place etc.
PRN	Personal pronoun
DUMP	Dummy pronoun IT
RPRN	Relative pronoun
DPRN	Demonstrative pronoun
IPRN	Indefinite pronoun
TIME	Time expression: dates etc.

Table 7: The possible values for TYPE in the Argument table

Argument Number Table

Possible Value	*Description*
S	Singular
P	Plural
M	Mass
-	No number (eg adjs)
?	Can't tell

Table 8: The possible values for NUMBER in the Argument table

Argument Rel_PP Table

Possible Value	*Description*
R	Relative clause
L	Participial clause
T	Infinitival phrase
P	Prepositional phrase
M	More than one p.m.
C	Noun comp clause
-	No postmodifier
?	Not sure

Table 9: The possible values for REL_PP in the Argument table

Argument Role Table

Possible Value	*Description*
Comp	nominalized complements of nominalizations
A	
S	
O	
SO	For unaccusatives, e.g. open, extend
AS	For unergatives, e.g. attack, eat
AO	e.g. Bob-Bill hostility
1	Logical subject (A or S)
2	Logical object (P/O
3	Indirect object, recipient etc.
Pred	Predicate NP (comp. of "be" etc.)
Loc	Place
Time	Time
Inst	Instrumental
Com	Comitative
Obl	Other obliques
Poss	Possessor (generalized)
Mod	Noun Modifier
Adj	Adjective
App	Apposed NP
Conj	Conjoined NP
?	Can't tell

Table 10: The possible values for ROLE in the Argument table

REFERENCE and LOOKBACK are used as in the Clause table. REFERENCE contains a unique string identifying the referent of the predication. This list of possible referenced objects must be built for each source type. LOOKBACK is a utility field in which a utility program that computes referential distances will load its result.

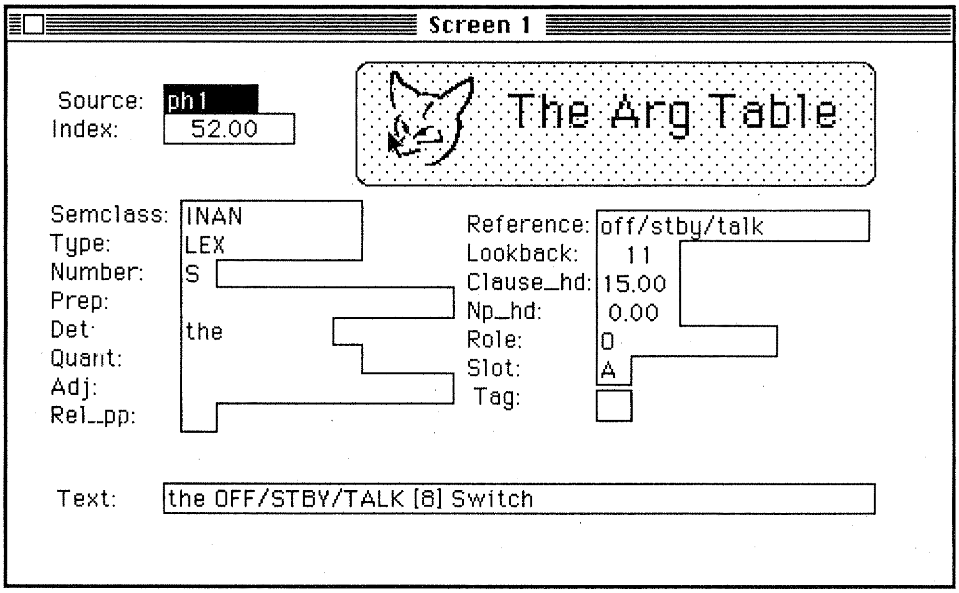
CLAUSE_HD is the index number of the clause which contains this argument.

NP_HD is the index of the argument which the current argument modifies. This is used in the case where the argument is a possessor or an adjectival prepositional phrase. It is also used to indicate the first member of a conjoined or apposed NP in the record for the following ones.

ROLE gives the case role of the argument. The range of possible values are given in table 10.

SLOT indicates the position of the argument relative to its head. The range of possible values is " ", "B" for before the predicate, and "A" for after the predicate. " " is only used for incomplete sentences containing no predicate.

TAG is used as in the Clause table. Some arbitrary character is put in TAG to indicate any problems that need to addressed later. It does not represent any syntactic or semantic information concerning the argument.



Source: ph1
Index: 52.00

Semclass: INAN
Type: LEX
Number: S
Prep:
Det: the
Quant:
Adj:
Rel_pp:

Reference: off/stby/talk
Lookback: 11
Clause_hd: 15.00
Np_hd: 0.00
Role: O
Slot: A
Tag: □

Text: the OFF/STBY/TALK [8] Switch

Figure 3: An example NP record from the Argument table

TEXT, as in the Clause table, contains the raw text of the argument directly from the source. The text here will be repeated in the Clause table in the record for the clause head.

2.2.2 Examples

Figures 3 and 4 are example records from the Argument table for an instructional text database. The first example shows how a noun phrase is represented. It is denoted as a singular inanimate object expressed as a noun phrase with a lexical head. The REL_PP field is blank because it does not apply. CLAUSE_HD is a reference back to the Clause table record discussed above and shown in figure 2. The second example shows how a prepositional phrase is represented. Here we have the preposition field filled with the preposition "to." This record, as in the previous example, refers the clause record discussed earlier as it is a separate argument to that clause.

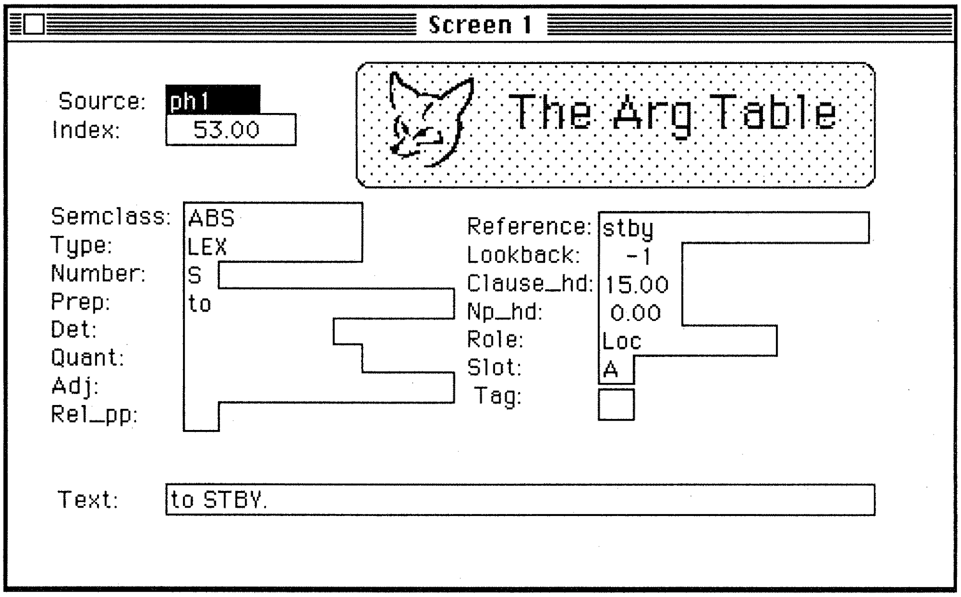
2.3 Rhetorical Structure of the Text

The RST table stores information concerning the rhetorical structure of the database. It is an implementation of Mann and Thompson's Rhetorical Structure Theory (Mann and Thompson, 1988). One record is stored for each node of the RST tree for the given text.

2.3.1 The Fields

This section details the contents and use of each field. The primary key for the RST table is the combination of SOURCE, SYNTAX_IDX, and SYNTAX_CD. A quick reference to all the fields in the Clause table can be found in table 11. A more detailed discussion of each field and its possible values follows directly.

SOURCE is used as in the Clause and Argument tables. The source text is indicated in the SOURCE field.



Source: ph1
Index: 53.00

Semclass: ABS
Type: LEX
Number: S
Prep: to
Det:
Quant:
Adj:
Rel_pp:

Reference: stby
Lookback: -1
Clause_hd: 15.00
Np_hd: 0.00
Role: Loc
Slot: A
Tag: □

Text: to STBY.

Figure 4: An example PP record from the Argument table

RST Quick Reference Table

Field Name	*Field Type*	*Field Size*	*Description*
SOURCE	Character	003	A code indicating the source text of the argument.
SYNTAX_IDX	Number	008	The clause or argument that is the focus of the record.
SYNTAX_CD	Character	001	The type of that entry (clause or argument).
CHILD	Number	008	The index of the child of the source (if a joint schema).
CHILD_CD	Character	001	The type of the child (clause or argument).
RELATION	Character	012	The type of RST relation.
NUCLEUS	Number	008	The nucleus of the relation (if a nuclear relation schema).
NUCLEUS_CD	Character	001	The type of the nucleus (clause or argument).

Table 11: Quick reference of RST table field types

RST Code Table

Possible Value	*Description*
I	Internal (high level node)
C	Leaf assoc. with a clause
A	Leaf that is an argument
?	To be filled in later.

Table 12: The possible values for SYNTAX_CD, CHILD_CD, and NUCLEUS_CD in the RST table

SYNTAX_IDX, CHILD, and NUCLEUS: The RST table uses a single record type to represent both parent child relationships (called JOINT schemas in RST) and nucleus-satellite relationships. SYNTAX_IDX is the index of the the clause or argument corresponding to parent or satellite respectively. CHILD and NUCLEUS are the indexes of either the child or the nucleus involved. Only one of these is filled in for any record.

SYNTAX_CD, CHILD_CD, and NUCLEUS_CD: Because we allow both clauses and arguments to enter into rhetorical relations, an additional field, SYNTAX_CD is included to indicate which table the SYNTAX_IDX refers to, the Clause or the Argument tables. Additionally, SYNTAX_CD can indicate that the node is a node internal to the RST tree which is associated with neither an argument or a clause. Similarly, the child and nucleus indexes require a code. The range of possible values for SYNTAX_CD, CHILD_CD, and NUCLEUS_CD is given in table 12.

RELATION indicates the type of RST relation that pertains between the nucleus and satellite or the parent and the child. The range of possible values are taken from Mann and Thompson's work, but can be augmented by more specific relations that prove useful in the particular domain of interest. Those values that have proved useful for the instructional text type are given in table 13.

2.3.2 Examples

An example of the representation is probably the best way to see how the fields in this table are used. We will show a sample analysis and representation of the following text, taken from an instruction manual for a cordless telephone (Code-a-phone, 1989).

> 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. Fully extend the base antenna [12]. Extend the handset antenna [1] for telephone conversations.

To begin with we segment the text into clause spans as follows:

(1) When the 7010 is installed

(2) and the battery has charged for twelve hours,

(3) move the OFF/STBY/TALK [8] Switch to STBY.

(4) The 7010 is now ready to use.

(5) Fully extend the base antenna [12].

(6) Extend the handset antenna [1] for telephone conversations.

RST Relation Table

Possible Value	*Description*
elaboration	elaboration
title	Title elaboration
joint	Joint schema
joint-and	Logical and
joint-or	Logical or
funct	Functional-Unknown order
funct-use	Functional-usefulness
funct-alt	Functional-alternatives
funct-pop	Functional-popularity
sequence	Procedural sequence
enablement	enablement
purpose	purpose
inform	Sat is information given
evidence	Proc. to verify action
background	Background material
precond	RST condition
circ	Circumstance
postcond	RST result (vol. or not)
cocond	Simultaneous condition
motive	Motivation
example	Exemplification (elab.)
contrast	Contrasting information
restate	Restatement

Table 13: The possible values for RELATION in the RST table

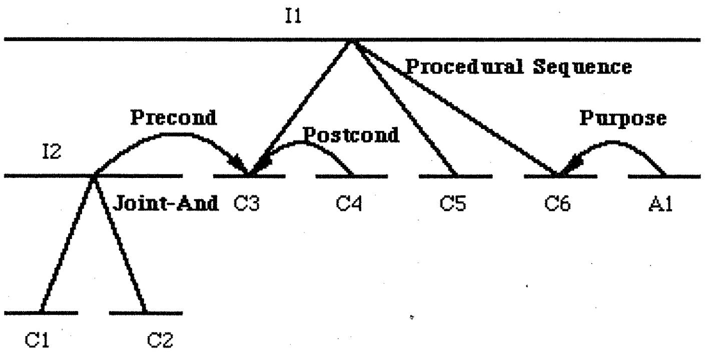


Figure 5: The RST analysis for the example text

Source	Syntax_Idx	Syntax_Cd	Child	Child_Cd	Relation	Nucleus	Nucleus_Cd
ph1	1	C			Joint-And	2	C
ph1	2	C			Joint-And		
ph1	3	C			Sequence	5	C
ph1	4	C			Postcond	3	C
ph1	5	C			Sequence	6	C
ph1	6	C			Sequence		
ph1	1	A			Purpose	6	C
ph1	1	I	3	C	??	?	?
ph1	2	I	1	C	Precond	3	C

Table 14: The RST table representation of the example RST structure

The RST structure is for the text is presented in figure 5. Note that it includes a reference to each of the text spans given above and also a reference to the phrase "for telephone conversations" which is labeled as A1[2] in the figure. The representation in the RST table for this tree is given in table 14. The source field simply indicates that the RST fields refer to text from the first phone manual ("ph1"). **Syntax_Idx** and **Syntax_Cd** give the index and code of the text span associated with the record. In this example we have the 6 text spans given in the segment list above, the single phrase node that enters into rhetorical relations (called A1), and the two internal nodes (called I1 and I2), shown in figure 5. **Child** and **Child_Cd** indicate the index and code of the first child of the node if it has one. In our example, the two internal nodes I1 and I2 are the only nodes with children. **Relation** indicates the RST-type rhetorical relation which holds between the span specified by **Syntax_Idx** and either its child, specified by **Child** or its nucleus, specified by **Nucleus**. There are two parents in this example, I2, I1, which are mentioned above. There are three satellites, I2, C4, and A1, each one of which refers to a nucleus using **Nucleus** and **Nucleus_Cd**. Notice that I2 is both a satellite and a parent. In the large context, I1 is probably also a satellite and parent as it is the node which would attach to higher level nodes in the full RST tree for the manual. This is indicated by the question marks in the table.

[2] The notation "A1" indicates that this is the first record in the Argument table for this particular source text. "C1" indicates the first record in the Clause table for this source. These codes are represented in the RST table using the index and code field pairs, one for the central node of the record, one for the first child of that node, and one for the nucleus of that node, called SYNTAX_IDX and SYNTAX_CD, CHILD and CHILD_CD, NUCLEUS_IDX and NUCLEUS_CD respectively.

3 Instructional Text

This database system has been used in a number of projects to address various issues. We now give a brief overview of one these projects, paying particular attention to the nature of the data collected and the problem being addressed.

The IMAGENE project, described more fully in Vander Linden (1992b; 1992a), is an attempt to address the relation between rhetorical structure and lexicogrammatical coding in the domain of instructional text. A database of approximately 1700 words from the procedural sections of two cordless telephone manuals has been created. This database includes clause, argument, and RST information.

This database is being used primarily to determine the mapping between local rhetorical relations such as purpose and precondition, and the lexicogrammatical form of their expression. Consequently, the database is commonly used to retrieve all the clauses and/or phrases which are represented as satellites in particular types of rhetorical relations. Important functional information can typically be derived immediately from the database. For example, purposes with global scope (i.e. concerned with a number of actions rather than just one) are fronted and put in "to" infinitive form. The scope of a purpose is stored in the RST table by representing its nucleus as either singular (local scope) or an internal node with multiple joint children (global scope). In addition, the database can easily be expanded to include any functional information that is important in determining the form of expression as dictated by the particular research question at hand.

4 Current Implementations

The database system described here is largely based on the tables discussed. Very little if any application software is necessary for its operation. Because of this, the choice of a particular database environment for the system should be based on your resources, needs, and skills, not on compatibility requirements. Most relational database systems for personal computers are easily capable of importing textual versions of existing databases. As we saw in the last section, the queries that are performed are typically simple enough to be done with the standard query interface provided with the product. Some queries, such as referential distance computation, are too complex for a QBE interface, but are very easy to implement in most database languages. This database system was originally implemented in Paradox, a relation database tool for the IBM PC, and has been ported to FoxBASE+/Mac. The porting of the system from Paradox to Fox took less than a week. The databases themselves were ported in less than a day with the help of a good IBM to Mac data link. The bulk of the porting time was spent getting Fox to perform the nice data management features that were provided automatically in Paradox. You can avoid this problem by choosing a database with a full QBE interface and high level data entry and report features. It should cause very little problem, for example, to port to FoxPro or 4th Dimension.

5 Conclusion

We have discussed a relational database tool for the representation of textual information at the syntactic and rhetorical levels. The various categories of information it represents have been detailed and examples of its practicality have been given. It is our hope that such a discussion will lead to the use and general distribution of databases of this sort.

References

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

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,* June 3–6, Dawson, PA.

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.

Vander Linden, K., Cumming, S., and Martin, J. (1992a). The expression of local rhetorical relations in instructional text. Technical Report CU-CS-585-92, the University of Colorado.

Vander Linden, K., Cumming, S., and Martin, J. (1992b). Using system networks to build rhetorical structures. In Dale, R., Hovy, E., Röesner, D., and Stock, O., editors, *Aspects of Automated Natural Language Generation*, pages 183–198. Springer Verlag.

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE NATIONAL SCIENCE FOUNDATION