Beyond Correlation: Bringing Artificial Intelligence to Events Data

John C. Mallery
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square, NE43-797
Cambridge, MA 02139-4301
Phone: (617) 253-5966
Internet: JCMA@AI.MIT.EDU

International Interactions, forthcoming, 1994.

May 17, 1994

Abstract

The *Feature Vector Editor* offers a user-extensible environment for exploratory data analysis. Several empirical studies have applied this environment to the SHERFACS International Conflict Management dataset. Current analysis techniques include boolean analysis, temporal analysis, and automatic rule learning. Implemented portably in ANSI Common Lisp and the Common Lisp Interface Manager (CLIM), the system features an advanced interface that makes it intuitive for people to manipulate data and discover significant relationships. The system encapsulates data within objects and defines generic protocols that mediate all interactions between data, users and analysis algorithms. Generic data protocols make possible rapid integration of new datasets and new analytical algorithms with heterogeneous data formats. More sophisticated research reformulates SHERFACS conflict codings as machine-parsable narratives suitable for processing into semantic representations by the RELATUS Natural Language System. Experiments with 244 SHERFACS cases demonstrated the feasibility of building knowledge bases from synthetic texts exceeding 600 pages.

Contents

List of Figures

1 Overview

Two lines of research are reported here. The first is a tool for manipulating and analyzing large databases of feature vectors (sequences of attribute value pairs). The second is a technique for reconstructing referentially-integrated knowledge representations from feature-vector data. The article introduces the "Feature Vector Editor," a new type of editor for events data that makes it easy, quick, and intuitive for people to code and modify feature-vector data. The editor is used to implement a "dynamic codebook" for the SHERFACS International Conflict Management dataset. The codebook is dynamic because it interactively translates an unreadable internal format for the human user. This user-friendly interface builds from the presentation-oriented user interface technology of the Common Lisp Interface Manager (CLIM). A user-extensible sorting and filtration facility allows data to be partitioned and reformulated. Implemented as an object-oriented database system using the Common Lisp Object System (CLOS), the Feature Vector Editor defines generic data protocols to mediate all interactions with data, including interactions with the user interface and with analysis algorithms. This data protocol allows the system to support heterogeneous data formats and means that analytical algorithms can work on any data, independent of internal format. It also greatly simplifies integration of new analytical algorithms by encouraging code sharing. Several analytical systems are currently supported: a human-directed boolean analysis based on the filtration facility; an interval-based temporal indexation that extends boolean analysis to temporal relationships; an entropy-based decision-tree learner induces explanatory rules for cases. These initial analytical tools show the power of a flexible, extensible, and cumulative environment for exploratory analysis of international events datasets.

Another deeper, semantic mode of analysis uses the Feature Vector Editor to reformulate conflict cases from the SHERFACS dataset as "reconstituted," narrative precis. Next, the RELATUS Natural Language System converts these machine-parsable textual accounts to referentially-integrated, relational graphs as it syntactically analyzes and semantically represents them. Representing unique events, actors, and relations as unique graph nodes, this referentially-integrated graph representation provides a computational foundation for symbolic reasoning processes, such as deduction, induction, or analogy. The process of building and analyzing these graph representations is illustrated through application to a text synthesized from the SHERFACS coding for the Cuban Missile Crisis. Larger experiments with synthetic texts for 17 and 244 SHERFACS cases demonstrate the feasibility of creating knowledge representations from 600-page texts. The discussion elucidates data issues relevant for application of the technique to the entire SHERFACS dataset, and suggests future coding practices to sensitize events datamakers to the needs of computational politics. Finally, automatic coding of events data from narratives is examined and a method proposed to convert graph representations into feature vectors.

2 Introduction

This article explains a facet of artificial intelligence (AI) research at M.I.T. oriented towards *computational politics*, or symbolic models of politics (Mallery, 1988). It discusses new computational ontologies for datamaking and data analysis that go beyond conventional quantitative approaches. These advances build from modern, high-productivity tools available for the Common Lisp programming language, and its cousins. As the information revolution takes hold, tools that used be so expensive that only a few major research laboratories could afford them are now becoming generally available as workstation performance increases and cost decreases. Today, political modelers are limited primarily by their imagination and the quality of their software environment.

Political scientists have developed a wide variety of datasets for quantitative analysis. Could this carefully structured information be used by contemporary AI systems? Yes, the idea is quite simple. A collection of data describing a political event, such as an international conflict, is actually a metricized precis (Alker, 1975; Mallery, 1988: 17-18; Alker & Mallery, 1988) containing what the codebook designers deem important variables for the analytical units with which they conceptualize the phenomena.[1] If these structured datasets contain the core elements of political phenomena, then why not convert this data into a form suitable for use by an AI system? With these general directions in mind, the present article discusses the development of tools to manipulate datasets, learn if-then rules summarizing regularities in them; map them into English sentences, and represent them as text models[2] using a natural language processing system.

The application of the RELATUS Natural Language System (Duffy, 1987; Mallery, 1988, 1991) to large scale text-data has been limited by the performance of the present implementation because:

- the syntactic parser does not provide complete coverage of English;
- the reference system does not provide sufficient inferential power and variety;
- and the amount and variety of background knowledge is inadequate.

There are two ways around these limitations. First, the natural-language processing model can be extended through further research. Unrestricted natural language, however, is AI-complete, requiring the creation of a complete artificial intelligence. Unfortunately, this full solution to the AI problem is not a short-term possibility. Political analysts will therefore need to rewrite the text they wish to model according to the linguistic processing model supported by available natural language systems. The rewriting typically involves making explicit background inferences, such as metaphors or deductions, and making syntactic constructions conform to the grammar implemented by the parser.[3] In practice, political text modeling uses a combination of the two strategies. With progress in AI research, the amount of rewriting will necessarily decrease, yet previously machine parsable text will *remain accessible* as knowledge sources.

[1] The selection of the analytical units preselects the category system that the codebook and the dataset reify. If there are multiple, competing category systems corresponding to different research paradigms, the symbolic analysis of the coding categories together with their data may help synthesize across them and characterize their differences.

[2] Those unfamiliar with text modeling or natural language models of politics can find accessible introductions in (Mallery, 1988; Alker, *et al.*, 1991).

[3] Hurwitz and Mallery (1987) explain how to rewrite text to conform to the existing processing model of the RELATUS system.

SHERFACS' *datamodel*, the organization of data, is a hierarchical tree of feature vectors.[4] *Feature vectors* are collections of attribute value pairs, or features. Often, they are represented as a vector of values that are indexed positionally rather than by an explicit marker. A feature may also be interpreted as a variable with an associated value, whose type may range from two-valued boolean to some complex encoding. Alternatively, a feature vector can be viewed as a list of properties with associated values, or as a frame with slots and values. However, there is a key restriction: no value can be another feature vector. In this case, feature vectors become graphs. Graphs are a more complex data structure and have received less formal study by computational theorists. However, they are known to be more general and they frequently appear in advanced AI applications (Winston, 1975, 1980). For example, the semantic representation used by RELATUS is an arbitrarily expressive graph – a representation capable of representing anything. By virtue of its unlimited expressibility, this graph representation is capable of expressing natural-language meanings. The important point is that feature vectors are significantly different from graph representations.

The data format that RELATUS uses is relational graphs, constructed from natural language sentences according to a model of language restricted by the current syntactic, reference, and pragmatic capabilities of the system. The curious information theoretic question is how feature-vector data can be expanded into a relational graph? Is there not too little information in the feature vector? Yes and no. Feature vectors can be used to encode relations by using more than one variable and providing enough attribute value pairs. For example, three variables could be used such that one encodes the subject of a relation, another the relation, and a third the object of the relation. In SHERFACS, the relationships of interest are those between actors, management agents, and outcomes in a conflict. The parties are identified and the relationships are noted. Thus, in many cases it is possible to reconstruction the relationships between the parties. In fact, this is the primary goal of the research reported here. More will be said below about how this was done and what problems were encountered (see section 4.1).

3 The Feature Vector Editor

3.1 Design Criteria

Before a relational model could be reconstructed from SHERFACS, a tool was needed to edit, inspect, manipulate, and prepare the SHERFACS data. As it evolved into an independent tool, the following design criteria emerged for the Feature Vector editor.

1. **Use Issues:**

   - **Ease of Use:** The editor should exploit the available computational technologies to make modification of codings easy. It should eliminate the need

[4] Unseld and Mallery (1992) provide more detail on feature-vector datamodels and the organization of data in SHERFACS.

for the human to know the internal representation of the data, and thereby, significantly simplify use.

- **Intelligibility:** It should present information to the human in ways that minimize the amount of specialized knowledge required to use the editor. Instead of presenting internal codings to the human, it should present full text descriptions, use good mnemonics, and provide context-sensitive help to users on demand.
- **Minimization of Coding Errors:** It should minimize the possibility of miscodings due to typographical errors, incorrect internal code selection, or contradictory codings.

2. **Manipulation Issues:**

   - **Filtration and Sorting:** It should provide a native filtration and sorting framework to allow users to partition and order their data. These operations facilitate the application of analytical methods to conceptually partitioned subsets of the data. They also allow a user to obtain a rough idea of how different variables partition the data.
   - **Reformulation of Datasets:** The Editor should make reformulation and selective recombination of datasets quick and easy.[5]

3. **Data Issues:**

   - **Data Independence:** The Editor should achieve independence from any particular data by using declaratively specified codebook to interpret variables for a particular dataset.
   - **Flexible Data Formats:** The underlying data model should be flexible enough to allow heterogeneous formats for source data. It should support whatever data formats analytical algorithms might require.
   - **Data Compression:** Flexibility notwithstanding, the data should be stored space efficiently so that the system can be used with large datasets.

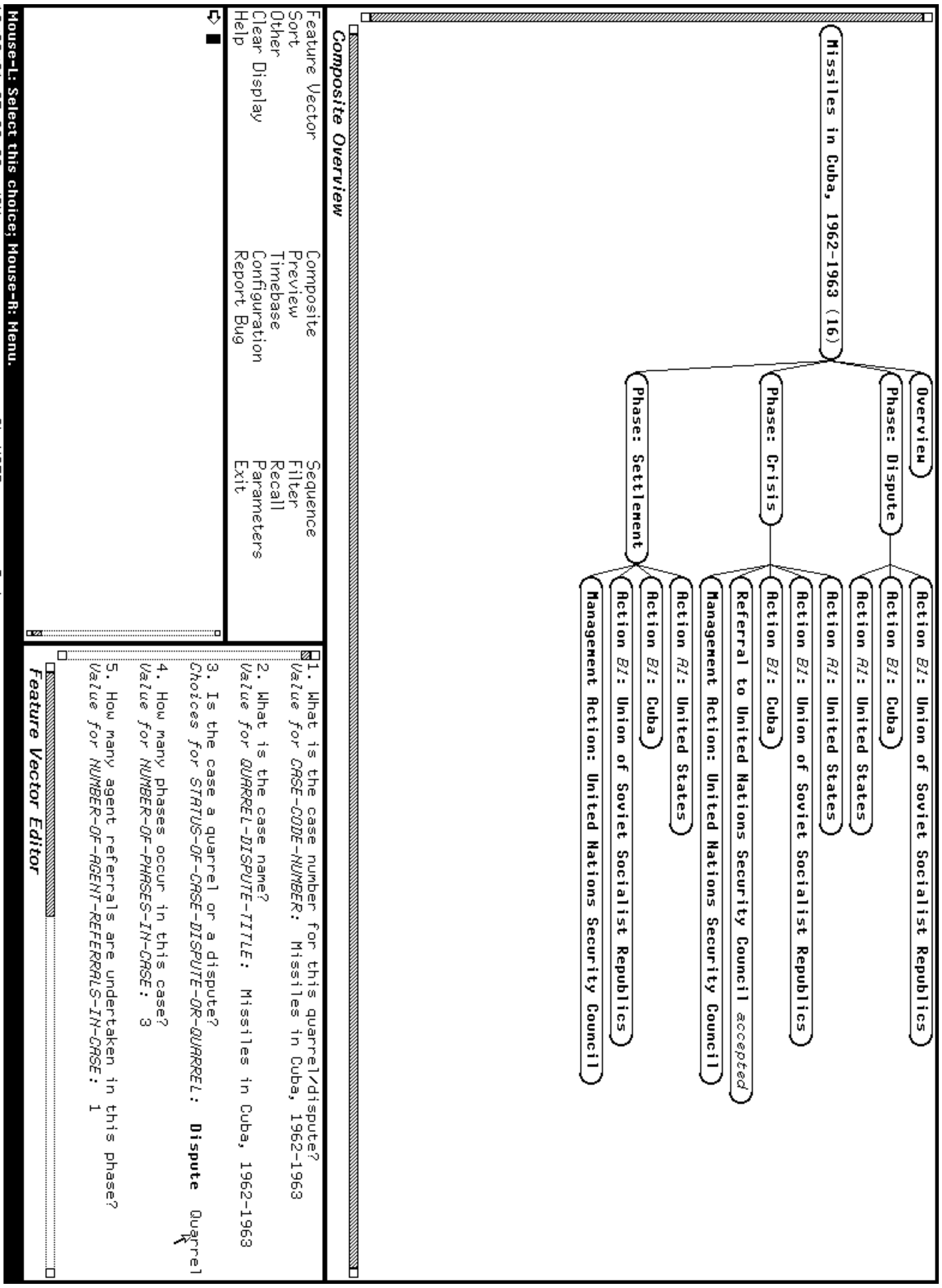
4. **Extensibility Issues:**

   - **Evolutionary Design:** The Editor should embody evolutionary software design principles that facilitate a continuous extension of its functionality and allow researchers to take advantage of developments in workstation computing as they become available.
   - **User Extensibility:** Since all functionality which a user might want cannot be anticipated by system designers, the Editor should make it easy for users to extend the system.

3.2 A Dynamic Codebook

[5] A user interface for defining and reformulating datasets for non-programmers, *a dynamic codebook editor*, has not been implemented yet.

Figure 1: The overview of the composite description for the Cuban Missile Crisis.



Instead of using coding sheets and subsequent data entry (perhaps using punched cards), the Feature Vector Editor exploits modern user interface technology. When editing a feature vector, human coders see on the computer screen something resembling a multiple choice questionnaire. The questions and the answers are written out in unabbreviated English so they can be readily understood without need to refer back to a paper codebook. The difference between the presentation of the coding for a feature vector on the screen and a paper multiple choice is that the coder can enter or modify a value simply by clicking the mouse on a choice. Better yet, the human needs only to look at the full English rendition of the coding question and the acceptable answers, instead of having to memorize tedious internal codes or shuffle through the codebook to find the internal code required at the time.

The point and click approach (figure 1 and figure 2) eliminates various time-consuming data conversion tasks that typically underutilize human cognitive capabilities.

- **Head to Paper Transfer:** It eliminates coding cases on paper coding sheets.
- **Paper to Computer Transfer:** It eliminates data entry from the coding sheets.
- **Category to Code Conversion:** It eliminates the need to contend with internal codes with low mnemonic value.

By eliminating these tedious tasks that incline people to error, the dynamic codebook approach increases coder productivity. But, the consequences for error go beyond even this; elimination of these three data conversion operations also removes the possibility of introducing errors through these mechanisms. All error in the dataset will arise from conceptual errors or accidental selection of an incorrect choice.[6]

Perhaps the major benefit of a dynamic codebook is that it frees coders from the coding mechanisms and allows them to concentrate on the material to code. Between them, automation of data conversion and greater focus on the coding task eliminates important sources of error and dramatically increases productivity.

Unlike practically any other feature-oriented datasets, SHERFACS has considerable structure per coded case. Instead of a simple feature vector, SHERFACS uses five different types of feature vectors to describe cases. Fortunately, the Editor allows feature vectors to be organized in clusters associated with cases. Thus, instead of considering SHERFACS as five different datasets, the different feature-vectors were combined in a tree structure describing each case. The top is a case overview, the next level contains a temporally-ordered sequence of feature vectors describing each conflict phase, and finally, the bottom level encapsulates (for each phase) all actions of parties, referrals to conflict management agents, and actions by conflict managers. Below the top level, the number of feature vectors can vary, depending on the number of phases (time periods) coded and the number of actors or conflict managers involved in each phase. This hierarchical

[6]Experience with menu-driven data acquisition of lexical information in RELATUS has shown that high speed data entry can introduce error. People become good at clicking on the correct items and they then tend go too fast to fully check the menus. This lack of care is a new source of error. But, the use of constraints between possible values and among variable categories can prevent semantically ill-formed, menu-elicited entries.

organization introduces the opportunity to help human coders or analysts see the overall structure; a graphical display of the feature-vector tree for SHERFACS cases (figure 1) provides a high-level summary.[7]

Despite full English choices available in the dynamic codebook, there is no loss of space efficiency in the coded data. Space-efficient data representation are maintained because the feature vector editor automatically translates between the compact internal format and the presentation the human sees. This translation occurs only for the data with which the coder is interacting. Because the protocol mediating the translation process is generic and application independent, the operation of the dynamic codebook does not depend on any specific internal representation for feature vectors. It, therefore, allows the use of any internal format, according to the the efficiency needs of the application. For very large databases, the data might actually be computed rather than stored, or it might be served by a data server analogous to a fileserver over a local, or non-local, computer network.

3.3 The Implementation

The Feature Vector Editor is written in Common Lisp to serve as a user-extensible environment for data manipulation and analysis. Because the goal is to develop a fine-grained vocabulary of operators that programs and software developers share, Common Lisp is the best available choice today. Once operators are abstracted, programs and users call them, instead of (re-)writing them again. This fine-grained abstraction yields the highest productivity for research that develops new analysis algorithms because:

- **No Code Duplication:** No effort is wasted writing once again code that already exists elsewhere;
- **Powerful Abstractions:** Function-compositional languages, such as Common Lisp, allow, and even encourage, programmers to build up high-level abstractions to meet functional requirements. Indeed, writing a package of programs to achieve a purpose becomes a process of developing a vocabulary of operators to specify the desired functional process.[8]

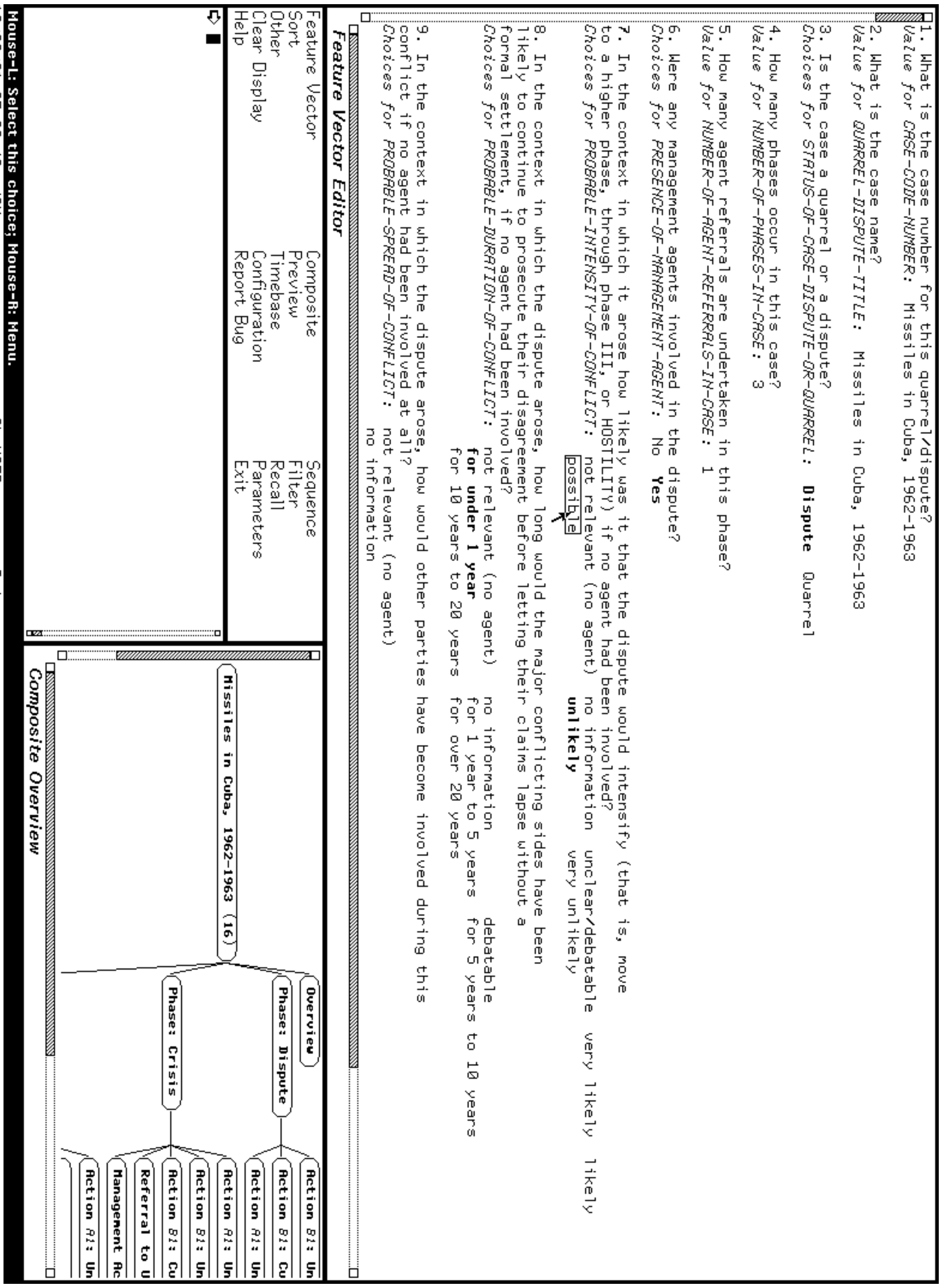
A high productivity environment reduces the work required to develop new packages and makes most efficient use of the scarce time of scientists.[9] Generalizing the individual case to a community of researchers, sharing this kind of environment and each other's

[7]The structure of feature vectors describing an example and the most salient variables will naturally vary across applications; thus, the system described here provides a general mechanism for graphical summarization that is easily tailored to each application.

[8]See Abelson and Sussman (1985) for a detailed introduction to abstraction in a functional language.

[9]The bulk of the 24,000 lines of Lisp code (.96 megabytes) that comprise the basic Feature Vector Editor – independent of components such as the time indexation system or the I^2D rule induction system – were written by the author over two weeks. Thereafter, new extensions were added incrementally, when need for research. Using Common Lisp in a good programming environment, a competent programmer can write as much as 1,000 lines of debugged code per day! After the generally-accepted ten-to-one

Figure 2: Editing the overview feature vector for the Cuban Missile Crisis.



extensions maximizes the rate at which the community searches the space of methods for events data analysis. Moreover, well-abstracted code joins with self-documenting capabilities in the Feature Vector Editor to make it an effective pedagogical environment; everything is readily available for inspection, modification, and experimentation.[10]

3.3.1 CLOS

Common Lisp embeds a native object-oriented programming language, the Common Lisp Object System (CLOS). All significant datastructures in the Feature Vector Editor are implemented as CLOS objects (Keene, 1989).[11] CLOS distinguishes classes and instance objects, the behavior of instance objects is defined by associated classes, which can inherit functionality and state variables from multiple superior classes. Instances can have local state, held by instance variables, and they can have methods, or operations, which they support. Generic operations constitute a protocol that instances of different classes all support with their own, possibly different methods. A pervasive object-oriented implementation helps enforce abstraction and modularity as it enhances flexibility and code sharing.

3.3.2 CLIM and Presentation Types

Modern Common Lisp provides a high-level window system tool, the Common Lisp Interface Manager (CLIM).[12] By introducing high-level abstractions for specifying window interfaces, CLIM makes it easier and quicker to define sophisticated window interfaces.

conversion for the abstractive power of Lisp, equivalent functionality would require 10,000 lines of C or Fortran, or one person-year of industrial-strength work in those programming languages. However, competence in Lisp may be elusive because it demands understanding not just fairly extensive language constructs and programming tools, but also, how to effectively abstract problems. Fortunately, Lisp productivity is so high that it makes sense for those most capable of effective abstraction, the scientists, to express problem solutions directly in Lisp – a far more concise and debuggable rendition than textual descriptions. Indeed, certain problems, such as natural-language understanding, are so complex, difficult, and experimental that progress simply would not occur in less productive languages.

[10] Apart from the low cost of suitable workstations, the key element underlying the popularity of UNIX operating systems is the availability of large, and often free, program libraries. Shareware in C can occurs at the program level, not at the subroutine or operator level, because C neither encourages nor effectively supports the kind of fine-grained operator use found in Lisp. From a technical standpoint, then, Common Lisp provides a better programming language and environment for sharing code because more sharing can take place, down to the subroutine level. Moreover, Common Lisp always supports dynamic linking, incremental compilation, and a host of other language features that encourage an evolutionary programming model, based on successive refinement of individual operators.

[11] The Feature Vector Editor was originally implemented in October 1987 using the Flavor System (Symbolics, 1986), the first object-oriented programming language embedded in Lisp. In 1992, it was converted to CLOS for portability.

[12] CLIM (Rao, *et al.,* 1991) is a more efficient and portable reimplementation of the Symbolics Dynamic Window System (Symbolics, 1986). Written in Common Lisp, CLIM runs on a variety of workstations, implementing its machine-independent abstractions with native window systems and mimicking their look and feel.

Instead of requiring many months, the bulk of user interface for the Feature Vector Editor was written in several days.[13] Beyond general window system abstractions, the Feature Vector Editor heavily exploits a new and important CLIM abstraction; *the presentation system* controls how the user specifies or perceives arbitrary Lisp objects by means of presentation types associated with Lisp types – such as feature-vector objects and the values they may hold. Each *presentation type* has a method to *present*, or print, the object on the computer screen and a method to *accept*, or receive input, of an object of the type from the user, whether via the keyboard or the mouse. The Feature Vector Editor uses this presentation system to uniformly translate between the internal format and the displayed format of feature-vector values. This bidirectional translation is implemented by some Lisp code, provided by each application to cover the range of data types requiring translation. Once defined, these presentation types mediate all data entry and display, and thus, users interact only the external, user-friendly representation, and never the internal representation.

3.3.3 Dataclasses, Feature-Vector Types, and Variable Types

A *dataclass* is a a taxonomic hierarchy whose root is a class spanning an entire dataset. The inferiors of the dataclass are either data subclasses within the dataclass, general divisions within the dataset, or it feature-vector types. A *feature-vector type* contains all the information required to support operations on feature vectors of the type. The dataclass controls operations over the entire dataset that the subclasses and feature-vector types share. Subclasses may perform analogously for subsets of feature-vector types. For example, the dataclass supports remapping of compressed variable names into longer names with better mnemonics.

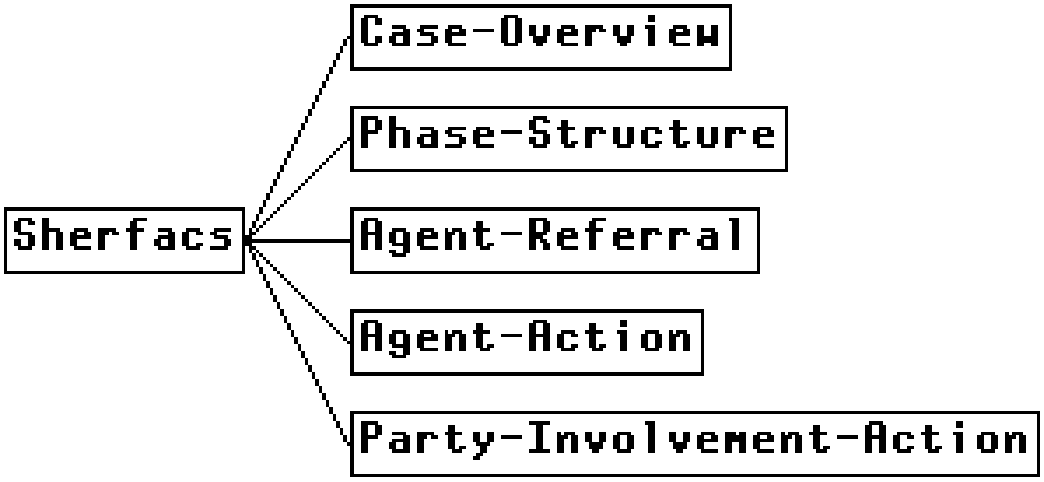
Feature-vector types supply all the functionality of individual feature vectors they govern, including methods for accessing, manipulating, and presenting the data they hold. When they are created, feature-vector types parse a variable coding specification from their definition of the dynamic codebook and instantiate *variable type* objects for every variable that many have a value in a feature vector of their type. The resulting variable type objects know how to access, display, and modify the specific values for variables in feature vectors. Analogously to data classes and feature vector types, variable types may also inherit functionality or class membership from *variable classes* that span them, typically to avoid redundant definitions at the variable-type level. In general, the functionality available from feature vectors usually inherits via variable types from the feature-vector type.

3.3.4 Feature Vectors

Feature vectors are implemented as CLOS instances with three instance variables. They have a pointer to their feature vector type object, an identifier, and a pointer to their internal data representation. For SHERFACS, the internal representation is a Common Lisp vector that can be accessed with numerical indices. The SHERFACS feature vectors

[13] It was actually written in October 1987 using the Symbolics Dynamic Window System, and 1992, converted to CLIM for portability to other platforms.

Figure 3: Feature-vector types in the SHERFACS dataclass.



are identified by a case code for the conflict case. Access, display, and modification operations on feature vectors are mediated by their feature-vector type object. Retrieving the value of a variable is an access operation. Displaying all the values of variables is a display operation. Clicking on a multiple value choice for a variable is an operation combining display and modification, when the feature vector is updated with the selected value. The feature vector type, in turn, organizes this functionality according to variables (or frame slots).

When special operations beyond these standard operations are needed, an application may define a dataset-specific CLOS class that inherits the CLOS class feature-vector, (and therefore all the usual generic operations defined for it) and define the additional operations on this CLOS class. Only if operations do not generalize beyond the application should they be defined in this way. In the SHERFACS application, for example, generation of narratives expressing the information of a feature vector relies on declarative specifications for variables held by feature-vector type as well as some special operations inherited from a SHERFACS-specific CLOS class to the feature-vector class used by SHERFACS. The general rule is that when functionality is not permanent, such as when codings may change, that functionality should be specified declaratively by the dynamic codebook.

3.3.5 Composites

In SHERFACS, there are various types of feature vectors describing cases. It therefore makes sense to group the different types of feature vectors for a conflict case together. *Composites* are CLOS instances that group feature vectors of the same or different types together because they describe a unitary example or case. Composites have pointers to their dataclass and any subclasses to which they belong. They have an identifier which their dataclass object associates with them for lookups. More interestingly, they have a STRUCTURED-FEATURE-VECTORS instance variable that maintains an application-specific organization of the feature vectors contained by the composite. For SHERFACS, a structuring function defined on a specialization of the CLOS class for composite is given in the definition of the SHERFACS dataclass. This structuring function is responsible for sorting unstructured feature vectors into a structured tree with a *case overview* (CO) at the top, temporally-ordered *phase structure* (PS) feature vectors at the next level, and below each phase, temporally-subsumed feature vectors for *party involvement action* (PIA), management *agent referral* (AR), and management *agent action* (AA). Complex datasets that contain heterogeneous feature-vector types demand an organizing abstraction like composites. Thus, most data level operations use the composite object as the primary data object. Simpler datasets will also need to associate feature vectors with composites to ensure that the operations supported by composites will be available for their datasets.

An application will typically define useful operations on its composites. SHERFACS, for example, defines operations on its specialization of composite to show or edit the source text from which a conflict case was coded (figure 4).

3.3.6 Sequences

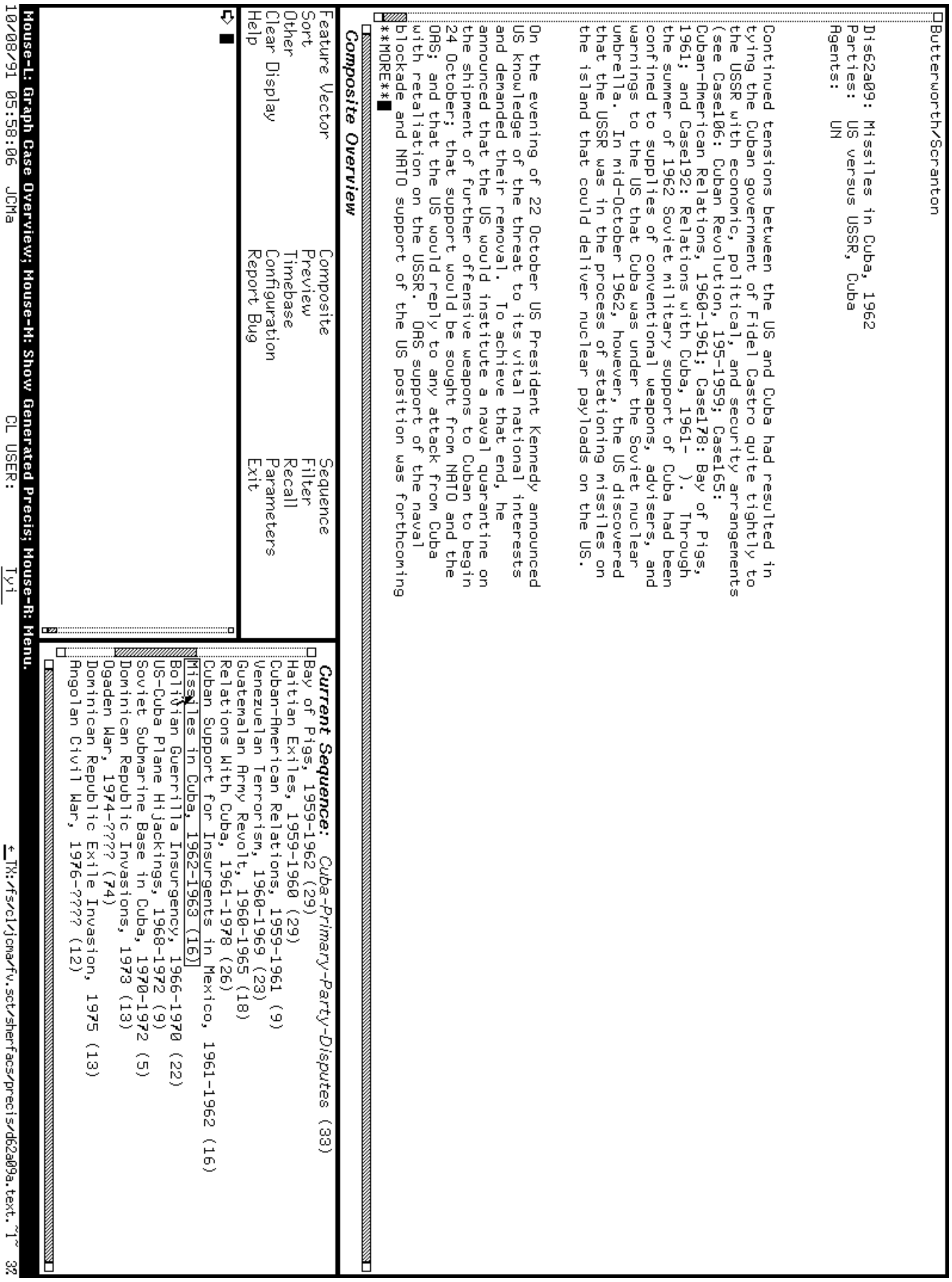
Just as composite group related feature vectors together, the Feature Vector Editor organizes collections of composites into *sequences*. These collections of composites are the main unit to which analytical algorithms are applied. For example, figure 5 shows the Editor in the sequence configuration, with a sequence of Cuban disputes as the current sequence. *Filters* provide a means to reduce sequences into smaller collections satisfying the filter-tested parameters. *Sorters* provide the means to order them for human viewing or for order-dependent analyses. The Feature Vector Editor provides general mechanisms for defining and invoking filters and sorters, but each application must define their own. Set operations, such as union or intersection, *inter alia*, provide other ways of combining, reducing, or creating sequences. Naturally, sequences can be saved to and restored from files.

3.3.7 Generic Operations

All the significant components of the Feature Vector Editor are implemented with CLOS instances capable of supporting generic operations (Moon, 1988; Keene, 1989). Generic operations enforce a protocol of communication between objects that encourages modularity and data abstraction, largely because external objects need not know the internal data format of the objects with which they interact. The protocols supported by each type of CLOS object can be clustered into three categories:

- **Data Protocol:** These generic operations provide an advertised program interface that analysis packages can use to obtain and modify variable values in composites, or feature vectors. This protocol also spans sequences of composites.

Figure 4: The Butterworth-Scranton summary source text for the Cuban Missile Crisis is called up by clicking on the composite for the case.



- **User Interaction Protocol:** These generic operations provide an advertised program interface that analysis packages can use to elicit or present information.
- **Internal Operations:** Other generic operations are internal to the feature vector editor and should not concern developers of new analytical packages.

The definition of the dataclass and feature-vector types for a specific dataset provide *all* the information that controls data protocol and the user interaction protocol for a specific application.

Use of generic operations, and advertised protocols, has various beneficial consequences:

- **Specialized Objects:** An application may define its own specialized objects that inherit from the generic objects provided by the feature-vector system. By default the new application object inherits all the functionality from the generic objects. It may however define additional operations relevant to the application or even redefine some otherwise inheritable operators with ones specialized to its circumstances. In SHERFACS, there are application-specific objects for all feature vectors, which among other things, define the specialized printing methods for each. In general, inheritance provides a modular way to customize functionality.
- **Extensibility:** The ability to customize generic operations builds extensibility and flexibility into the implementation.
- **Heterogeneous Data:** Because generic functions may be shadowed by feature vectors (or variable objects), the application may override the default implementation of variable access to feature vectors in order to support its own data formats. Assuming commensurate datamodels,[14] this allows analytical methods to use heterogeneous data formats. Heterogeneous data formats can be used because the data knows how to convert itself to answer the protocol by means of which it interfaces to the analytical methods.[15] For example, the I^2D rule induction system (Section 3.4.4) can now be used with any data, answering the feature vector data protocol.

3.4 Analytical Techniques

3.4.1 Partitioning Datasets with Filters

Partitioning a dataset with filters and organizing partitions with sorters provides a convenient method to prepare coherent subsets of a dataset for analysis. At present, coherently partitioned sequences are the input for tools that learn if-then rules summarizing regularities in data (section 3.4.4) and construct knowledge bases with texts synthesized

[14] Datamodel commensurability is beyond format conversion; it must be possible to compute from the source data the values required by the algorithm that will use it.

[15] See (Rowley, Shrobe, Cassels & Hamsher, 1987) for a similar use of generic operations to implement a "protocol of inference" that gives an rule-based system access to heterogeneous knowledge structures.

from SHERFACS dispute codings (section 4.1). Normally, the initial partition is created by applying a filter to the entire dataset. The user can further filter and manipulate a sequence by clicking on the "Filter," "Preview," or "Sort" items in the command pane (visible in figures 4 and 5). Sorters and filters are defined for *specific* datasets because they only make sense for in those applications. The Feature Vector Editor supports various set operations on sequences, such as commands for union, intersection, and set disjoint. Negation is provided by the "Negate" filter which prompts for another filter to negate. The subset command allows new sequences to be created from existing sequences automatically by applying filters or manually by selecting composites. Previous sequences can be recalled, saved to disk, and restored from disk. Other useful operations may found by clicking on "Help."

3.4.2 Boolean Analysis

Interestingly, filters introduce a basic boolean analysis facility for the Feature Vector Editor because they allow an analyst to successively constrain data elements according to theoretically informed criteria. *Boolean hypothesis testing* applies a series filters that test composites for a boolean combination of variable values or relationships, for example sequences of phase types (figure 5). When a series of filter applications identifies useful cases, the series can be abstracted and defined as a single filter, available for future use. Although simple, boolean hypothesis testing is a powerful tool for quickly discerning the outlines of a dataset.[16]

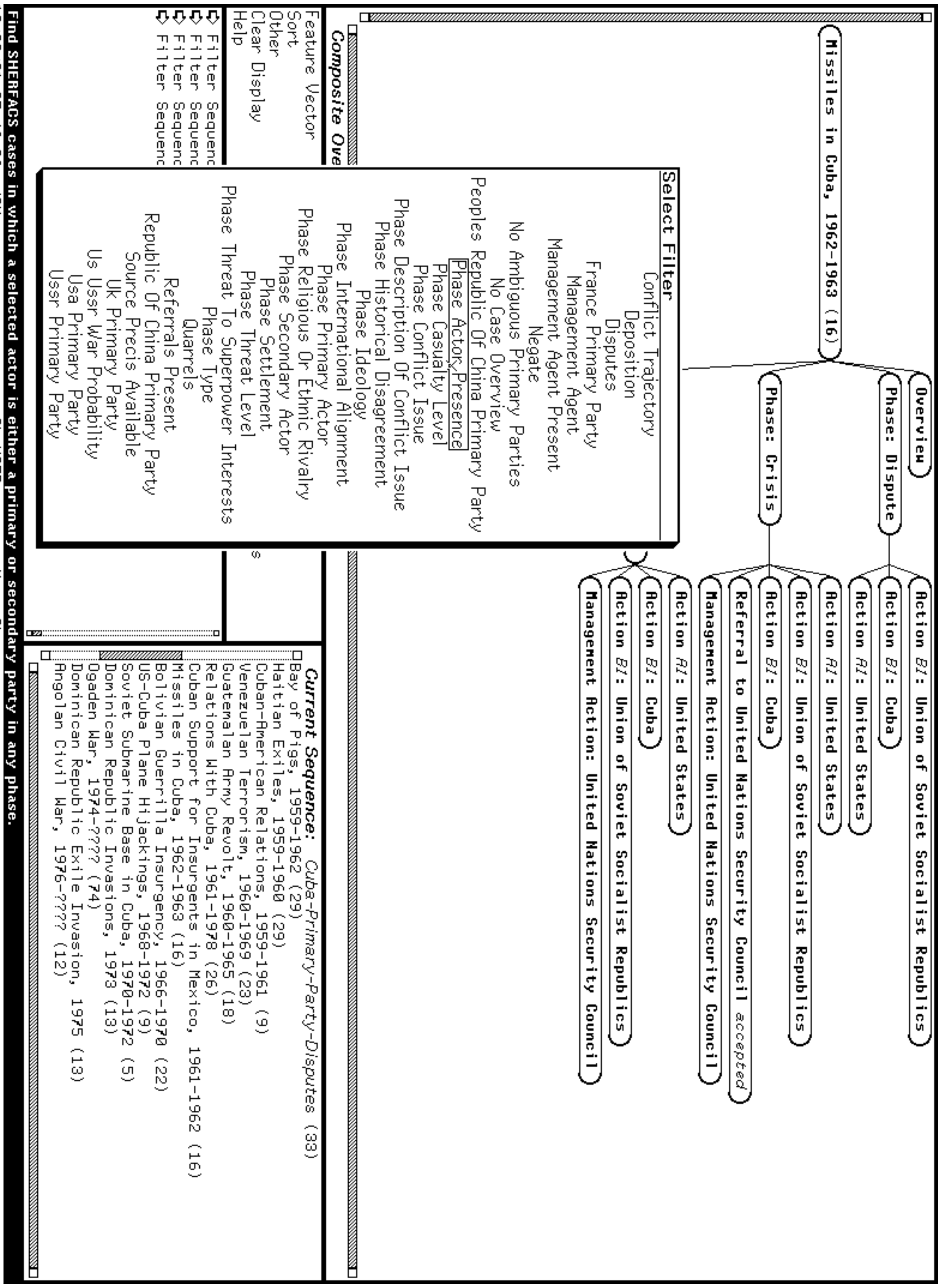
3.4.3 Temporal Analysis

An interval-time indexation system (Duffy, 1987, 1991) has been interfaced to the Feature Vector Editor. The present interface allows the elements of sequences, created in whatever way, to be mapped into the timebase system and mapped back out into sequences. At present, it is possible to create:

- **Sequence Timebases:** Users can create timebases for SHERFACS feature vectors and composites. The usual temporal operations supported in the Timebase Window can then be applied to these sequences. These include: scatter plotting the intervals (figure 6), viewing data found in temporal regions, creating chronicles (duration maps seen in figure 7), and writing chronologies. From the Feature Vector Editor Window, timebases can be created for sequences, sequences chronicled, sequence chronicles displayed, and chronologies written.
- **Temporal Sequences:** Using the "Data Region Sequence" command in the Timebase Window, a user can specify a time region in the timebase scatter plot, and

[16]Boolean hypothesis testing is actually a user-driven form of data classification based on discriminating filters. This process implements a human-directed decision tree. Here, the human is walking a decision tree in the construction of the the working sequence, and performing a task very similar to the one automated by I^2D. Since the human applies powerful inferential machinery to the classification process, the results are not just more effective but also more in tune with the purposes of the analyst.

Figure 5: Selecting a filter to partition the current sequence of composites.



then, map all the feature-vector data indexed within the selected time region back into the Feature Vector Editor as a newly created sequence.

Boolean analysis and temporal indexation facilities are routinely used to examine SHERFACS data and to partition case descriptions before applying other analytical techniques. However, boolean and temporal analysis alone can yield interesting insights. For example, Hurwitz (1991) empirically characterized the evolution of the relationship between the United States and the Soviet Union in the post-war period. He found evidence that the Cuban Missile crisis represents a turning point in the superpower relationship; the previous propensity to participate in conflicts as primary parties on opposing sides is superseded by a "chicken game" in which both apparently try to avoid direct confrontation.[17]

3.4.4 Learning If-Then Rules with I^2D

A system for learning if-then rules from a dataset is available in the Feature Vector Editor. Perhaps the most successful AI algorithms are a family of entropy-based techniques for inducing decisions trees (Quinlan, 1979, 1983; Breiman, *et al.*, 1984).[18] Because they rely on global information measures, these techniques require a one-to-one mapping between values of compared variables that is found only in flat datamodels (one feature vector). Complex datamodels – for example the hierarchically-structured feature vectors in SHERFACS – almost always give rise to one-to-many mappings between values of compared variables, but this violates the fundamental assumption of conventional decision-tree learners, rendering them unusable for all but the most limited analyses of SHERFACS.[19] Unseld and Mallery (1992) found a unifying solution to this problem that generalizes earlier techniques along several dimensions in order to handle complex datamodels. The key insight is to decompose examples into local observations until one-to-one comparisons again become possible, and then, to recombine these local comparisons to ultimately arrive at global information measures. Their *Induction Interaction Detector* (I^2D) implements an inductive rule learner suitable for studying complex datasets like SHERFACS because it supports learning on the tree-structured temporally related feature-vector data. Significant features include:

- **Time-Dependent Analysis:** Support for temporally structured data makes it possible to learn decision trees and rules to predict outcomes on the basis of prior variable values;

[17]Hurwitz' breakpoint analysis would have benefitted from statistical facilities native to the Feature Vector Editor; these would have made it easy to test statistical significance and examine variance for smaller time periods.

[18]*Decision trees* organize tests hierarchically such that running the tests along a path from the top to any leaf classifies an example within a category.

[19]In 1988, Guy Bleloch and the author experimented with ID3 on SHERFACS data in the Feature Vector Editor. Although some interesting ideas emerged, such as inducing time-lagged decision trees and partitioning disputes into coherent sequences, very few variables could be meaningfully used. They found that ID3 would require generalization to work reasonably on the complexly-structured SHERFACS data.

Figure 6: Scatter plot of disputes in which the the U.S. and U.S.S.R are primary parties.

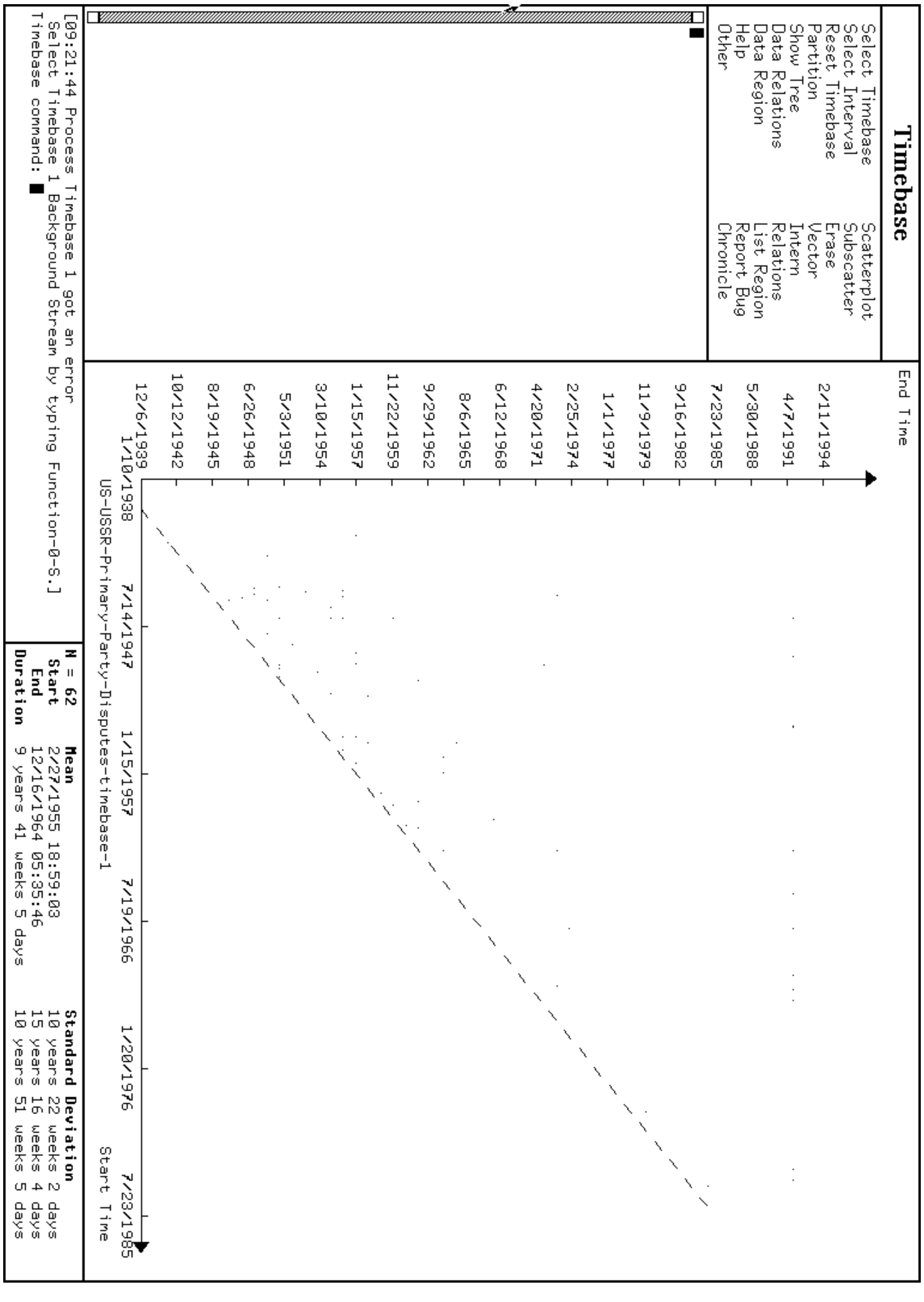
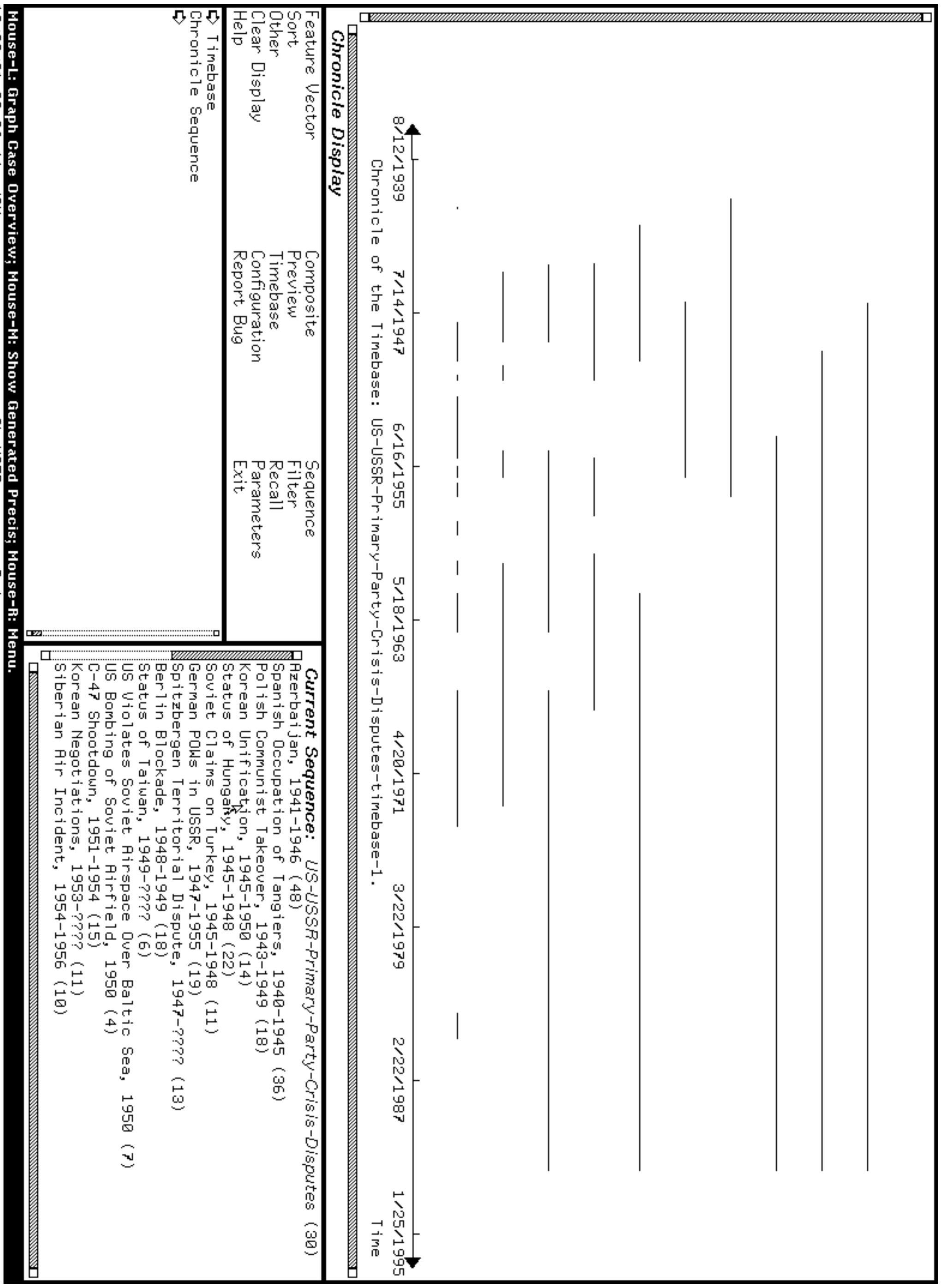


Figure 7: Chronicle of crises with the the U.S. and U.S.S.R as primary parties.



- **Regularity Recognition:** Instead of narrowly focusing on classification, I^2D is designed to identify regularities between dependent and independent variables;

- **Exploratory Data Analysis:** Because it can identify regularities in data and is not limited by degrees of freedom, I^2D is an excellent tool for exploratory data analysis, sifting through new or little-understood data to find the significant relationships that statistical methods might later test.

Considerable work went into handling various kinds of variables that appear in SHERFACS, including variables that range over multiple values within examples, variables whose number of values varies dynamically, and variables whose values are sets or sequences.[20]

The full integration of I^2D with the Feature Vector Editor provides an effective environment for learning decision trees and if-then rules. Current research using I^2D to explore the SHERFACS dataset (Mallery & Sherman, 1993a, 1993b) suggest that I^2D yields the most useful results when event data has been partitioned into coherent sequences. Figure 8 shows rules that I^2D learned from the SHERFACS coding for 30 postwar crises in which the U.S. and the Soviet Union were primary parties. The rules predict values for the dependent variables, the level of intervention by external actors in a conflict phase, based on independent variables[21] deemed relevant by I^2D in the preceding phase. The I^2D control matrix in figure 9 shows the information provided by the analyst. I^2D incorporates into the antecedents of rules only the information-bearing independent variables, normally a subset of those suggested by the analyst. For example, rule 1 predicts no external involvement when the conflict issue is any of the low-intensity issues whose disjunction is denoted by equal signs. At 100 percent, the local precision is excellent. *Local precision* refers to a rule's coverage of partial observations in the sample from which it was derived. The *global precision*, or the percentage of cases explained by the rule, is 62 percent, suggesting a rule with quite wide applicability. More interestingly, rule 2 predicts that external actors will provide military or diplomatic aid in a phase when the preceding phase is a dispute phase concerning irredentist border claims without likihood of superpower war and there are threats but no fatalities. This rule seems very good for three disputes, or 12 percent of the total. Despite the small local sample, I^2D has unified three disputes falling within a category that might be termed "superpower meddling in ongoing, ethnically-based border disputes leading to a crisis involving the superpowers." These examples illustrate how inductive rule learning could guide theory formation by detecting interesting variable clusters, but the task remains to refine these learning procedures and hone their usage into methods of empirical inquiry and theory formation.

[20]Although I^2D can now use most SHERFACS variables, continuing research aims to handle a few kinds of variables not treated as of this writing. Unseld and Mallery (1991) provide a detailed technical discussion of I^2D, but additional work needs to develop superior methods to assess the significance of decision trees and rules.

[21]Section 3.3.5 explains the abbreviations for feature-vector types prefixing variable names in figure 9.

Figure 8: Learning If-Then Rules about U.S.-Soviet Crises with I^2D.

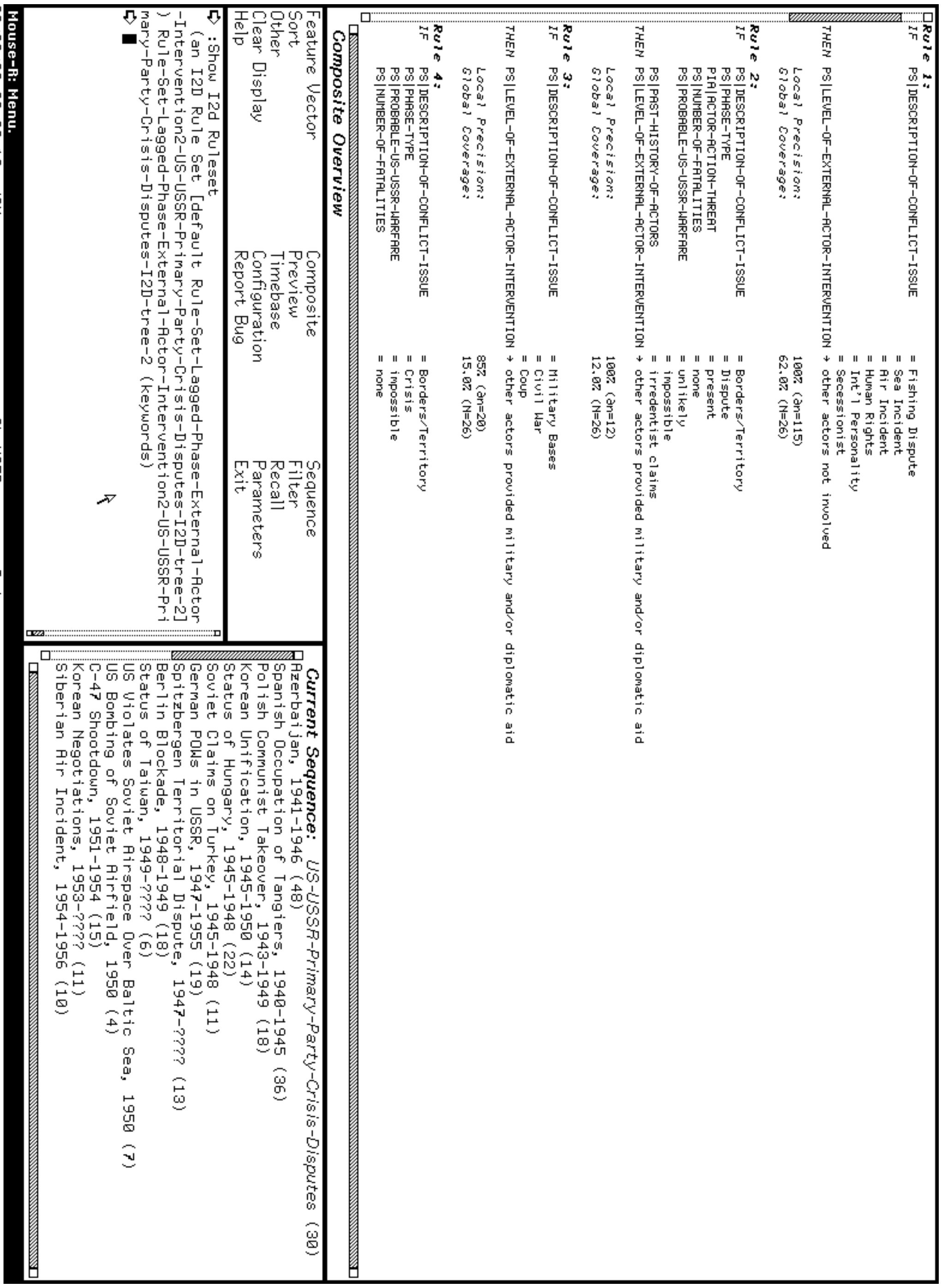


Figure 9: A control matrix, variables and parameters, that guide I^2D's rule learning.

Description the control matrix, Lagged-Phase-External-Actor-Intervention

```
Value Pairing Method:        I2D:SHERFACS-I2D-TIME-LAGGED-METHOD
Dependent Variable:          PS|LEVEL-OF-EXTERNAL-ACTOR-INTERVENTION
Independent Variables (25):  CO|PRESENCE-OF-MANAGEMENT-AGENT
                             CO|PROBABILITY-OF-US-USSR-WARFARE-IN-CASE
                             PIA|ACTOR-ACTION-ADVANTAGE
                             PIA|ACTOR-ACTION-APPEAL
                             PIA|ACTOR-ACTION-DIPLOMATIC-PROTEST
                             PIA|ACTOR-ACTION-DISPLEASURE
                             PIA|ACTOR-ACTION-DIVISION
                             PIA|ACTOR-ACTION-REFUSE-AID
                             PIA|ACTOR-ACTION-REPRESSION
                             PIA|ACTOR-ACTION-SNUB
                             PIA|ACTOR-ACTION-SUBVERSION
                             PIA|ACTOR-ACTION-SUPPORT
                             PIA|ACTOR-ACTION-THREAT
                             PS|ACTOR-POLITICAL-ALIGNMENT
                             PS|AGGRESSOR
                             PS|DESCRIPTION-OF-CONFLICT-ISSUE
                             PS|GENERAL-ISSUE-TYPE
                             PS|IDEOLOGICAL-FACTORS-INVOLVED
                             PS|NUMBER-OF-FATALITIES
                             PS|PAST-HISTORY-OF-ACTORS
                             PS|PHASE-TYPE
                             PS|PROBABLE-US-USSR-WARFARE
                             PS|RELIGIOUS-ETHNIC-RIVALRY
                             PS|THREAT-TO-GREAT-POWER-INTERESTS
                             PS|THREAT-TO-VALUES-OF-PARTIES
```

4 Creating Relational Graphs from Feature Vectors

4.1 Generating English Precis from SHERFACS

Using the infrastructure provided by the Feature Vector Editor, the dynamic codebook specified for SHERFACS was augmented with criteria for generating English case descriptions. These criteria were specified for 137 distinct variables out of SHERFACS's 275.[22] The generation criteria for these sentences conform to the model of syntactic parsing and semantic reference implemented by the RELATUS Natural Language System (Duffy, 1987; Mallery, 1988, 1991). Therefore, these sentences are guaranteed to parse and reference successfully in RELATUS. Using this method, a precis for the Cuban Missile crisis was synthesized (figure 10) from the SHERFACS feature-vector data for the case. The Butterworth (1979) summary for the case provides a point of contrast (figure 4).

4.2 Assessing the Generated Text

The two texts seem quite different. The synthesized SHERFACS text (figure 10) uses carefully specified English to avoid any processing difficulties for RELATUS. For example, repeating the phase and case for all the phase actions provides explicit constraint that allows the reference system to avoid confusions across phases, and thus, to find unique referents.[23] Apart from the syntactic and referential regularity, salient points of contrast with the Butterworth text (figure 4) are:

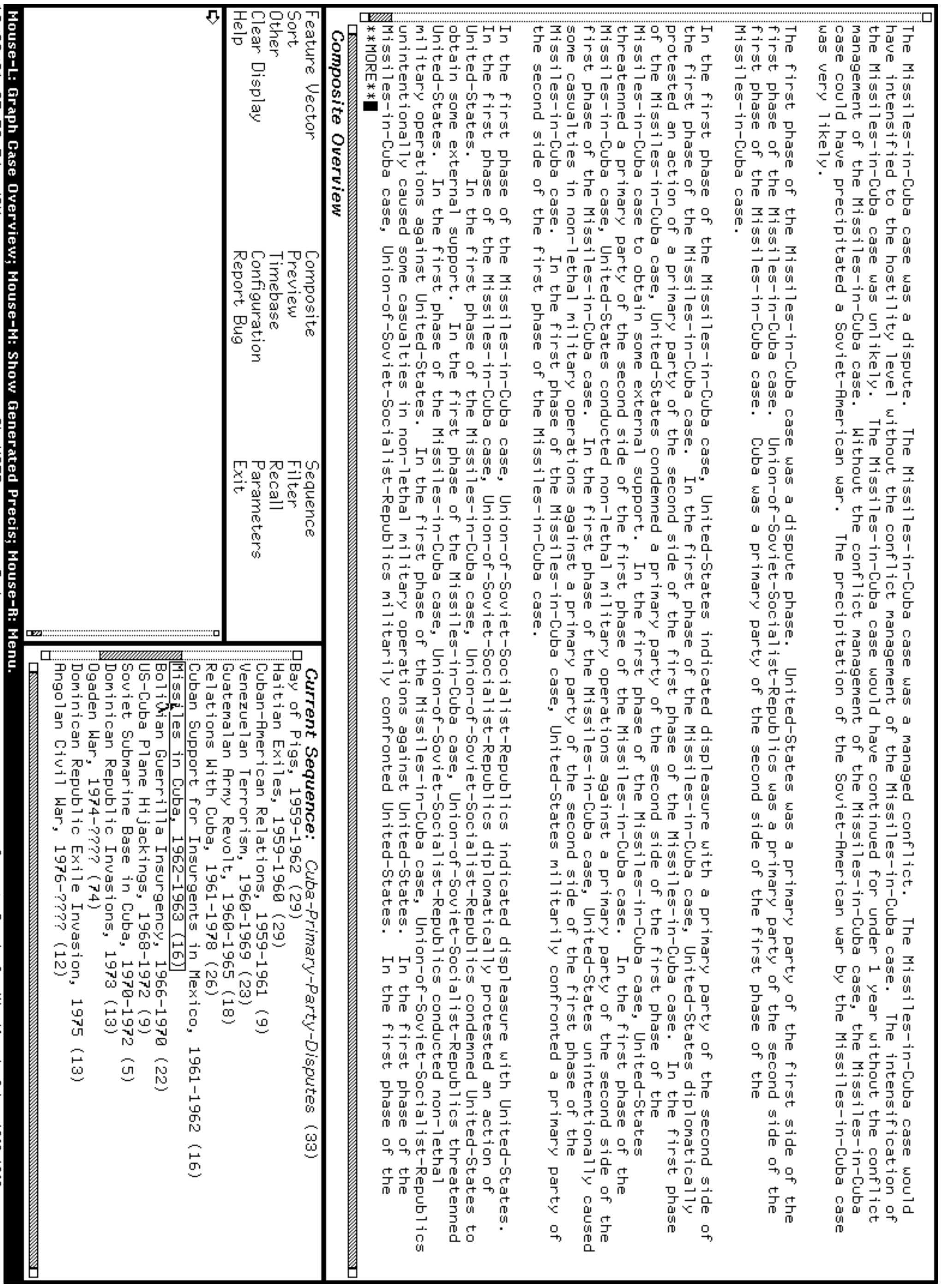
- There is greater attention to detail and relational structure in the Butterworth text, even though it is less extensive. For example, the Butterworth text mentions nuclear weapons explicitly and a naval quarantine (detail) as well as beliefs about the other party's beliefs (relational structure), whereas the SHERFACS text provides neither.
- The SHERFACS text makes the phase structure explicit.
- The SHERFACS text is more extensive, covering more actions.[24]
- The SHERFACS text more rigorously encodes a category system.
- The SHERFACS text is couched at a higher level of aggregation, reducing detail into more general categories.

[22] The variable type objects use a Common Lisp format string and arguments to write a stream of generated sentences to an editor buffer or file. This kind of text generation is *strictly ad hoc*. It is intended merely to convert SHERFACS cases into sentences, and not to provide a general model of natural language generation, for example, like the sentence generator in RELATUS.

[23] A cleaner way of accomplishing the same result is to use a simple context mechanism to switch the context from phase to phase and automatically insert constraints for every sentence referenced.

[24] This follows from the fact that the SHERFACS dataset draws on more extensive sources and is not space limited to a page or two of written text.

Figure 10: Generating a machine parsable narrative for the Cuban Missile Crisis.



Reading a SHERFACS text provides a pretty good feel for how an econometrics equation sees the world; it sees little nuance or detail through its limited degrees of freedom. But how will our AI system see the world through the lens of this text. The good points are:

- The conceptual regularity imposed by the discipline of a codebook constrains the universe of possible inductions, and therefore, simplifies the task of an inductive concept learner;
- The higher aggregation of information provides focus on the important information;
- The explicit nature of the text eliminates the implicit complexities of ordinary English usage;
- The explicit phase structure provides an important landmark for encoding temporal sequences and for reasoning about conflict processes.
- Together with the phase structure, the repertoire of actions provide a basis for identifying higher level patterns with the lexical classifiers (see section 4.4), and for inducing interesting categories.

4.3 Representing the Text

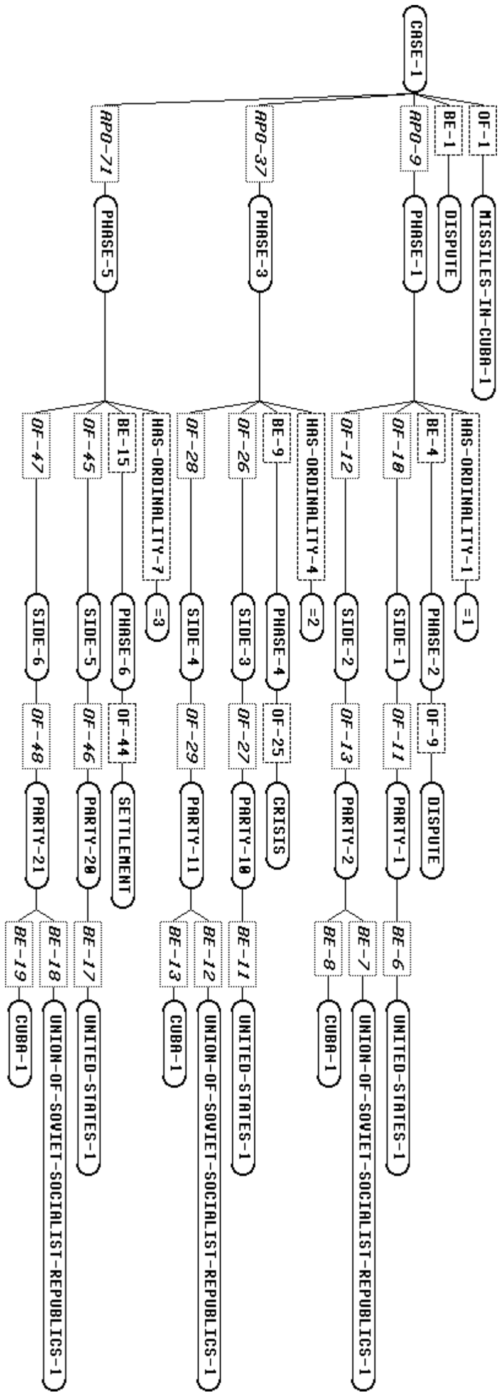
A textual description of the Cuban Missile Crisis was synthesized from SHERFACS data (figure 10), and then, parsed into a RELATUS representation in about one minute. This 6 page text (1600 words in 24 sentences) yielded 1229 nodes in the semantic representation, and an additional 369 after running recognizers for SHERFACS categories.[25] Aspects of the representation appear in figures 11, 14-19, which were automatically created using the Belief-System Examiner (figure 20).[26] The case overview in figure 11 shows the three phases in the Cuban Missile Crisis, their phase types (dispute, crisis, settlement), and the primary actors in each phase (the United States, the Soviet Union, and Cuba).

The RELATUS representation is primarily relational, expressing objects as the junctions of relations. For this reason, there are many more relations than objects. All nodes in the representation, whether relations or objects, are first-class objects, and therefore, may participate in relations as either their subject or their object. In the graphical displays of these knowledge structures, relation names are enclosed in rectangles while object names are enclosed in ellipses. In figure 11, "CASE-1" is an object node whose class is dispute, as shown by the "BE-1" relation linking it as a subject to the relational object, "DISPUTE." These ternary relations can be traversed bidirectionally,

[25] All text models in this article were created running RELATUS on a Symbolics XL1200 Lisp Machine with 4 megawords (144 megabytes) of physical memory.

[26] The Belief-System Examiner graphically displays portions of the semantic representation on a scrollable window. Starting from a seed node, the display process walks and graphs all semantic structure satisfying a set of constraints. For SHERFACS knowledge bases, some special constraint combinations were designed to graph dispute overviews (figure 11), actions in specific phases (not shown), and actions constrained by actor, dispute, and possibly phase (figures 14-19).

Figure 11: Graphical display of semantic structure that overviews the Cuban Missile Crisis.



either from subject to object, the *subject-relation* direction, or from object to subject, the *object-relation* direction. Thus, the course dashes of the rectangle surrounding "BE-1" indicate that it is a subject relation. In contrast, the fine dashes and the italic type face for "APO-9" mean that the node is an object relation, with "CASE-1" as the object and "PHASE-1" as the subject. These object relations are most easily understood by reading from right to left. Thus, "APO-9" means that "PHASE-1" is "a part of" of "CASE-1." They are presented as object relations because the displays are graphed from left to right, out from the seed node, here "CASE-1." Naturally, these semantic displays are more easily interpreted online using interactive facilities to obtain more detailed information, for example, clicking on a node to make the system display the node's frame representation or generate (speak) a sentence fragment for the node (figure 20).

The stock sentences used to create synthetic SHERFACS texts yield quite good semantic representations, in specific for the Cuban Missile Crisis and in general for any SHERFACS case. But, there are some serious problems:

- **Over Aggregation of Parties:** Careful attention to the actions in phases reveals anonymous parties as the objects of United States actions (see figures 14-19). The *anonymous party problem* arises whenever it is impossible to disambiguate the object of the action because there are two primary parties the side which is the object of the action – in this case the Soviet-Cuban side.

- **Overly General Actions:** Frequently, the action categories are too general and fail to distinguish information that could be relevant. For example, "discuss issues" (figures 14-16) does express the object of discussion, the specific issues. There are generally no details about actions.

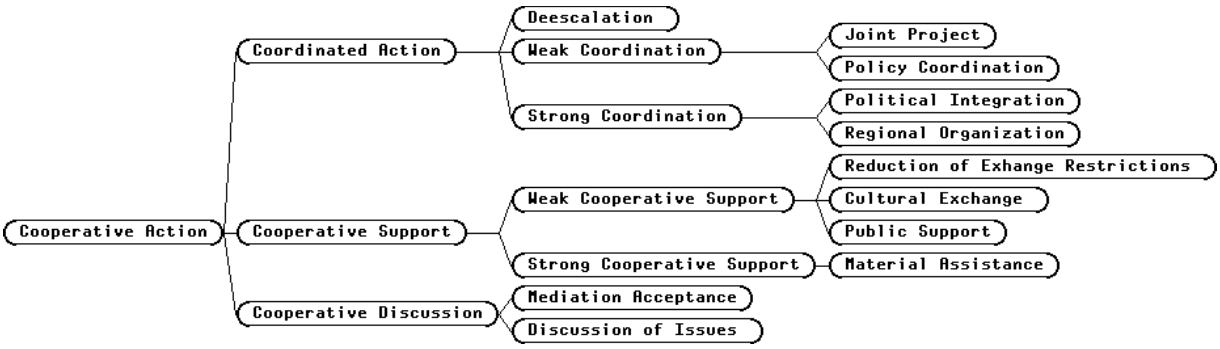
Both these problems are artifacts of a correlational mindset that prestructured the SHERFACS codebook, and its predecessors in the quantitative international relations tradition. Even though this aggregation conserves degrees of freedom for statistical approaches, it denies the lexical variation and structural detail that symbolic AI systems can use. RELATUS does not need so much pre-abstraction because it can employ constructive classification with the lexical classifier (section 4.4) to achieve the same result even as it retains the greater level of detail.[27] In short, over aggregation restricts the knowledge available, and thereby, impoverishes the conclusions that can be inferred and the generalizations that can be learned (Mallery, 1988, 1991). The aggregation of parties is quite serious because it prevents representation of exactly who did what to whom. In the present SHERFACS dataset, 448 out of 702 interstate disputes suffer from the anonymous party problem in at least one phase.

[27]As discussed in Mallery (1988: 52-54), a lexicalist natural language system can constructively classify the initial input to recognize categories and meaning equivalences. By precoding into semantic primitives, SHERFACS loses potentially relevant information and precludes reinterpretation of the initial information. This "coding problem" (Mallery, 1988: 55-56) could be avoided by representing significant elements of the source text from which the SHERFACS was coded.

4.4 Identifying SHERFACS Actions

A first step towards any useful analysis of knowledge bases for SHERFACS conflicts involves identifying the various SHERFACS actions represented and organizing them taxonomically. For this purpose, a SHERFACS lexical classification system was created.[28] This classification system contains 96 categories, divided between 60 base-level lexical recognizers and 36 taxonomically-organized categories. *Lexical recognizers* find instantiations of categories in RELATUS representations and label them as category instances. Normally, an entire tree of recognizers is run to find instances and instantiate the taxonomy, bottom up to the top of the tree. For example, once the lexical classifiers in figure 12 are run, the knowledge would encode the fact that "discussion of issues" is a "cooperative action" and it would taxonomically connect the instances of "discussion of issues" with the representation for the concept. Figures 12 and 13 show parts of the larger SHERFACS taxonomy of lexical recognizers.

Figure 12: Lexical classifier hierarchy for cooperative SHERFACS actions.

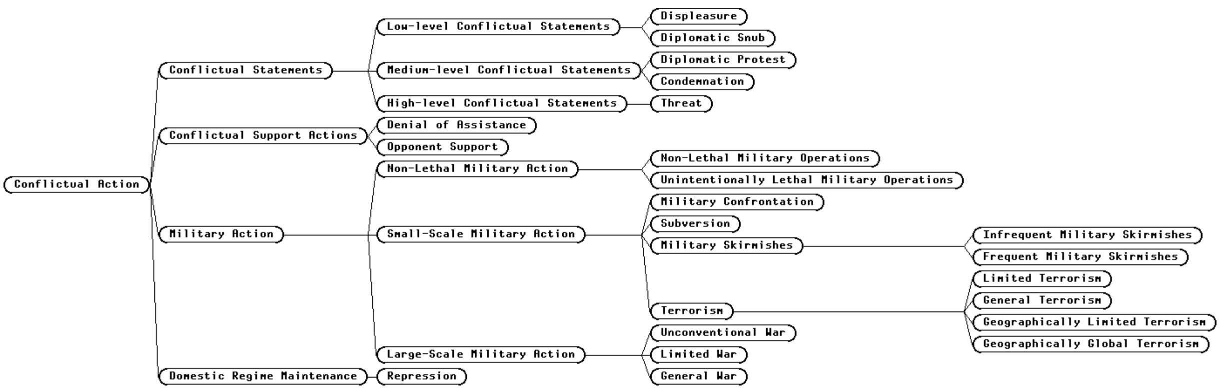


Once identified in the knowledge representation, SHERFACS categories can inform operations on the knowledge base. For example, they can provide the sources of constraint needed to graphically display actions by parties according to classes of actions. Figures 14-16 show the cooperative actions for the United States, the Soviet Union, and Cuba over the three phases (dispute, crisis, and settlement) of the Cuban Missile Crisis. Similarly, figures 17-19 show their military actions. The RELATUS Belief-System Examiner supports commands that allow the an analyst to generate displays such as these, or equivalent English sentences, by merely providing the conflict case and an action category. Because they are driven by the contents of the knowledge representation, these capabilities to present precisely focused information are easily adapted to answer different kinds of questions.

These examples concisely profile the significant changes visible in the SHERFACS knowledge representation for the Cuban Missile Crisis. Although military postures for the

[28]Mallery (1991) discusses lexical classification in greater detail.

Figure 13: Lexical classifier hierarchy for conflictual SHERFACS actions.



primary actors remained symmetrical, declining during the settlement phase, cooperation was asymmetrical in several ways. After an early cooperative posture by the Soviet Union in the first phase, the United States and the Soviet Union symmetrically increase cooperative actions in the face of, essentially, intransigence by Cuba. The United States begins cooperative discussion ("DISCUSS-2") during the crisis phase, and moves to weak coordination of its actions, engaging in joint projects ("ENGAGE-1") and coordinating policy ("COORDINATE-1") during the settlement phase. The Soviets were ready to cooperatively discuss the issue ("DISCUSS-1") during the first and second phases, and paralleled the United States move to weak coordination of its actions ("ENGAGE-1" and "COORDINATE-1") during the settlement phase. Interestingly, Cuba engages in minimal cooperative actions, cooperative discussions ("DISCUSS-4"), only during the crisis phase. Thus, Cuba was not a party to the settlement.

The military actions of the parties tell a perhaps more concrete story. The United States, the Soviet Union, and Cuba perform symmetrical military actions. During the dispute phase, they undertake the lowest intensity military actions, non-lethal military actions ("CONDUCT-1," "CONDUCT-2," "CONDUCT-3") and unintentionally cause some casualties ("CAUSE-1," "CAUSE-2," "CAUSE-3"). Their actions even reach small-scale military actions when they confront each other militarily, ("CONFRONT-1," "CONFRONT-2," "CONFRONT-3"). The profile of military actions remains the same in the crisis phase, and then in the settlement phase, military actions disappear.

The ability to automatically display aspects of these knowledge representations and to analyze them with the lexical classifiers show that referential integration operated correctly, and thus, that the texts were correctly represented. Beyond providing some glimpses into the knowledge representations, an important purpose of figures 14-20 is to demonstrate that the synthetic SHERFACS texts were indeed represented accurately in RELATUS.

Figure 14: United States cooperative actions in the Cuban Missile Crisis.

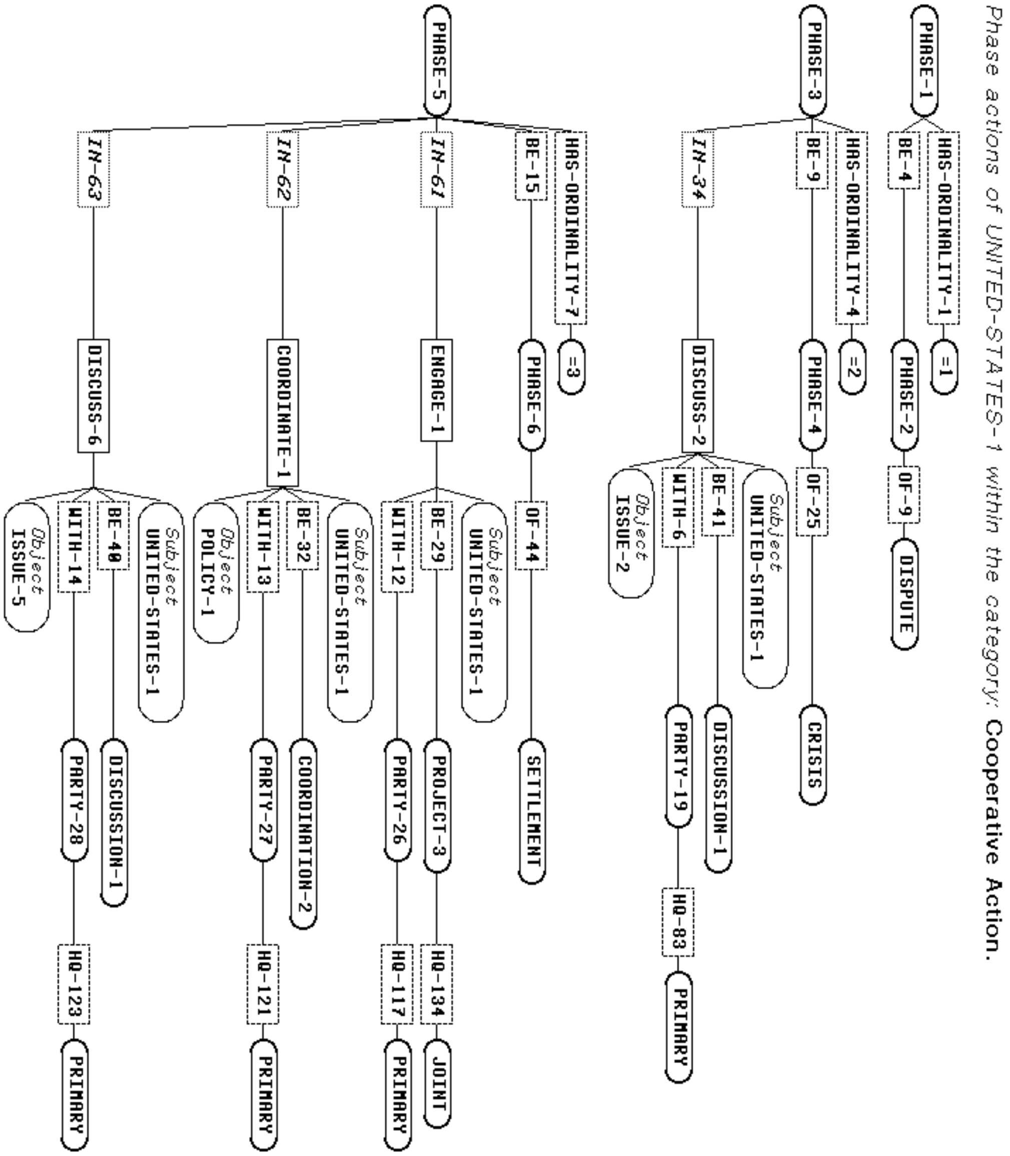


Figure 15: Soviet cooperative actions in the Cuban Missile Crisis.

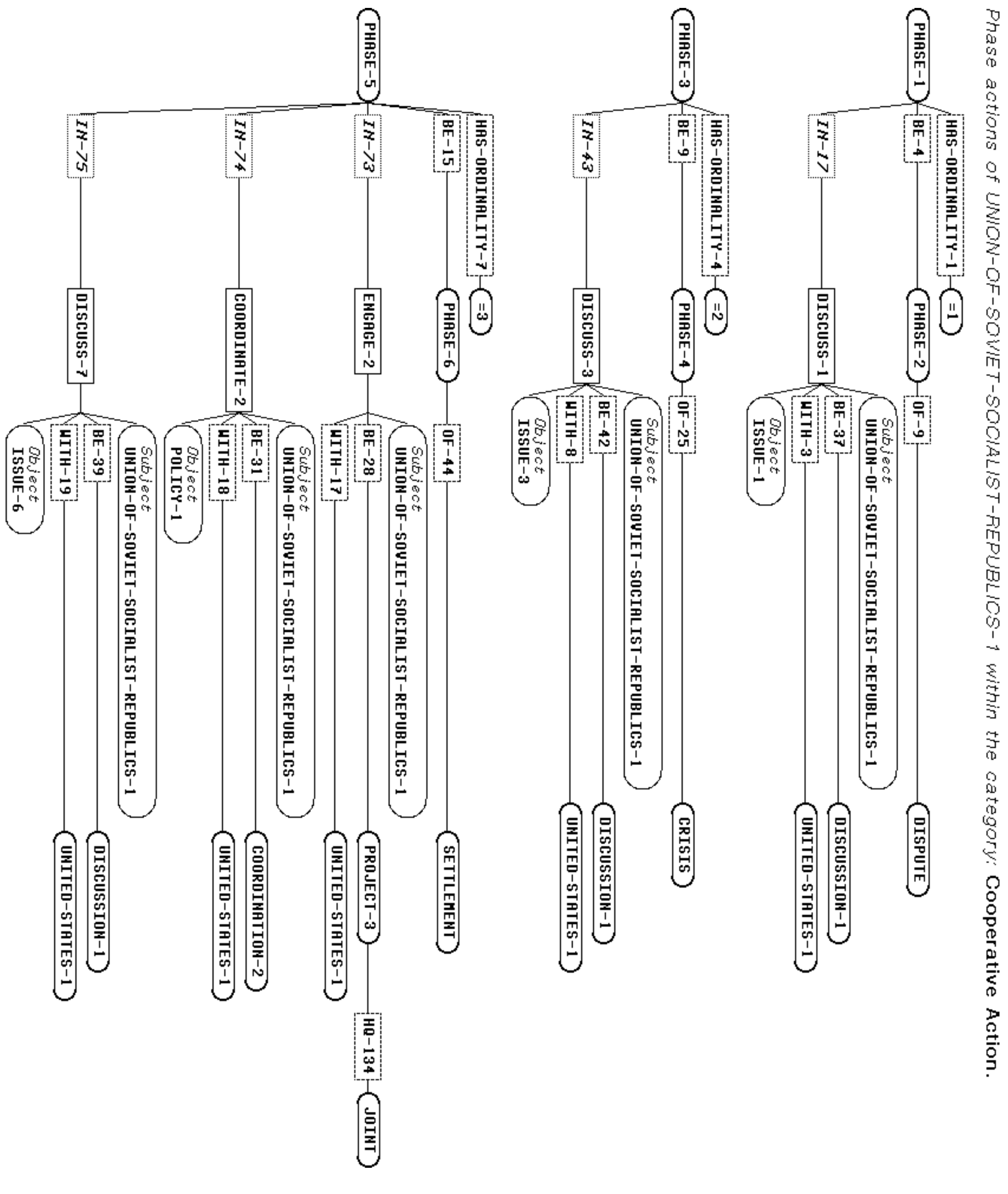


Figure 16: Cuban cooperative actions in the Cuban Missile Crisis.

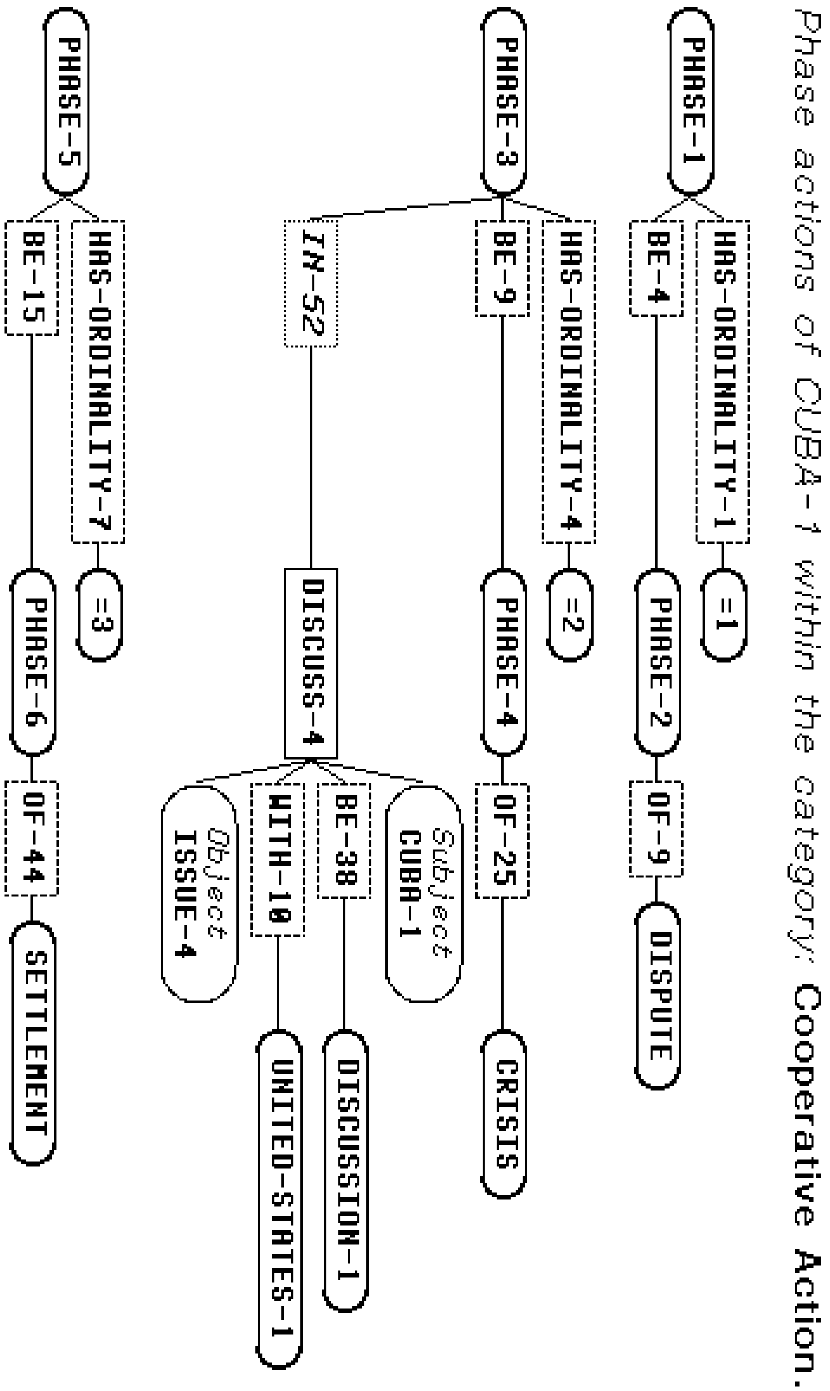


Figure 17: United States military actions in the Cuban Missile Crisis.

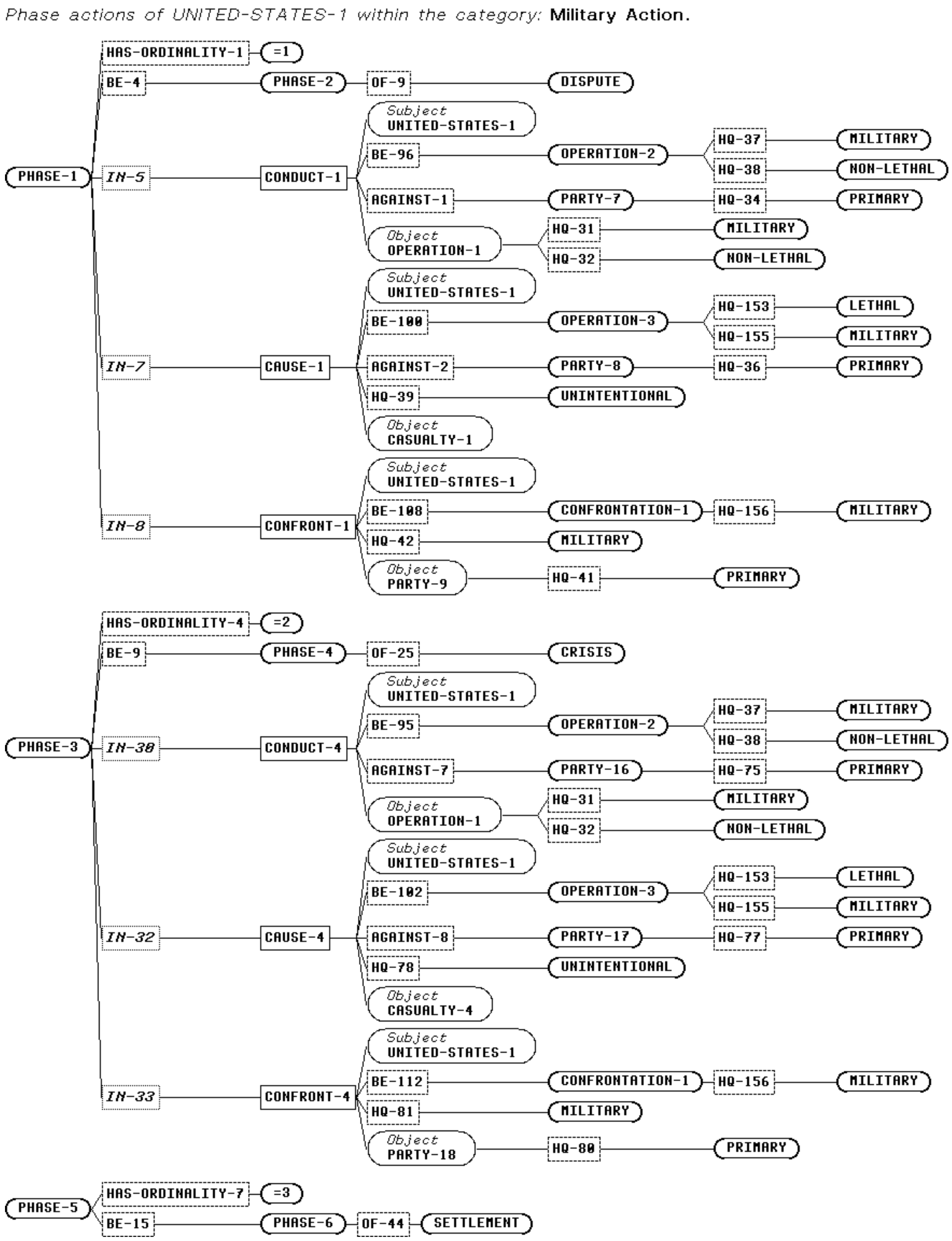


Figure 18: Soviet military actions in the Cuban Missile Crisis.

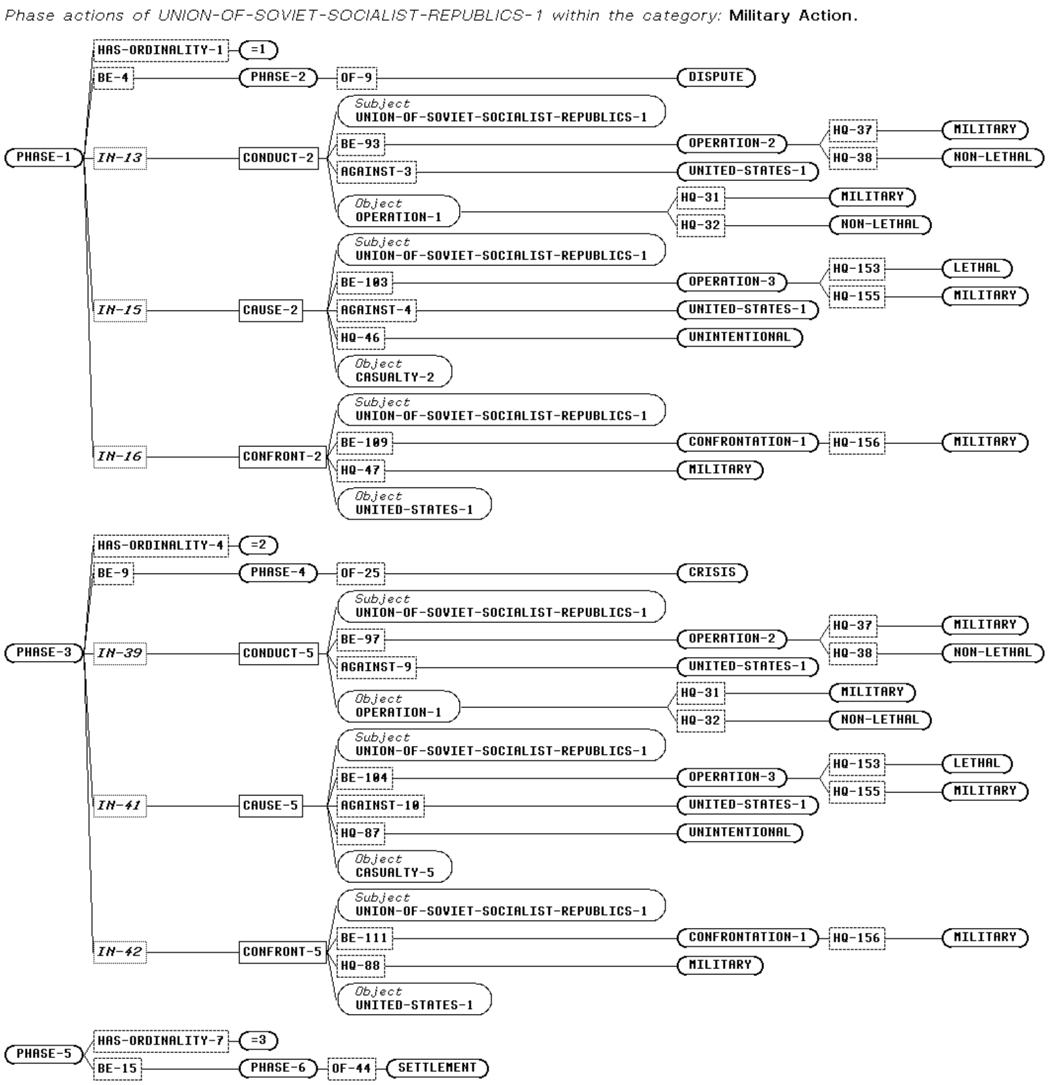
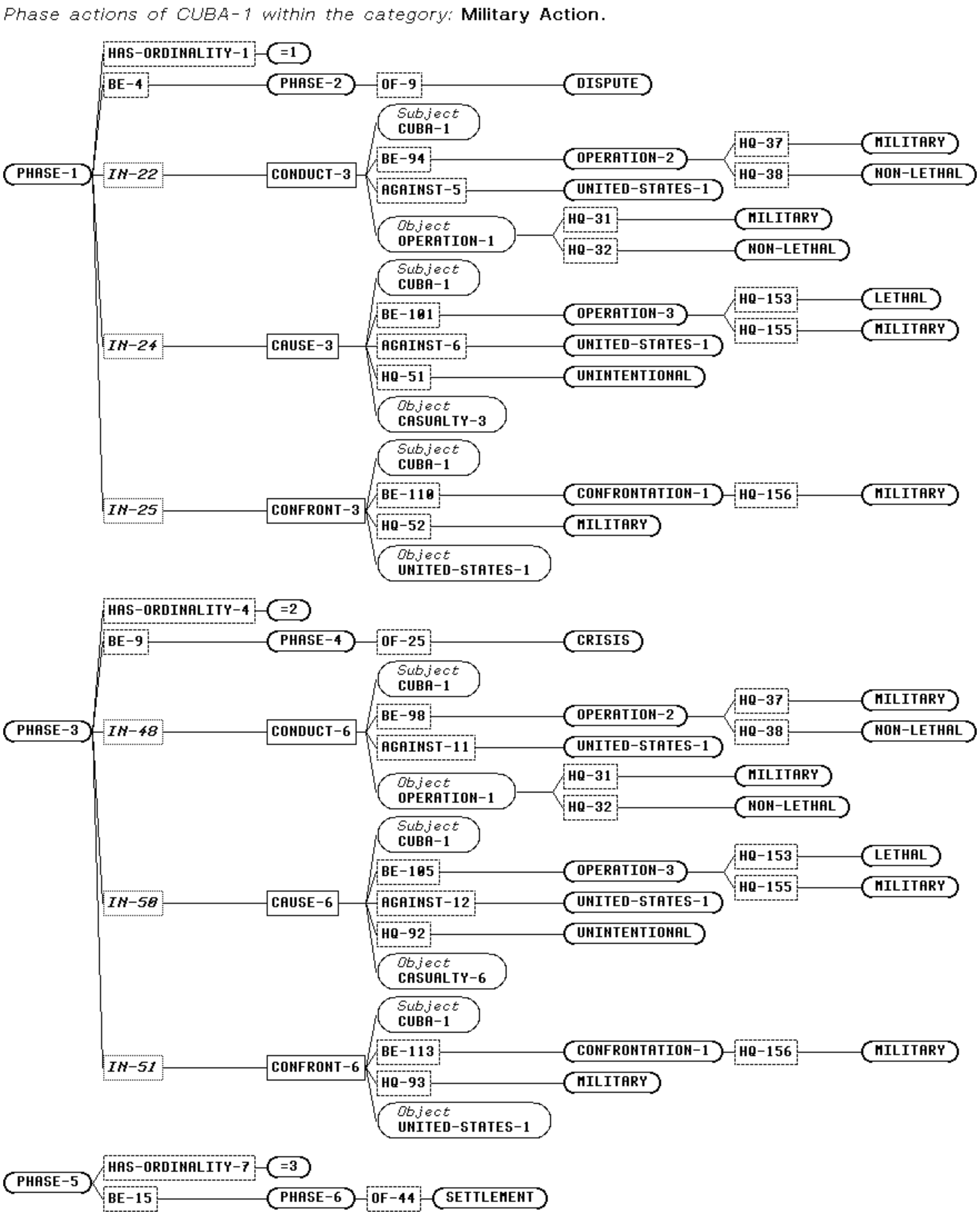


Figure 19: Cuban military actions in the Cuban Missile Crisis.



Once conflict descriptions are represented as knowledge structures in RELATUS, an analyst can elicit simple and precisely focused information. Since RELATUS can create knowledge representations for other English narratives about a case, the analyst could obtain a more highly specified representation by combining event-data codings with narrative accounts relevant to a case. Indeed, the hermeneutically inclined analyst could combine narratives expressing differing interpretations of a case in order to study the connections between interpretations and more tangible actions. The ability to broaden and deepen representations for a single case also carries over to the universe of cases represented.

4.5 Representing Multiple SHERFACS Cases

If one SHERFACS case can be automatically represented represented in RELATUS, it should be possible to represent all of the 1600+ SHERFACS cases (700 disputes and 900 quarrels). Early experiments show promise for RELATUS, and also, some areas for improvement. In one experiment, a 180-page synthetic text (50,000 words in 2459 sentences) was generated from 17 SHERFACS cases of United States interventions in Latin America. RELATUS parsed and represented this text to create a 32,000 node knowledge base in 54 minutes, and ran the SHERFACS lexical classifiers in 10.6 minutes! RELATUS operated surprisingly well considering that this size text is more than an order of magnitude larger than the texts previously read into RELATUS.

From a technical standpoint, paging performance was quite good.[29] During parsing and representation, RELATUS spent 20 of 54 minutes paging. For classification, referentially a much less local operation, the system spent 8.3 of 10.6 minutes paging. Such a large referentially-integrated knowledge base contains so much information at many nodes that the user can be overwhelmed when he display large nodes, for example 100-screen displays of a node. This problem was resolved by extending the RELATUS belief system examiner so that displayed information could be constrained by case, phase, or anything else a user might require. Representing these large texts also revealed a problem with how RELATUS stored dependencies on the sentences that create knowledge structures. By moving records of these dependencies from the knowledge representation into a compact, specialized datastructure, the size of knowledge bases was halved.

The RELATUS representation for the 17 Latin American texts contained 929 anonymous primary parties. Extrapolating to the total database size for the SHERFACS disputes based on these Latin American cases predicts about 3 million nodes,[30] of which 86,400 would be anonymous primary parties.[31] Anonymous primary parties do not just

[29] Paging, or the processor's data input/output, is a key issue for large-scale knowledge representations. Paging is the process of reading (writing) data into (from) physical memory from (to) virtual memory on its disk drive. As the ratio of physical memory to knowledge base size becomes small, a knowledge-based system may stop making progress due to *paging thrash* – a state in which the entire working set for a computation never gets into physical memory, and consequently, the processor can never perform the computation. Superior system designs for hardware, operating systems, and AI programs can avoid, or at least postpone, this debilitating condition.

[30] The actual size depends not just on the average size of individual descriptions but also on the referential integration (the shared structure) across international conflicts since 1945.

[31] Extrapolations like this are at best crude and at worst spurious; but they should fall within the

reduce the information content of the representation, *they seriously degrade performance as a function of knowledge base size.*[32]

Another experiment circumvented the problem of anonymous primary parties. A sequence of 244 diadic disputes without anonymous primary parties was partitioned from SHERFACS with a special filter and converted into a 600-page text (148,000 words in 10,000 sentences). This synthetic text was parsed and represented in RELATUS overnight, in about 10 hours (4.2 hours paging). The resulting knowledge base contained about 120,000 nodes. Next, a complete system of lexical classifiers for all represented SHERFACS variables was applied to the knowledge base, and 3.6 hours (1.3 hours paging) later, 17,000 additional nodes linked the SHERFACS taxonomy to instances in the representations of 244 disputes. Figure 20 shows a small piece of this taxonomy around the arbitration category. This knowledge base is now available for analysis using graph-based methods for induction (Winston, 1975, 1984) or analogy (Winston 1980, 1984; Mallery & Hurwitz, 1987). This technical result for SHERFACS data demonstrates that the small example of the Cuban Missile Crisis scales up smoothly by two orders of magnitude; large knowledge bases can be efficiently created for any suitable dataset and usefully analyzed on contemporary workstations.

5 Conclusions

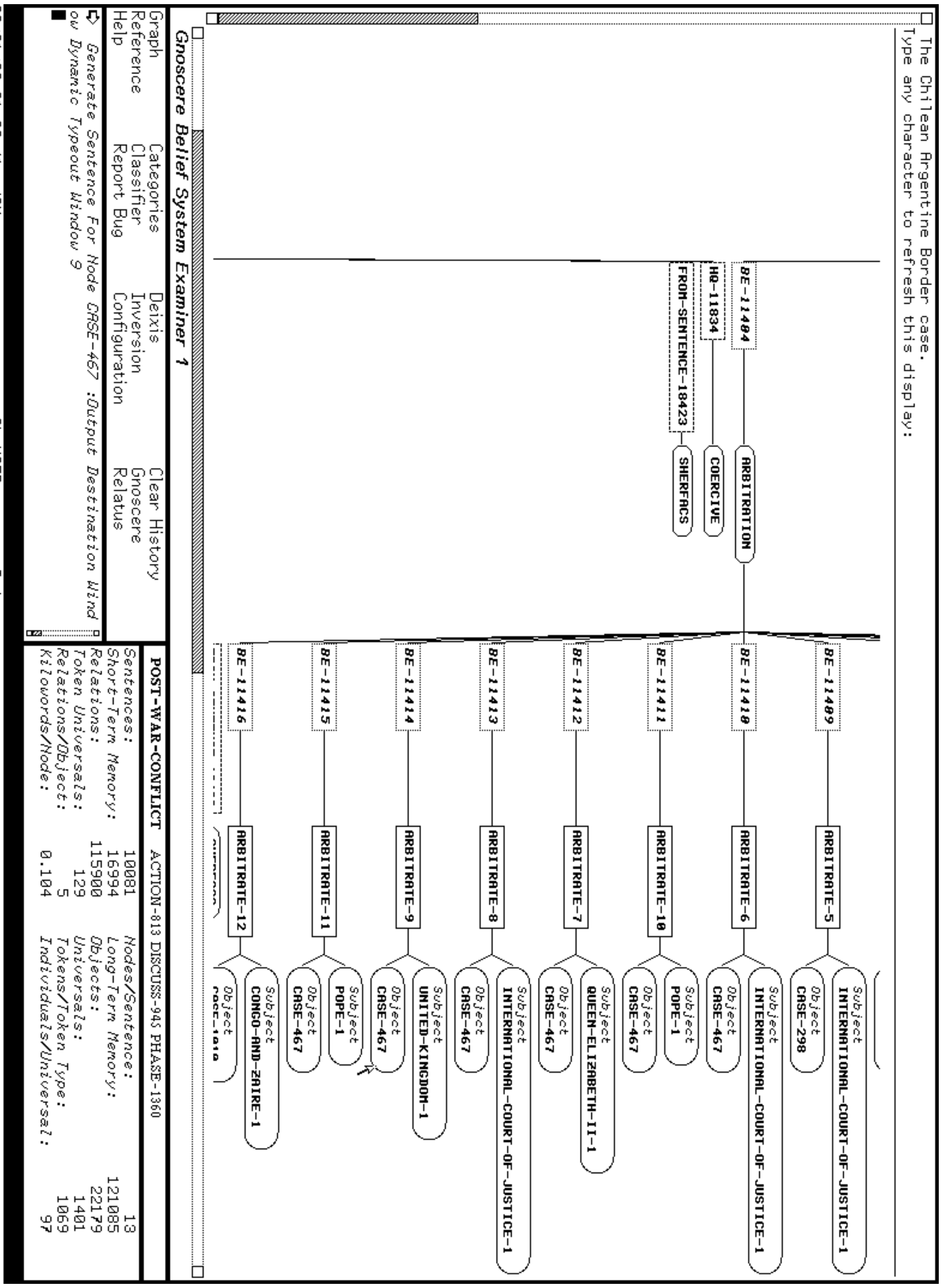
The Feature Vector Editor is general environment for examining, modifying, partitioning and analyzing feature-vector datasets for any domain. While the presentation-oriented interface makes the editor easy for people to use the generic protocols – which connect data, analytical algorithms, and the user interface – make the system readily extensible. New analytical techniques and new data can be added by merely making them conform to (or perhaps extend) the existing protocol. The ability to map feature-vector data into referentially integrated representations offers the possibility of applying symbolic AI techniques for learning and inference to knowledge encoded as feature vectors. The main thrust of these conclusion will be to suggest directions for further improvements in the Feature Vector Editor, the mappings into relational graphs, the collection of feature-vector data, and the generation of feature-vector data from relational graphs derived from natural language texts.

---

correct order of magnitude.

[32]By the end of the parsing and reference process, each mention of an anonymous party requires unusually long time to reference because each reference to one requires creation of a new instance, bloating the number of possible instances and the associated possibility spaces. During future references to an anonymous party, this forces the reference system to sort through needlessly large number of potential referents. However, the frequency of anonymous parties in the text for these 17 Latin American conflicts is higher than in the 700 SHERFACS disputes. Thus, these extrapolations somewhat overstate the problem.

Figure 20: Viewing some inferred taxonomic structure in the RELATUS Belief-System Examiner, the user has clicked on "CASE-467" to request generation of an English sentence fragment describing the node (seen in the upper left corner of the display).



5.1 Feature Vector Editor

Considerable development remains to realize the computational possibilities already opened for the Feature Vector Editor. New additions include:

- **More AI Algorithms:** Various other AI algorithms with promise for international relations research should be added. Learning algorithms based in feature vectors such as syntactic pattern recognition (Fu, 1982; Schrodt, 1984, 1987a), conceptual clustering, genetic learning (Holland, 1975, 1986; Schrodt, 1986, 1987a) and backpropagation (Rumelhart, Hinton, & Williams, 1986) could prove useful enhancements to the feature vector side of the system.

- **A Statistical Package:** The standard statistical techniques provide an important validational complement to AI techniques. The presence of statistical methods in the package would provide not just useful functionality but also a bridge for quantitatively oriented researchers who might like the new computational methods but wish to maintain their connection with statistical analysis.

- **Tacking Dependencies with a Non-linear Text Editor:** As large amounts of on-line text become accessible, there is a need to organize it for use and retrieval. Non-linear text editors (*e.g.*, Marshall, 1986; Goodman, 1987) with sophisticated indexation strategies provide the means to cope with this complexity. Once the coder locates text indicating a specific event coding, the coder should record a dependency between the coded value and the source text on which it depends. When a dataset contains a complete dependency record for every coding and every conclusion derived from the dataset carries dependency records back to the data on which it depends, it becomes possible to find, debate, and revise codings. It will also be possible to change the coding and observe the consequences for conclusions. Conversely, backtracking the support structure for incoherent or contradictory conclusions could find the source of erroneous codings. The emergence of mass-market non-linear text editing in Hypercard (Goodman, 1987) and the advent of permanent object systems capable of supporting large communities of users make this a timely project.

5.2 Data Development

Several conclusions concern data development in general.

- **Actor Attributes Over Time:** The present SHERFACS dataset has limited, if any, actor attributes. Thus, there is a pressing need for characterization of international actors by, at a minimum, their military, political, and economic capabilities. Socio-political histories for key internal actors within the current SHERFACS actors, such as political parties, individuals, or extra-legal entities, could help elucidate the decision-making process within actors. General economic and military indicators are needed to assess the action possibilities and power differentials between parties to conflicts. These characterizations are important for inducing of taxonomies of international actors and associating these categories with similarly induced international conflict profiles.

- **Spatio-Temporal Knowledge:** The general approach employed by Duffy (1991) can be extended to spatial knowledge. Since machine readable map coordinates are available, this information could be organized in a spatial indexer and combined with the existing temporal indexer to yield a spatio-temporal model of international relations. Geographic modeling would seem an important ingredient in testing various realist hypotheses. Spatio-temporal representation would allow identification of the conceptions of conflict "spread" or "diffusion" by conflict type or actor category. For example, the quarrels component of SHERFACS could be used to study the contagion properties of coups in the third world.

- **Alternate Classifications:** A codebook reflects a specific set of categories, whether formulated on the basis of explicit or implicit theory. The dataset is the history or interpretation according to those categories. To keep with a hermeneutic orientation (Mallery, Hurwitz, & Duffy, 1987) it is desirable to seek out datasets embodying a variety of category systems for international relations.

- **Greater Variety of Datasets:** The flexibility and generality of data in the Feature Vector Editor argue for integrating additional datasets to make them accessible to existing analytical algorithms in the environment. It will also provide a larger base of information from which reformulation may draw to create new datasets or knowledge bases.

- **Extending SHERFACS:** Although analysts can use the Feature Vector Editor to reformulate and combine datasets, those sharing the underlying conceptual apparatus of SHERFACS may wish to extend SHERFACS with additional events data. Even without addressing unit of analysis comparability questions, an analyst using an extended SHERFACS could explicitly justify the inclusion or exclusion of variables or work with a subset of cases that share variables across data studies.

5.3 Regenerating Relational Graphs

Large quantities of data are presently available in feature-vector form. The ability to regenerate relational graphs from feature-vector data can provide an important source of knowledge about international relations and the social world in general. Thus, the aim in these conclusions is to suggest ways in which new coding efforts can increase the utility of their data for AI modelers using symbolic approaches, especially ones based on relational graphs.

- **Interval Time Data for Phases:** The demonstrated ability to efficiently index (Duffy, 1991) and utilize interval time information (Hurwitz, 1991) means that time intervals for all event data should be coded. Because temporal indexation systems can easily handle resolution down to one second for the entire twentieth century, higher precision should be sought for interval time codings.

- **Less Aggregation:** Data users can always aggregate information but they can rarely disaggregate data. Computational modelers require as much disagregation as possible. The tradeoff between extensive and intensive datasets will become very real.

From research to date, some general conclusions about events datasets have emerged. Simply put, datasets aiming to recover relational structure need to attempt to answer the question: Who did what, to whom, when, where, how, and why? For an AI system to reason about actions it needs to know *how goals combine with function and structure* (Sztompka, 1974; Winston, 1984; Minsky, 1987). It also needs temporal and locational information to rule out spatio-temporally impossible interactions. A dataset for serious use in AI modeling needs to code relations, structure, function, and teleology. These are essential for models of planning and problem-solving. In the real phenomena, we know that collective cognitive entities engage in planning and problem-solving to achieve their goals in the face of competition and cooperation by other actors. Functional transformations of structures, including self (re-)production, are selected according to their plans and the availability of functional and structural resources. Datasets that fall short of this are inherently correlational in so far as they merely provide some markers that could be correlated with the structural determinants of action. The following general conclusions suggests steps to get beyond correlational datasets toward more cognitively and computationally adequate characterizations of the international system.

- **Code Relational Information:** Using entire feature vectors to code actions and relationships for actors may avoid repeating the same actor. Each action or relationship must be specified with at least a single bit. But, unlike the present SHERFACS, no attempt should be made to compress the object of the action.
- **Differentiate Actions:** Relational coding is not a substitute for differentiating actions. In SHERFACS for example, coding military confrontation in the Cuban Missile Crisis blurs the differences between naval quarantines and overland invasion.
- **Replace Correlational Variables:** The general maxim is to replace correlational variables wherever possible with relational variables grounded in a problem-solving perspective.
- **Assess Instrumentalities:** What purposes do actions serve? Coding the state of affairs that actions bring about, given enabling structures, is a crucial element in predicting what actions an actor may take. This calls, essentially, for causal explanations in international affairs.
- **Code Teleology:** Knowing the order of purposes of an actor, their goal priorities, provides one of the few ways of predicting action or inaction when the capability is present.[33]
- **Top-down Logical Closure:** The great detail that these coding maxims open raises the specter of intractability. To constraint data collection, coding should proceed top-down and logically close possibilities at each level of generality.

[33] Historians engage in great debates about the teleology that explains historical events. Although some might wish to side-step the thorny issue of coding teleology, the absence of teleological codings seriously limits the utility of the data for problem-solving simulations. A controversial coding is better than no coding at all. However, the reasons for the imputed teleological coding should be recorded for later criticism, and perhaps, revision. Certainty factors can help avoid reliance on dubious codings.

5.4 Reducing Relational Graphs to Feature Vectors

This article has discussed a general-purpose editor for feature-vector data and how it can help reconstruct a relational representation for SHERFACS disputes. The obvious complement to the research described in this article is a system that creates knowledge representations from unrestricted natural-language texts from which it automatically codes feature-vector datasets. Assuming the ability to create these knowledge representations in RELATUS, this task reduces to a system that recognizes the knowledge to code and maps it back into a feature-vector format.[34] The way in which RELATUS could do this is:

- **Lexically Classify Variables:** Apply a lexical classification system developed by the analyst to label all occurrences of every variable coded in the feature vectors. Since classification will make readily available the relational subjects and objects, they can be carried over to the feature-vector representation, avoiding the problem of anonymous or underspecified objects.

- **Semantic Inversion:** Like the facility RELATUS uses to graph semantic structure or generate sentences (Mallery, 1991), a new activity must be defined that walks classified knowledge structures and builds a feature vector from the associated structures.

The lexical classification phase is necessary because of the data reduction problem: Only a small part of the semantic representation is meaningfully carried over into the feature-vector representation. However, the lessons of this article suggest that it should retain relational information and as much detail as possible. The resulting feature-vector encoding will support regeneration of relational representations without the problems noted in this article. Thus, one might think of this process more as a compression strategy for relational information than as a source of events data – although it certainly would be. An important caveat on general idea of creating event data from natural language texts is:

- Until the AI problem is finally solved, a human will have to make the text conform to the referential processing model supported by the natural language system at the time (Mallery, 1991).

In the meantime, as various "sparse parsing" strategies mine unrestricted text for events data, human post-processing will need to compensate for any shortcomings in correctness and consistency.[35]

[34]This obvious, but technically non-trivial, idea arose during discussions at the MIT-DDIR Conference on Events Data in November 1987, and earlier at roundtables on developing textual archives for international relations at APSA 1987 and ISA 1987.

[35]As of this writing, computational linguists have yet to develop generally acceptable techniques for verifying the aspects of language that syntactic parsers cover correctly, but it is an active research area. Much progress in parser verification is essential before strong political conclusions can be drawn

6 Acknowledgments

This article was improved by comments from Roger Hurwitz. The international conflict data was provided by Frank L. Sherman, who also collaborated in the specification of a large number of English sentences (150) used to map SHERFACS cases into English and suggested some effective presentations of the data. Elements of this article were presented in 1988 to panels at *The Annual Meeting of the International Studies Association* and *the XIVth World Congress of the International Political Science Association.*

The Feature Vector Editor was conceived, designed, and implemented by the author. The temporal indexation system was designed and implemented by Gavan Duffy (1991). The interface between the Feature Vector Editor and the temporal indexation was collaboratively developed by Gavan Duffy and John C. Mallery. The I^2D rule learner was collaboratively developed and integrated into the Feature Vector Editor by John C. Mallery and Sigrid D. Unseld, while she was visiting fellow at the M.I.T. Department of Political Science between 1990 and 1992. The RELATUS Natural Language System owned by John C. Mallery and Gavan Duffy, each holding copyrights (1983-1992) to their respective components. Gavan Duffy designed and implemented the syntactic parser, categorial disambiguator, lexicon, and related utilities. John C. Mallery designed and implemented the semantic representation system, the sentential constraint poster, the noun-phrase quantification system, the reference system, the semantic inverter, the question answering system, sentence generator, the lexical classifier, the precedential reasoning system, and related utilities, including the RELATUS ZMACS mode.

The author was supported in part by a 1987 John D. and Catherine R. MacArthur Foundation grant for research on international security and arms control to the Center for International Studies of the Massachusetts Institute of Technology. Some recent work was partially supported by National Science Foundation Grant number SES-9025130 to the Data Development for International Relations Project at the University of Illinois Urbana-Champange under the M.I.T. subcontract number 91-118. This article describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the M.I.T. Artificial Intelligence Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-85-K-0124.

---

from events data acquired from unrestricted text using machine parsing techniques alone. In some special cases, stylized writing conventions used in textual data sources enable machine coding within acceptable standards of accuracy for not just lead sentences but perhaps even paragraphs. As text producers become more interested in supporting automatic parsing system downstream, there may be more upstream movement of text coding for machine parsing.

7 References

Abelson, H. and G. J. Sussman, *The Structure and Interpretation of Computer Programs*, Cambridge: M.I.T. Press, 1985.

Alker, H. R., Jr., "Polimetrics: Its Descriptive Foundations," In *The Handbook of Political Science, volume 7*, F. Greenstein and N. Polsby, eds, Reading: Addison-Wesley, 1975: 140-210.

Alker, H. R., Jr., G. Duffy, R. Hurwitz, and J. C. Mallery, "Text Modeling for International Politics: A Tourist's Guide to RELATUS," In *Artificial Intelligence and International Politics,* V. Hudson, ed, Boulder: Westview Press, 1991: 97-126.

Alker, H. R., Jr., and J. C. Mallery, "From Events Data To Computational Histories: A RELATUS-Based Research Programme In International Cooperation and Conflict," paper presented at *The 1988 Meeting of The International Studies Association*, 1988.

Breiman, L., J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees*, Belmont, CA: Wadsworth International Group, 1984.

Butterworth, R. L., with M. E. Scranton, *Managing Interstate Conflict, 1945-74*, New York: Knopf, 1976.

Butterworth, R. L., *Managing Interstate Conflict, 1945-79*, Final Report to ARPA, State College: Department of Political Science, Pennsylvania State University, February, 1980.

Duffy, G., *Language, Politics, and Method: Grounding a Computational Hermeneutic*, Cambridge: Doctoral Dissertation, Department of Political Science, M.I.T., August, 1987.

Duffy, G., "Time Space: Representing Historical Time for Efficient Event Retrieval," In *Artificial Intelligence and International Politics,* V. Hudson, ed, Boulder: Westview Press, 1991: 347-385.

Farris, L., H. R. Alker, Jr., K. Carley, and F. L. Sherman, "The Phase/Actor Disaggregated Butterworth-Scranton Codebook," Working Paper, Cambridge: Center for International Studies, M.I.T., 1980.

Fu, K. S., *Syntactic Pattern Recognition and Applications*, Englewood Cliffs: Prentice-Hall, 1982.

Goodman, D., *The Complete Hypercard Handbook*, New York: Bantam Books, 1987.

Haas, E. B., *Collective Security and the Future International System*, Denver: University of Denver, Monograph Series in World Affairs, 5(7), 1968.

Haas, E. B., R. L. Butterworth, and J. S. Nye, *Conflict Management by International Organizations*, Morristown: General Learning Press, 1972.

Haas, E. B., *Why We Still Need the United Nations*, Berkeley: Institute of International

Studies, University of California, 1986.

Holland, J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

Holland, J. H., "Escaping Brittleness: The Possibilities of General Purpose Algorithms Applied to Parallel Rule-Based Systems," In *Machine Learning: An Artificial Intelligence Approach, Volume II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Los Altos: Morgan Kaufman, 1986: 593-624.

Hurwitz, R., "Down The Up Staircase? Toward a Practical Theory of De-escalation," In *Timing and the De-escalation of International Conflicts*, L. Kriesberg and S. Thorson, eds, Syracuse: Syracuse University Press, 1991.

Hurwitz, R. and J. C. Mallery, "RELATUS User Notes," Mimeo, Cambridge: M.I.T. Artificial Intelligence Laboratory, May, 1987.

Keene, S. E., *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS*, Reading: Addison-Wesley, 1989.

Mallery, J. C., *Thinking About Foreign Policy: Finding an Appropriate Role for Artificially Intelligent Computers*, Master's Thesis, Cambridge: Department of Political Science, M.I.T., February, 1988.

Mallery, J. C., "Semantic Content Analysis: A New Methodology for The RELATUS Natural Language Environment," In V. Hudson, ed, *Artificial Intelligence and International Politics,* Boulder: Westview Press, 1991: 347-385.

Mallery, J. C., and R. Hurwitz, "Analogy and Precedent in Strategic Decision-Making: A Computational Approach," paper presented at *The 1987 Meeting of the American Political Science Association*, 1987.

Mallery, J. C., R. Hurwitz, and G. Duffy, "Hermeneutics," In , S. C. Shapiro, ed, *The Encyclopedia of Artificial Intelligence,* New York: John Wiley, 1987: 362-376.

Mallery, J. C., and F. L. Sherman, "Learning Historical Rules of Major Power Intervention in the Post-War International System," paper presented at *The 1993 Meeting of the International Studies Association*, Acapulco, Mexico March, 1993a.

Mallery, J. C., and F. L. Sherman, "Learning Rules of Phase Transition in the Presence of Conflict Managers: Identifying Historical Precedents in the Post-War International System," paper presented at *The 1993 Meeting of the International Studies Association*, Acapulco, Mexico March, 1993b.

Marshall, C., *Notecards Release 1.2 Reference Manual*, Palo Alto: Xerox Special Information Systems, Intelligent Systems Laboratory, Xerox Palo Alto Research Center, April 3, 1986.

Minsky, M., *The Society of Mind*, New York: Simon and Schuster, 1987.

Moon, D. A., "The Common Lisp Object-Oriented Programming Language Standard," In

W. Kim and F. Lochovsky, eds, *Object-Oriented Concepts, Applications, and Databases*, Reading: Addison-Wesley, 1988.

Quinlan, J. R., "Induction Over Large Data Bases," Technical Report No. HPP-79-14, Palo Alto: Computer Science Department, Stanford University,, 1979.

Quinlan, J. R., "Learning Efficient Classification Procedures and Their Application to Chess End Games," In , R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds, *Machine Learning: An Artificial Intelligence Approach* Palo Alto: Tioga, 1983: 463-482.

Rao, R., W. M. York, and D. Doughty, "A Guided Tour of the Common Lisp Interface Manager," In *Lisp Pointers*, 4(1991).

Rowley, S., H. Shrobe, R. Cassels, and W. Hamsher, "Joshua: Uniform Access to Heterogeneous Knowledge Structures or Why Joshing is Better than Conniving or Planning," In *Proceedings of The 1987 National Conference on Artificial Intelligence*, Los Altos: Morgan Kaufman, 1987: 48-52.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," In D. E. Rumelhart, J. L. McClelland, *et al.*, *Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Volume 1: Foundations*, Cambridge: M.I.T. Press, 1986: 318-362.

Schrodt, P. A., "Artificial Intelligence and International Crisis: The Application of Pattern Recognition to The Analysis of Event Sequences," paper presented at *The 1984 Meeting of the American Political Science Association*, 1984.

Schrodt, P. A., "Set Prediction of International Behavior Using a Holland Classifier," paper presented at *The 1986 Meeting of The International Studies Association*, 1986.

Schrodt, P. A., "Pattern Matching, Set Prediction, and Foreign Policy Analysis," In S. J. Cimbala, ed, *Artificial Intelligence and National Security*, Lexington: Lexington Books, 1987a: 89-107.

Schrodt, P. A., "Classification of Interstate Conflict Outcomes using a Bootstrapped CLS Algorithm," paper presented at *The 1987 Annual Meeting of The International Studies Association*, Washington, D.C., April, 1987b.

Sherman, F. L., "Quarrels and Disputes: An Expansion of The Phase/Actor Disaggregated Butterworth-Scranton Codebook," State College: Department of Political Science, Pennsylvania State University, mimeo, December, 1985.

Sherman, F. L., *Partway to Peace: The United Nations and The Road to Nowhere*, State College: Doctoral Dissertation, Department of Political Science, Pennsylvania State University, 1987.

Sherman, F. L., "Four Major Traditions of Historical Events Research: A Brief Comparison," paper presented at the M.I.T./D.D.I.R Conference on "New Directions for Storing, Indexing, Retrieving, Coding and Analyzing Information on International Events," Cambridge: Center for International Studies, M.I.T., November, 1987.

Sherman, F. L., "SHERFACS: A New Cross-Paradigm, International Conflict Dataset," paper presented at *The 1988 Meeting of The International Studies Association*, 1988.

Sherman, F. L., J. C. Mallery, S. D. Unseld and G. Duffy, "Grammars of Conflict and Co-operation in the Post-War International System," paper presented at *The 1992 Meeting of the International Studies Association*, 1992.

Symbolics, Inc., *Reference Manual for Symbolics Computers*, Cambridge: Symbolics, Inc., 1986. Revised 1987, 1988.

Sztompka, P., *System and Function: Toward a theory of Society*, New York: Academic Press, 1974.

Unseld, S., and J. C. Mallery, "Interaction Detection in Complex Datamodels," AI Memo 1298, Cambridge: M.I.T. Artificial Intelligence Laboratory, May 1992.

Winston, P. H., "Learning Structural Descriptions from Examples," In P. H. Winston, ed, *The Psychology of Computer Vision*, New York: McGraw-Hill, 1975: 157-210.

Winston, P. H., "Learning and Reasoning by Analogy," In *Communications of the ACM*, December, 23(1980).

Winston, P. H., *Artificial Intelligence*, Reading: Addison-Wesley, 1984.