Research in Text Theory
Untersuchungen zur Texttheorie

Editor
János S. Petöfi, Bielefeld

Advisory Board
Irena Bellert, Montreal
Maria-Elisabeth Conte, Pavia
Teun A. van Dijk, Amsterdam
Wolfgang U. Dressler, Wien
Peter Hartmann, Konstanz
Robert F. Longacre, Dallas
Roland Posner, Berlin
Hannes Rieser, Bielefeld

Volume 3

Text Processing
Textverarbeitung

Papers in Text Analysis and Text Description
Beiträge zu Textanalyse und Textbeschreibung

Edited by
Wolfgang Burghardt
and Klaus Hölker

Walter de Gruyter · Berlin · New York
1979
SHELDON KLEIN  
John F. Aeschlimann  
David F. Balsiger  
Steven L. Converse  
Claudine Court  
Mark Foster  
Robin Lao  
John D. Oakley  
Joel Smith

Automatic Novel Writing: A Status Report*

Abstract

Programmed in FORTRAN V on a Univac 1108, the system generates 2100 word murder mystery stories, complete with semantic deep structure, in less than 19 seconds.

The techniques draw upon the state of the art in linguistics, compiler theory, and micro-simulation. The plot and detailed development of events in the narrative are generated by a micro-simulation model written in a specially created, compiler-driven simulation language. The rules of the simulation model are stochastic (with controllable degrees of randomness) and govern the behavior of individual characters and events in the modelled universe of the story. This universe is represented in the form of a semantic deep structure encoded in the form of a network—a directed graph with labelled edges, where the nodes are semantic objects, and where the labelled edges are relations uniting those objects. The simulation model rules implement changing events in the story by altering the semantic network. Compiler or translator-like production rules are used to generate English narrative discourse from the semantic deep structure network (the output might be in any language). The flow of the narrative is derived from reports on the changing state of the modelled universe as affected by the simulation rules. Nodes of the semantic network may be atoms, classes, or complex predicates that represent entire subportions of the network. Atom nodes and relations are linked to expression lists that may contain lexical stems or roots that are available for insertion into trees during the generation process. (Low level transformations convert the roots into appropriately inflected or derived forms. High level transformations mark the tree for application of the low level ones.) These expression lists may also contain semantic network expressions consisting of objects and relations which may themselves be linked to expression lists, thereby providing the generator with recursive expository power. An atom node may also function as a complex predicate node with status that may vary during a simulation.

Class nodes may refer to lists of object nodes, and the complex-predicate nodes can be linked to pointers to sub-portions of the network that includes themselves, allowing them to be recursively self-referential. (This would permit generation of sentences such as “I know that I know that—(sentence’)."

We are also testing a natural-language meta-compiling capability—the use of the semantic network to generate productions in the simulation language itself that may themselves be compiled as new rules during the flow of the simulation. Such a feature will permit one character to transmit new rules of behavior to another character through conversation, or permit a character to develop new behavior patterns as a function of his experiences during the course of a simulation. This feature, combined with the complex-predicate nodes helps to give the system the logical power of at least the 2nd order predicate calculus.

Theoretical motivations include an interest in modeling generative-semantic linguistic theories, including case grammar and presuppositional formulations. The dynamic time dimension added to the semantic deep structure by the simulation makes it possible to formulate more powerful versions of such theories than now exist.

Table of Contents

1.0. Introduction ............................................. 340
2.0. Historical Background and Related Research ................. 341
3.0. Semantic Network & Discourse Generation System ............ 342
4.0. Highlights of the Simulation Language ..................... 349
5.0. Novel Writer Features and Futures ....................... 357
5.1. Style Control ........................................... 358
5.2. Private Semantic Universes for Individual Characters ....... 359
5.3. Simulation of Simulations: Look-Ahead, Planning, Time Travel and Dreams ........................................ 359
5.4. Semantic Parsing ........................................ 360
5.5. Linguistic and Behavioral Learning: Self-Modifying Behavior and Natural Language Meta-Compiling .................. 360
6.0. Significance for Linguistics, Sociolingustics and the Behavioral Sciences in General ................................ 360
7.0. Appendix ................................................. 361
7.2. Transformations ............................................ 361

* Portions of this research were sponsored by National Science Foundation Grant No. GS 2595, and by the Wisconsin Alumni Research Foundation.

Abridged version of University of Wisconsin Computer Sciences Department Technical Report No. 183, July 1973, University of Wisconsin, Madison, Wisconsin 53706, USA.
1.0. Introduction

The novel writer described herein is part of an automated linguistic tool so powerful and of such methodological significance that we are compelled to claim a major breakthrough in linguistic and computational linguistic research. What is emerging is a system for modelling human linguistic and social behavior through time, including the transmission of language and complex patterns of social behavior across generations, through the mediation of language, and according to the dictates of any generative semantic linguistic theory currently in existence, including the case grammar of Fillmore, the presuppositional model of Lakoff, and the 1972 semantic theory of Katz, as well as theories of far greater power than any heretofore suggested.

The key components are a compiler driven simulation language system that manipulates events in the form of a semantic deep structure network notation, and which has the power of at least the 2nd order predicate calculus, and a linguistic generative system that can map the semantic deep structure notation into any natural language using grammars within the framework of a variety of linguistic theories, and which can also generate productions in the language of the simulation system itself, providing a natural language meta-compiler capability.

The novel writer described here is a particular application and testing of the more general system in progress. While the computer generated stories contained in the appendix are in English, they might as easily have been produced in any natural language without alteration of the simulation rules or the semantic deep structure. The simulation system that generated the plot can be used to generate any kind of human behavior, within any time scale, with any level of detail, and all within the framework of any theoretical model of behavior that a researcher may care to formulate.

For the novel writer, the simulation language was used to describe the potential behavior of a set of characters in a partially random set of situations. The determinisitic aspects guarantee a murder story within the context of a weekend houseparty, arising from possible motives of greed, anger, jealousy or fear. The particular murderer and victim may vary with the random number source and with the particular specification of character traits prior to the generation. The motives for murder arise as a function of events during the course of the generation of the story.

The rules of the simulation model are stochastic, with controllable degrees of randomness, and govern the behavior of individual characters in the modelled universe of the story. This universe is represented in the form of a semantic deep structure that is encoded in the form of a network, a directed graph with labelled edges, where the nodes are semantic objects and where the labelled edges are relations uniting those objects.

The simulation rules alter events in the universe as a function of the passage of time. As the simulation progresses, the newly created events serve as the semantic deep structure input to a generative device that uses compiler or translator like rules to generate discourse in the selected natural language. The flow of the narrative is derived from successive reports on the changing state of the modelled universe.

Much of the semantic, behavioral and presuppositional information can be incorporated in the behavioral simulation rules as well as in the semantic deep structure network. The rules and the deep structure are intimately related in a number of ways. As indicated, the rules can alter the universe, and yet the rules themselves can be represented in the semantic deep structure; and the rules can be used to generate sentences in the simulation language itself, thus permitting the modification of old behavior patterns or the creation of new ones. The ability to partition the semantic deep structure into static and dynamic components, coupled with the higher order predicate calculus power permits the formulation of behavioral linguistic theories and models more powerful than any currently in existence.

In the balance of this paper we shall briefly cite relevant literature and then proceed to a discussion of the system in its novel writing aspect. The appendix includes a complete listing of the simulation language program that generated our several 'novels', and a sample story, length 2100 words, produced by the program complete with semantic deep structure and English text. We also include interesting passages from three other versions of the murder mystery derived from the same basic simulation program.

We note here that the novel writing system, which is operational on a Univac 1108 computer, uses approximately 75,000 words of storage space, of which 35,000 is required for the control mechanisms of the simulation system, 20,000 for the simulation language compiler and 20,000 for the discourse generation component. Approximately 50% of this space is used for data structures. The program generates 2100 word stories, complete with semantic deep structure descriptions as well as text, in less than 19 seconds. The system is programmed in FORTRAN V.

2.0. Historical Background and Related Research

The direct antecedents of this research arise from a three-fold base: our work on dependency approximation to semantic networks in discourse generation

Other work involving automated semantic networks includes that of Quillian, 1966, Schank 1969, 1972, Schank & Rieger, 1973, Mel'chuk, 1970 (the list is non-exhaustive).


Work involving natural language compiling into semantic representations, inference languages or simulation languages includes (in addition to our own) Kellogg, 1968, Heidorn, 1972, Simmons (in preparation), as well as Green & Raphael, ibid and Coles, ibid (again the list is not exhaustive).

3.0. Semantic Network & Discourse Generation System

The following explication is quoted from Klein, 1973, pp. 3–11:

Semantic Network

The semantic network consists of objects and relations linking those objects. The object nodes and relations have no names in themselves, only numbers. But they are linked to lexical expression lists that contain lexical variants as well as other expression forms. In examples of semantic network representations of deep structures bracketed lexical items selected from the associated lexical lists are provided with the objects and relations for convenience in reading. As an example consider the discourse:

“The man in the park broke the window with a hammer.”

“John knows that.”

The deep structure network representation might resemble:

\[
\begin{array}{c}
0 \text{(man)} \rightarrow \text{R (break; -time)} \rightarrow 0 \text{(window)} \\
| \\
R \text{(in)} \rightarrow \text{R (with)} \\
| \\
0 \text{(park)} \rightarrow 0 \text{(hammer)}
\end{array}
\]

(where the -1 represents a time earlier than present)

But the actual representation of the semantic deep structure is more subtle and has properties not obvious in this example illustration. The network is actually composed of semantic triples. A semantic triple can consist of any sequence of 2 or 3 objects and relations. Every object in the system has a unique number or address. Every triple in the system also has a unique number and is also associated with its time of creation. The network is actually stored in the form of a hash table, wherein the actual semantic network is implied and computable rather than overtly listed. The time of creation of each triple makes the application of tense transformations easy: the simulation system maintains a clock representing ‘now’. Accordingly the relative time sequence among deep structure triples is readily computable, and serves as data for generation of surface structure expression of tense, etc. The actual representation of this sentence is closer to:

1. 0 (man) \rightarrow R (break, -time) \rightarrow 0 (window)

\[
\begin{array}{c}
R \text{(break; -time)} \rightarrow R \text{(with)} \rightarrow 0 \text{(hammer)}
\end{array}
\]

2. 0 (man) \rightarrow R (in) \rightarrow 0 (park)

where the second triple in 1. is not actually listed separately; multi-place predicates are indexable through the primary triple.

It is worth repeating that the objects and relations are actually numbered locations with links to other objects and relations. They contain no associated content expression form other than what appears on their lexical expression lists that are also linked to them. However, a lexical expression list may contain other data than just pointers to lexical stems in a dictionary. These items include semantic triples that are not in the network (for expression of idomatic type structures) and pointers to triples that are in the network.

The objects and relations in these triples have their own links to their own lexical expression lists. The lexical expression list of an object or a relation may contain pointers to triples in the network that include triples of which it is a member.

Consider now the second sentence of the sample discourse:

“John knows that”.

encoded in the semantic network as,

3. 0 (John) \rightarrow R (know) \rightarrow 0 (that)

The 0 (that) is a complex predicate object. Its lexical expression list contains pointers to semantic triples 1 and 2. The representation could be self-referential; if the lexical expression list of 0 (that) contained a pointer to triple 3, the network would represent a message approximating:

“John knows that he knows that the man in the park broke the window with a hammer.”

This feature helps to give the system the logical power of the 2nd order predicate calculus (at least). Complex logical predicates are represented with such predicate nodes linked by logical connective relations. Thus the statement, if A then B, where A and B are complex bodies of semantic discourse representing large portions of the semantic network, is represented simply as, 0 (A) \rightarrow R \text{(implication)} \rightarrow 0 \text{(B)}, where 0 (A) and 0 (B) each point to lists of semantic triples that may also be of the same time- predications linking
predicate objects that have pointers to triples on their lists. (Always these lists may contain self-referential pointers - serving to justify the claim that the system has the power of at least the 2nd order predicate calculus.) (Other logical devices involving classes of objects and quantifiers are associated with the simulation language manipulates and modifies the semantic network.) A final schematic of the relevant data structures:

![Diagram of data structures]

**Generative Rules: surface structure // semantic network**

The phrase structure rules in the system are part of more complex rules that compile the semantic deep structure network from surface structure - and which also serve the function of generating surface structure from the network. The general form of such a rule is: phrase structure rule // canonical form of semantic triple, where the phrase structure rules are of the usual sort, where linked mappings between nodes in the right half of phrase structure and elements in the network specification are indicated. Strictly speaking the network specification need not be limited just to a semantic triple, as will be seen in the section on inference of rules. Some examples of rules:

\[
\begin{align*}
S & \rightarrow \text{NP } \text{VP} \quad // \quad 0 - R \\
\text{NP} & \rightarrow \text{NP} \quad \text{PP} \quad // \quad 0 - R \\
\text{NP} & \rightarrow \text{Det} \quad \text{NPP} \quad // \quad 0 \\
\text{NPP} & \rightarrow \text{adj} \quad \text{NPP} \quad // \quad 0 - R - 0 \\
\text{NPP} & \rightarrow \text{terminal} \\
\text{VP} & \rightarrow \text{VPP} \quad \text{PP} \quad // \quad R - R \\
\text{VPP} & \rightarrow \text{V} \quad \text{NP} \quad // \quad R - 0 \\
\text{VPP} & \rightarrow \text{V} \quad \text{NP} \quad // \quad R - 0 \\
\text{V} & \rightarrow \text{terminal} \\
\text{PP} & \rightarrow \text{prep} \quad \text{NP} \quad // \quad R - 0 \\
\text{PP} & \rightarrow \text{prep} \quad \text{NP} \quad // \quad R - 0 \\
\text{prep} & \rightarrow \text{terminal}
\end{align*}
\]

11. **Prep** $\rightarrow$ **terminal**

Assume that the semantic deep structure triple set to be used in the generation is:

0 (man) – R (ride) – 0 (bicycle)  
R (ride) – R (in) – 0 (park)  
0 (man) – R (is) – 0 (tall)

The overlap of various objects and relations in more than one triple is known to the generator by various link markings. The time associated with each triple is also part of the data. A starting symbol $S$ is selected. A prior selectivity mechanism has placed the triple representing the main predicating of the sentence at the top of the triple list. The generative component inspects all $S$ rules whose right hand network description is of the same canonical form as that of the first semantic triple. Here the condition is not satisfied by the only $S$ rule, 1. The triple is then broken into two overlapping parts, 0 (man) – R (ride) and R (ride) – 0 (bicycle). The $S$ rules are then inspected for matches with the fractioned canonical forms. The first matches rule 1. At this point lexical stems are selected from the lexical expression lists associated with the objects and relations in the matched triple fraction. A selected lexical item is tentatively assigned to the node indicated by the link in the syntactic // semantic rule. Grammatical information associated with the lexical item in the dictionary indicates whether or not it can serve as the head of a construction dominated by the node under which it was selected. In this case:
S
NP VP
man ride
sg. pres.

Lexical Dictionary

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
<th>PP</th>
<th>ADJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ride</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A bit vector in the dictionary indicates the applicability of a particular node. Note that both man and ride could serve as nouns or verbs. The grammar also marks the forms when appropriate for application of low level transformations at a later stage. If man were selected as a stem to fill a slot defined by an adjective node, ADJ, it would at this time be marked for later application of a transformation that would add -ly to it. If the lexical dictionary should prevent the selection of a form, an alternate from the lexical expression list is tried. If none on the list are acceptable, another surface // semantic rule is selected to express the semantic triple. Number for objects is indicated directly in the lexical expression list associated with the particular object (some objects may be inherently plural, as in the case of objects that represent classes). As soon as the lexical items are selected and accepted (the stage in the preceding diagram), a test for applicability of a high level transformation is made. This transformation uses as its index information that never becomes more complex than the subtree indicated in the above diagram — "a nuclear family tree" — a parent node and its immediate descendents. Often, as in this case, the lexical items are not relevant to the transformation, that here marks the VP with the same number as the NP.

S

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>ride</td>
</tr>
<tr>
<td>sg.</td>
<td>pres. sg.</td>
</tr>
</tbody>
</table>

Low level transformations that operate only on terminals and their immediate parent nodes will actually convert the stems to the appropriate words at the end of the generation process. The transformation markings supplied by the high level transformations are carried with the lexical items and may serve as part of the data for defining the applicability of other high level transformations. This breaking up of the transformational component into two types of limited environment primitive operations permits extremely rapid transformational generation and parsing algorithms. The complex labor of searching for applicable environments common to most other automated transformational systems is avoided.

Tense information is obtained from the time marking of the triple. The simulation system maintains a clock, and the relative time order of the triples in the deep structure generation list can be computed, so that the proper items may be marked for application of transformations handling tense.

Continuing the generation process, the system saves the remainder of the first triple and skips to the second because of a special link between their relations indicating simultaneity. No VP rule matches the second triple, and it is split into the fractions R (ride) – R (in) and R (in) – R (park). The first fraction matches rule 6. After lexical item in is selected, the tree appears as:

\[ S \]

\[ NP \]

\[ man \]

\[ VP \]

\[ ride \]

\[ sg. \]

\[ PP \]

\[ prep \]

\[ in \]

\[ NP \]

\[ park \]

\[ sg. \]

The second triple fraction matches rule 10, yielding after lexical selection:

\[ PP \]

\[ in \]

\[ NP \]

\[ park \]

\[ sg. \]

At this point, the second fraction of the first triple is matched against rule 7, and, after lexical selection, the entire tree appears as:

\[ S \]

\[ NP \]

\[ man \]

\[ VPP \]

\[ ride \]

\[ pres. sg. \]

\[ PP \]

\[ prep \]

\[ in \]

\[ NP \]

\[ bike \]

\[ in \]

\[ NP \]

\[ park \]

\[ sg. \]

No rule matches the remaining triple 0 (man) – R (is) – 0 (tall). Rule 2 matches the first fraction, but the lexical list for the relation R (is) contains no item acceptable as a PP node descendant. Accordingly, rule 3 is selected. At this point a high level transformation marks the Det for conversion to an appropriate form at the final stage. (If the lexical item had been a proper noun, the Det node would have been marked for deletion.)
At this point rule 4 applies to the entire, unfractionalized, remaining triple, yielding the subtree:

```
  NP
  
  Det
  
  man
  sg.
  
  NPP
  
  adj
  tall
  
  man
  sg.
  
  N
  man
  sg.
```

At this point rule 3 is applied to the NP nodes dominating *bicycle* and *park*. The resultant tree is:

```
  S
  
  VP
  
  NPP
  
  Det
  
  N
  man
  sg.
  
  V
  
  ride
  pres. sg.
  
  NPP
  
  bicycle
  sg.
  
  Prep
  
  in
  
  Det
  
  NPP
  
  park
  sg.
```

The final, low level transformations are applied, yielding the sentence:

"The tall man rides the bicycle in the park"

Note that the semantic triple set might have generated more than one sentence to express the content — either by deliberate stylistic design, or because the rules might not have permitted a grammatically correct construction incorporating the entire semantic structure.

In addition to the features described in the preceding quoted excerpt, we note that the current system makes use of production rules that refer to subclasses of relations. While such subclassification is not logically necessary for the mapping of semantic triples into surface structure, it does increase the speed of generation through the elimination of wasted effort in matching semantic triples with inappropriate rules. In the novel writer data base, for example, there are categorizations of relations into prepositional and non-prepositional types (among others) and a coding logic that permits a retreat to a more general categorization upon failure to find a match in the grammar for a particular subcategory.

There are also relations having a numeric logical typing. Such a relation may be used to select a lexical expression item as a function of its current numeric value. For example a numeric relation signifying "affection" may vary on a scale of plus or minus 3, where plus 3 might be linked to the lexical item "adore" and minus 3 to the item "loathe". In between values link to less extreme terms. The value of such a relation can change dynamically in a simulation as a function of events — accordingly, the appropriate lexical expression of the changing relation follows automatically.

Other features include the listing on generation or change stack of deleted triples and the possibility of marking the lexical expression list pointers with plural transformation markers. This last feature is for semantic nodes whose logical status is always plural, such as nodes that represent class and whose lexical expression lists only contain pointers to terms descriptive of the entire class. (The dictionary only contains singular stems — hence the pointers to the dictionary connected to such nodes must receive prior plural marking.)

4.0. Highlights of the Simulation Language

A detailed description of an early version of the simulation language is contained in Klein, Oakley, Suurballe & Ziesemer, 1971. The basic function of the simulation component is to modify the semantic deep structure network as a function of stochastic behavioral rules that are evaluated in reference to an internal timekeeping mechanism.

A rule consists of two parts, a series of actions and a series of conditions for the implementation of those actions. The conditions are in the form of logical queries about the current state of the modelled universe as represented in the semantic network. Satisfaction or non-satisfaction of the various conditions contribute, either negatively or positively, to a cumulative probability of implementing the action list. A random number source is consulted after the conditions have been evaluated. If the preferred random number is less than or equal to the computed cumulative probability, the action list is implemented. The process can be made deterministic or random with any desired degree of control through manipulation of the probability parameters. Deterministic control is obtained by assigning very high values, such as plus or minus 10, to certain conditions because the range of the random number source is 0 to 1 (a value of 1 or greater indicates certainty and a value 0 or less is absolute rejection).

An internal clock mechanism determines the time of evaluation of groups of rules. Each group has a frequency of evaluation associated with it, and this frequency may be altered by action of some other rule. It may be increased or decreased or, in fact, temporarily or permanently turned off or disabled. A disabled rule may be reactivated.

There is also a directed sequence of evaluation through groups of rules in addition to the frequency factor. This sequence may be altered dynamically as a function of the actions of various rules.

The language also permits the use of classes of nodes in its actions and tests, and can also allow variables over those classes, as well as dynamic modifica-
tion of class membership. There are both subscripted and unsubscripted classes and the subscripted class notation permits a class intersection logic in rules with class variables. For example, a subscripted class \texttt{FRIENDS(X)}, where \(X\) is a node name or another class name, can function as part of a logical construct in rule condition evaluation expression or action lists.

We present next a grammar of the rules in BNF phrase structure notation, a description of the action types, and a series of examples and notes. The material should help the reader follow the murder mystery simulation program in the appendix, Section 7.6.

Grammar of the rules

\begin{align*}
\langle \text{single-valued field} \rangle &::= \langle \text{name} \rangle | \langle \text{loop-variable name} \rangle \\
\langle \text{multiple-valued field} \rangle &::= \langle \text{subrule-variable name} \rangle | \langle \text{general class reference} \rangle \\
&\quad | \langle \text{specific class reference} \rangle \langle \text{unsuperscripted-class name} \rangle \\
&\quad | \langle \text{subscripted-class name} \rangle \langle \text{unsuperscripted-class name} \rangle \\
&\quad | \langle \text{general class reference} \rangle \langle \text{unsuperscripted-class name} \rangle \\
&\quad | \langle \text{subscripted-class name} \rangle \langle \text{unsuperscripted-class name} \rangle \\
\langle \text{general node field} \rangle &::= \langle \text{single-valued field} \rangle | \langle \text{multiple-valued field} \rangle \\
\langle \text{unary op} \rangle &::= \langle \text{NOT} \rangle | \langle \text{FLOAT} \rangle | \langle \text{ABS} \rangle | \langle \text{ENTER} \rangle | \langle -1 \rangle \\
\langle \text{binary op} \rangle &::= \langle * \rangle | \langle * \rangle | \langle / \rangle | \langle \text{MOD} \rangle | \langle + \rangle | \langle - \rangle | \langle \text{EQ} \rangle | \langle \text{NE} \rangle | \langle \text{LT} \rangle | \langle \text{LE} \rangle | \langle \text{GT} \rangle | \langle \text{GE} \rangle | \langle \text{AND} \rangle \\
\langle \text{LENGTH function} \rangle &::= \langle \text{LENGTH} \rangle \langle \text{multiple-valued field} \rangle \\
\langle \text{CLOCK function} \rangle &::= \langle \text{CLOCK} \rangle \\
\langle \text{relation DUR function} \rangle &::= \langle \text{DUR} \rangle \langle \text{relation name} \rangle \\
\langle \text{subrule DUR function} \rangle &::= \langle \text{DUR} \rangle \langle \text{general node field} \rangle \langle \text{relation name} \rangle \\
&\quad | \langle \text{general node field} \rangle \langle \text{general node field} \rangle \\
\langle \text{constant} \rangle &::= \langle \text{number} \rangle | \langle \text{duration} \rangle \\
\langle \text{relation field operand} \rangle &::= \langle \text{relation name} \rangle | \langle \text{LENGTH function} \rangle \\
&\quad | \langle \text{CLOCK function} \rangle | \langle \text{relation DUR function} \rangle \\
&\quad | \langle \text{constant} \rangle \\
\langle \text{relation field subfactor} \rangle &::= \langle \text{relation field operand} \rangle \\
&\quad | \langle \text{relation field expression} \rangle \\
\langle \text{relation field factor} \rangle &::= \langle \text{relation field subfactor} \rangle \\
&\quad | \langle \text{unary op} \rangle \langle \text{relation field factor} \rangle \\
\langle \text{relation field expression} \rangle &::= \langle \text{relation field factor} \rangle \\
&\quad | \langle \text{relation field expression} \rangle \langle \text{binary op} \rangle \langle \text{relation field expression} \rangle \\
\langle \text{subrule-variable definition} \rangle &::= \langle \text{subrule-variable name} \rangle \langle \text{multiple-valued field} \rangle \\
\langle \text{sentence node field} \rangle &::= \langle \text{general node field} \rangle | \langle \text{subrule-variable definition} \rangle \\
\langle \text{sentence} \rangle &::= \langle \langle \text{sentence node field} \rangle \langle \text{relation field expression} \rangle \rangle \langle \text{sentence node field} \rangle \rangle
\end{align*}

Description of actions

I. ACTIONS affecting the network

I–1. Set triples in the network

where \texttt{triple: OBJECT(0) RELATIONSHIP(R) OBJECT(0)}

Forms: A. 0 R 0
B. 0 R = X 0
C. 0 R
D. 0 R = X

FORM OF TRIPLE DEPENDS ON RELATIONSHIP TYPE:
A. is transitive or intransitive relation, B. is numeric or quantitative intrinsitive, C. is attribute relation, D. is quantitative attribute relation or numeric attribute relation

I–2. To delete triples in the network

Form: 0 'NOT' R (0)

I–3. To modify numeric relationships in the network

Form: 0 R \pm X (0)

I–4. To set secondary triples in the network

*INSERT (TRIPLE) (SECONDARY TRIPLE) ...

Secondary triples are modifiers of primary triples and are transparent to the network, being accessible only through the primary triple which it modifies. The form of a secondary triple is arbitrary with the restriction that the second argument is a relationship and the number of arguments \(\leq 3\).

I–5. To delete secondary triples from the network

*DELETE (TRIPLE) (SECONDARY TRIPLE)

NOTE: replace all references to \(\langle \text{NODE} \rangle\) by \(\langle \text{GENERAL NODE FIELD} \rangle\)

II. ACTIONS affecting classes

II–1. To add nodes to a class
*ADD (NODE)'TO'(CLASS): adds all members of
   (GENERAL
   NODE FIELD) to (CLASS)

*MOVE (NODE)'TO'(CLASS): the contents of (CLASS)
   is replaced by (GENERAL
   NODE FIELD)

II-2. To remove nodes from a class
   *REMOVE (NODE)'FROM'(CLASS)

II-3. To remove all entries from a class
   *ERASE (class)

III. ACTIONS affecting lexical items

III-1. To add lexical triples at run-time where the lexical triples are arbi-
   trary combinations of 0's and R's ≤ 3 entries.
   *LEXTRP (arbitrary triple) ... 'TO'(NODE) | (RELATION)

III-2. To move lexical triples from one node or relation to another at run
   time
   *LEXADD (NODE) | (RELATION) ... 'TO'(NODE) | (RELATION)

IV. ACTIONS affecting predicate nodes

IV-1. To insert pointers to network triples to the predicate list of a node.
   *DISCADD (triple)...'TO'(NODE)
   this action will also create triples which do not already exist in the
   network

IV-2. To clear the list of pointers to network triples of a node
   *DISCLEAR (NODE)

V. Actions to control the scheduling of groups of rules

V-1. To activate a group
   *ENABLE (GROUP NAME) IN (DURATION)

V-2. To de-activate a group
   *DISABLE (GROUP NAME)

VI. Miscellaneous Actions

VI-1. To print a list of all triples with a specified node as the subject
   *DUMP (NODE)

VI-2. To control the printing of trace messages in the
   A. *TEST ABCDE = 1000
      ABCD and E are optimal trace types, the number to the right
      of = is a maximum line count for the number of traces to be
      printed.
   B. *TSTOP ABC
      Turns off the traces specified.

C. *TSTART AB
   Turns specified traces on or back on.

VI-3. To print a message
   *PRINT (PRINT ARGUMENT)

VI-4. To terminate simulation
   *END

Examples and notes:

Assume in the following examples that the names below have these associa-

tions:

Node names:  JOHN MARY GEORGE SUE BEDROOM
Relation names:  (A):  HAPPY SAD
                   (I):  LIKES LOVES IN HATES DISLIKES
                   (NI):  AFFECTION
Class names: unsubscribed: PEOPLE ROOMS
              subscribed:  FRIENDS ()
                          ENEMIES ()
                          ADJACENT ()

Loop-variable names:  PERSON ROOM X Y
Subrule-variable names:  P Q R

General notes:

(a) Input cards are read between columns 1 and 72; 73-80 are ignored.
(b) Free format. Blanks can be used freely except in the following cases. Blanks
    must not appear (1) within numbers, durations, or reserved words; (2)
    anywhere in an option field; (3) between trace characters.
(c) Names must start with a letter, followed by letters or digits to any length.
    However, only the first 8 characters are saved. Thus, LOOPNAME1 and
    LOOPNAME2 would be taken as the same variable by the system.
(d) Relations can be of the following types:
    A:  attribute (normal)
    I:  normal intransitive
    T:  transitive
    NA: numeric attribute (with synonym list)
    NI: numeric intransitive (with synonym list)
    QA: quantitative attribute (no synonym list)
    QI: quantitative intransitive (no synonym list)

(1) (multiple-valued field):  P
                                FRIENDS (GEORGE)
                                ADJACENT (ROOMS)
                                PICK (PEOPLE)
                                ENEMIES (PICK(FRIENDS(Q)))
                                PEOPLE
The PICK function returns a single node, chosen randomly, from its argument. Multivalued subscripts imply concatenation of the specified subscripted classes.

(2) (specific class reference): PEOPLE
FRIENDS (PERSON)
ADJACENT (BEDROOM)

(3) (general node field): JOHN MARY PERSON
P PEOPLE ENEMIES (PICK(Q))
PICK (PEOPLE)
ADJACENT (ROOMS)

(4) (unary op): The FLOAT operator operates on arguments of type logical, giving 1.0 for TRUE and 0.0 for FALSE. The ENTIER operator truncates the fractional part of a number (e.g., ENTIER(14.23) = 14.0).

(5) (binary op): The symbols =, !=, <, <=, >, >= can be used as synonyms for the relational operators EQ, NE, LT, LE, GT, and GE.

(6) (LENGTH function): LENGTH (PEOPLE)
LENGTH (ADJACENT (ROOMS))

Returns a number equal to the number of nodes in its argument.

(7) (CLOCK function): Returns a number which corresponds to the time of day, i.e., from OHOM to 23H59M.

(8) (relation DUR function): DUR (LIKES)
DUR (IN)
DUR (HAPPY)

This function occurs inside a sentence. (S DUR (R) 0) returns a number equal to the length of time this triple has been in the network. The relation name must be of a non-numeric relation. If the triple does not exist, a run-time error is printed and 0.0 is returned.

(9) (subrule DUR function): DUR (JOHN LIKES MARY)
DUR (PERSON IN R)

Returns a number equal to the length of time a triple has been in the network. The relation name must be non-numeric. While multiple-valued fields are allowed in the syntax, they must contain only a single value at execution time of a DUR function, or else a run-time error will result. Note that no subrule-variable updating ever occurs in a subrule DUR function. If the specified triple is not in the network, an error is printed out, and 0.0 is returned.

(10) (relation field expression): LIKES
LIKES AND NOT (HATES OR DISLIKES)
DUR (LIKES) GT 1H OR DUR (LOVES)
GT 30 M

Relation field expression can be either of type logical or type numeric. A relation name that is numeric or quantitative (ie, NA, NI, QA, or QI) is taken as a numeric operand. Other types (A, I, or T) all are assumed to be logical operands (except within a DUR function). The type of the relation expression determines what type of result the enclosing sentence will return, either a logical value or a numeric value. The operators have specified precedences not explicitly implied in the grammar, and checks are made for correct operand types.

(11) (sentence): (PERSON LIKES OR LOVES P. PEOPLE)
(JOHN AFFECTION MARY)
(X DUR (LIKES) Y)
(GEORGE DUR (LOVES) LT 1W SUE)
(FRIENDS (X) AFFECTION LT 0 Y)
(X HAPPY OR NOT SAD)
(MARY HAPPY AND LIKES JOHN)

All these sentences return a logical result except the second and third ones. If the relation expression in a sentence yields a numeric value, the subject and object fields of the sentence must be single-valued, or else an error will result.

(12) (option field): An optional field which specifies the options to be in effect. Currently used options are:
S Synchronous group flag. Used in the option field of a $GROUP statement to flag a group as synchronous. Eg, $GROUP'S NEWS: 1H/ON, defines a group which will be executed at hour intervals, on the hour.
O Optimization flag. (Sentences with side effects are not necessarily executed in the subrules, depending on the results of previous logical results).
C Current cycle flag. Allows sentences to test for triples which have been set true during the current time cycle. (Otherwise these are not available till a later time cycle, ie, they act as if they weren't there during the same time cycle).

An option field specified on a $GROUP, $LOOP, $RULE, or $SWITCH statement is in effect for all subrules within its scope, unless explicitly overridden by an option field at a lower level.

(13) (subrule): .2,0: (PERSON LIKES OR LOVES P. PEOPLE)
AND (P IN ROOM);
-10,0, C: (X NOT IN HOUSE) OR (Y NOT IN HOUSE);
(14) (action list):
A list of one or more actions, separated by commas. Actions can either add or delete triples from the network, or perform a control action such as manipulating classes, enabling or disabling groups, or specifying trace or print parameters.

(15) (branch field):
RULE1
$NEXT PERSON
$NEXT X
$ENDGROUP

A statement label gives the statement to branch to. A rule can branch anywhere within a group, including out of a loop into an outside loop, but not within a non-enclosing loop. The $NEXT format says to get the next value for a loop variable (equivalent to flowing into an $ENDLOOP statement for that loop). A branch to $ENDGROUP terminates the execution of the group, though it does not disable the group (a *DISABLE action is the only thing which can do this).

(16) ($RULE statement):
Basic unit of the language. The cumulative total of the subrule probabilities is tested against a random number which is generated. If the random number (between 0 and 1) is less than or equal to the cumulative total, the rule evaluates TRUE, and the action list of the rule is executed. If not, then it evaluates to FALSE and no actions in the rule's action list are executed. If a branch part is specified, the TRUE or FALSE result also tells where to branch to. E.g., $RULE, C ABC: T($NEXT X) X LIKES Y, *ADD Y TO FRIENDS(6); 
-2.0: (X HATES OR DISLIKES Y); 
.4.0: (P FRIENDS(X) LIKES Y) AND (X LOVES P); 

(17) ($SWITCH statement):
This is exactly the same as a $RULE statement except that an action list cannot be specified in the main part of the statement (i.e., subrule action lists are still allowed). This statement is used only for branching purposes.

(18) ($LOOP statement):
The specified loop variable will take on all values in the associated multi-

5.0. Novel Writer Features and Futures

The data base for the murder mystery simulation is rather simple and skeletal. A very small grammar was used with only a few transformations. The lexical expression lists contain only a limited selection of variants for the semantic nodes and relations. Some errors in the grammar codes of some dictionary items remain.
Our goal was to test the entire system. It is capable of operating with a vastly more sophisticated data structure. Also, not all features of the simulation language were exploited in the murder mystery program. The predicate node device was not used. Text involving productions such as "George knows that John loves Mary", were derived from exploitation of the same secondary triple device that handled expressions of the type, "John broke the window with a hammer." The reason: while the simulation language can dynamically add semantic triple list pointers to nodes and relations, the code for adding the indicated triples to the change stack is not fully implemented. The final implementation of this code will permit easy generation of direct discourse, e.g. constructions such as "John said, '(sentence1, sentence2 ... sentence_n)'"

5.1. Style Control

While some effort was made to control a few facets of style in the current simulation, most possibilities remain to be exploited. We have found that the simulation language itself can be exploited as a style control device. Various constructs in the rules indicate which triples may be combined into a single sentence according to a sequencing logic. Also, the repetition of the same action by several characters at the same time is usually expressed by a pronoun such as "They ..." or "Everyone ..." even though each individual action is separately tabulated in the semantic network. To achieve this a special "They" node was created in combination with a "They" class. Several individuals performing the same action in the same time period are assigned temporarily to the "They" class, and output makes use of a triple signifying the action with the "They" node functioning as the subject. Special commands such as UNLIST and LST alternately block and unblock the generation of uninteresting or repetitious semantic triples. This blocking is occasionally introduced as a random device to vary the output.

A crude and not always successful device is used to control the use of definite and indefinite articles. For the first occurrence of some nodes on the change stack "a" is selected - in successive productions "the" is used. (This tabulation holds for all succeeding time frames.) The device collapses where the simulation program data structure has apportioned only a single class type node for several objects (out of laziness or for economy).

Weighted probabilistic selection of syntactic rules is a device that, although not used in the current system, was actually successfully tested in an automatic essay paraphrasing and style control system described in Klein, 1965a & b.

Narration from the point of view of particular characters is another possibility, and is perhaps most interestingly implemented with the addition of private semantic universes (see section 5.2).

Addition of a complex network searching component will permit the system to add rich contextual detail to events. For example, where now a change stack may contain just some bare facts about recent changes, a network searching device could seek paths between nodes in apparently unrelated triples, and, if paths exist, add them to the change stack as linking background information.

It should also be possible to have different characters produce discourse in varying styles and dialects as a function of sociolinguistic context. The techniques are implicit in the following discussion of private universes.

5.2. Private Semantic Universes for Individual Characters

The ability to provide individual characters in a simulation with private semantic networks, personalized grammars, and even personalized behavioral simulation rules can be achieved with only mildly clever systems programming techniques. The operating system on the Univac 1108, and operating systems of perhaps all 3rd and 4th generation computers have system commands to facilitate a restart capability - that is, the ability to store on disc the current state of a program at specified intervals during execution so that in the event of system failure, the program may be restarted at the point of the last execution of a "store on disc command", without the necessity of starting the program from the beginning.

To implement private universes for individual characters, it is only necessary to add an executive program that will treat each private universe as the total universe when it is resident in core storage, and to save it on disc with a unique name when it is ready to process another character's private universe. The existence of core-resident buffers for communication between private universes is assumed.

5.3. Simulation of Simulations: Look-Ahead, Planning, Time Travel and Dreams

Implementation of the private universe capability permits some fascinating possibilities: An individual character could be made to resort to his own look-ahead simulation of events in order to evaluate decision making criteria about the implication of current actions on future events. This would require a private simulation using the data and rules of a private universe. The outcome or outcomes could serve as data to compute probabilities of courses of action for the private individual's actual, simulated real world behavior. Of course introspective, look-ahead simulation need not give accurate results, only hypothetical predictions based on the private rules of a private universe. Naturally, such a universe might contain models of other characters and their private universe. The device also lends itself to the modelling of dream behavior.

For those readers with an interest in science fiction fantasy, we note that this device can be used to model time travel stories, with all conceivable paradoxes. Essentially, it is necessary that the rules permit a private character to treat his introspective look-ahead (or look-back) as serious reality rather than speculation. In the case of travel into the past, all the other characters must take the look-back seriously also.
5.4. Semantic Parsing

The private universe concept makes it interesting to allow communication between modelled characters directly via conversational interaction. Of course sophisticated semantic parsing techniques are required. A great deal of work in this area has been attempted by numerous researchers. Although we have not implemented such programs in this system, preliminary study suggests that it will permit semantic parsing logic many times more powerful than any in programs currently in existence. The reason: we own the universe of discourse, a universe where all the subtleties of behavior, motivation and context over complex time intervals are all available as data for resolution of the ambiguity that always plagues development of sophisticated semantic parsers.

5.5. Linguistic and Behavioral Learning: Self-Modifying Behavior and Natural Language Meta-Compiling

The use of this system for modelling speech communities, language learning and language transmission in conjunction with sociolinguistic models has been explored in detail in Klein, 1965, 1966, 1972 and Klein et al. 1969.

The transmission and learning of complex, non-verbal behavioral patterns is also possible using the same mechanisms of the system. Simulation rules may also have a representation in the semantic deep structure network of private individuals. Also, the semantic deep structure may be used to generate sentences and texts (rules and rule groups) in the simulation language itself. The system already has the ability to compile dynamically and add to the simulation new rules that might be generated during the flow of a simulation. It thus becomes possible for characters to modify their own behavior rules in response to private introspection and look-ahead, or in response to verbal and non-verbal behavior of others.

The simulation rules governing rule generating behavior may themselves be modified and generated by the same mechanisms, providing the system with a natural language, meta-compiler capability.

6.0. Significance for Linguistics, Sociolinguistics and the Behavioral Sciences in General

We dare to say that Linguistic Theory has no future that is not linked to a computer based experimental methodology. Contemporary linguistic theoretical science has many brilliant theorists in the position analogous to that of a great mathematician attempting to formulate the methodology of long division using roman numerals.

The system described here, with its potential development, provides a means of expressing and testing a vast range of theoretical linguistic models in conjunction with a vast range of sociological and psychological behavioral models, all within the framework of a common, efficient, dynamic time-oriented notation. The implication is that, for the first time, it will be possible to test heretofore untestable theories of language and language related behavior in psychological, sociological and historical contexts.

7.0. Appendix

The semantic deep structure model, as reflected in the choice of nodes, relations and mappings has been more or less arbitrary and experimental, even deliberately inconsistent. The function of the system is independent of the choice of semantic units. One may substitute any scheme according to the dictates of any theory. However, preliminary results suggest that any number of semantic deep structure components will all work nicely, and that the usual arguments for economy or elegance that are to be found in linguistic literature are not necessarily valid in this system. We sense the possibility of proof that such arguments are really functions of the particular notational devices used. A basic principle in computational work is that there is an economy trade between static storage space versus computation time. The non-computational models of linguistic theorists ignore this fact in their proposals and arguments for models of human language behavior.


Logically, the system need not be limited to semantic 3-tuples and binary phrase structure rules, although such a convention has been used in this version.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>object</td>
<td>sub 1 = that</td>
</tr>
<tr>
<td>R</td>
<td>any relation</td>
<td></td>
</tr>
<tr>
<td>RA</td>
<td>attribute (adj)</td>
<td></td>
</tr>
<tr>
<td>RV</td>
<td>verb</td>
<td></td>
</tr>
<tr>
<td>RP</td>
<td>prep</td>
<td></td>
</tr>
<tr>
<td>RS</td>
<td>possessive</td>
<td></td>
</tr>
<tr>
<td>RADV</td>
<td>adverb</td>
<td></td>
</tr>
</tbody>
</table>

Pmap positionally defines mappings between PTYPE triple fragments and the phrase structure rule portions. E.g. in rule 1, the 0 is linked to the NP and the RV is linked to the VP; in rule 4, the first 0 is linked to NPP, the RS is linked to nothing and the second 0 is linked to PNP. Psub positionally lists relation type subscripts in parallel form. Ptrans indicates high level transformation mapping information associated with each rule:

1. = carry down bit vector (null trans.)
2. = OR (logical) bit vectors of new nodes
3. = set infinitive bits for both words
4. = set participle bit for second word
5. = set objective case bit for second word
7.2. Transformations
As indicated earlier, the system obtains its ability to model a variety of linguistic models, and at the same time a great speed of execution, by decomposing transformational operations into primitive components at several stages. Indications for applications of the transformational fragments are marked and tabulated throughout the generation process. Some of the transformation types themselves give directions for computing and assigning the transformational markings to the growing generation tree (as in section 7.1.).

Ultimately, every terminal element is associated with a bit vector indicating applicable low level transformations as assigned during the various stages of generation. The method avoids complex tree search after phrase structure generation, and in comparison with other automated transformational generation systems obtains thereby what may be a 100 to 1 speed advantage.

High Level Transformation Codes (non pronoun)
1. noun sing.
2. noun plural
3. adjectival form
4. prepositional form
5. adverbial form
6. participial form
7. verb (present sing.)
8. verb (present plural)
9. verb (past sing.)
10. verb (past plural)

High Level Transformation (pronoun)
1. subjective case
2. objective case

Low Level Transformation Codes
1. NULL
2. add “will”
3. add “s”
4. add “ing”
5. add “d”
6. add “ly”
7. add “y”
8. delete 1 character and add “ies”
9. add “ed”
10. delete 1 character, add “ing”
11. delete 2 character, add “en”
12. add “es”
13. add “er”
14. add “ings”
15. add “ers”
There are other kinds of high level discourse type transformations not listed here. Of special interest is the one in the form of a special triple of the form MX QQ (n): combine the next (n) head triples with the one preceding. It can be found in the simulation commands and on change stacks.

7.3. Dictionary

Lines 3–8 are patterns for setting grammar symbol bits in the dictionary.

The word TYPE delineates classes of words.

The line following TYPE sets bits in the dictionary bit vector (article/no article, pronoun, etc.). For example: line 179 – bit 2 is set for all words in that class for ‘no article’; in line 266, bits 2 and 9 are set for all words in that class for ‘no article’, ‘pronoun’.

The lines with pattern types (N, V, PREP, ADJ, ADV, PART) indicate which patterns of grammar bits set. For example: line 12, line 14 – for word “BE” all bits of pattern PART (line 8) and all bits of pattern V (line 4) will be set. Thus, “BE” is an allowable choice for V, VP, MOD, AP, VP2, or PART when matching in grammar rules.

The lines following pattern types indicate transformations to be associated with all words in the class. For example: in line 25, noun sing. transformation is TRANS #1 on word 0; noun pl. transformation is TRANS #3 on word 0. Stem alternates are listed with their associated transformations. Word 0 = main entry Word 1 = 1st stem, Word 2 = 2nd stem, etc. For example: in lines 15–16, V present sing. is TRANS #1 on stem 1 (null trans on “is”); V past sing. is TRANS #1 on stem 2 (null trans-on “are”).
7.4. Nodes, Relations and Classes

The input data for the nodes contains a listing of node names followed by a lexical expression list. Numbers separated by spaces indicate the following:

0 = singular
1 = Plural
2 = Singular, but definite article even on 1st occurrence
3 = plural, and always associated with a definite article

Note that this information is eventually passed on to both high level and low level transformation components; other devices may also determine number at later stages.

Three pieces of information are associated with the relation input in addition to the specification of the lexical expression list. The letter codes indicate logical type:

A = attribute (normal)
T = transitive
N = numerical intransitive: with lexical expression list
QA = quantitative attribute (no lexical expression list)
I = normal intransitive
NA = numerical attribute (with lexical expression list)

‘Transitive’ and ‘intransitive’ here refer to logical transitivity as opposed to syntactic transitivity. E.g. “if A R B and BRC, then ARC” implies that R is transitive. The first number following the letter code represents the relation type:

3 = general class
2 = attribute class
4 = prepositional class
6 = adverbial type
5 = possessive

These are not grammar codes, but rather devices for speeding up selecting of
rules for generation. The designations as preposition, adverb, etc. are arbitrary; they actually represent a higher order semantic classification. The third number represents an additional subclass marking for partition of the class specified by the 1st digit.

The class listing contains the class names followed by a listing of elements; the listing may be empty or include both nodes and other class names.

GXGT CS*0BRL.ABSSFC
1 \$LIMITS START = 19W3D10H* END = 20W2D14H;
2 %
SILLRDM0 = 'BILLIARD ROOM';
3 % **************************** NODES ****************************
4 %
5 % $NODES;
6 THAT 0 = 'THAT';
7 MX 0 = 'THAT';
8 LST 0 = 'THAT';
9 ULST 0 = 'THAT';
10 ACCUSATION 2 = 'ACCUSATION';
11 ACTIVITIES 1 = 'ACTIVITY';
12 ADULTERY 0 = 'ADULTERY';
13 ASHES 1 = 'ASH';
14 BATHROOM 2 = 'BATHROOM';
15 BED 0 = 'BED';
16 BEDROOM 2 = 'BEDROOM';
17 BILLIARDS 0 = 'BILLIARDS';
18 BILLRDM 0 = 'BILLIARD ROOM';
19 BLOOD 2 = 'BLOOD' 'GORE';
20 BOOK 0 = 'BOOK' 'PAPERBACK';
21 BOOKS 1 = 'BOOK';
22 BOTTLE 2 = 'BOTTLE';
23 BREAKFAST 0 = 'BREAKFAST';
24 BRIDGE 0 = 'BRIDGE';
25 BUSINESS 0 = 'BUSINESS';
26 BUTLER 0 = 'BUTLER' 'CLIVE';
27 BUTTON 0 = 'BUTTON';
28 CANDLHOLD 0 = 'CANDLE HOLDER';
29 CARDGAME 2 = 'CARD GAME';
30 CATHY 0 = 'CATHY' 'CATHERINE' 'LADY CATHERINE';
31 CHESS 0 = 'CHESS';
32 CIGARS 1 = 'CIGAR' 'STOGY' 'Havana';
33 CLUES 1 = 'CLUE' 'HINT';
34 CLUE1 0 = 'CLUE';
35 CLUE2 0 = 'CLUE';
36 COFFEE 0 = 'COFFEE';
37 COMPANION 0 = 'COMPANION';
38 CONVERSATION 0 = 'CONVERSATION';
39 CONVERTNS 1 = 'CONVERSATION';
40 COOK 0 = 'COOK' 'MAGGIE';
41 CORPSE 2 = 'CORPSE' 'BODY';
42 COUCH 0 = 'COUCH' 'DIVAN' 'DAVENPORT';
43 CRIME 2 = 'CRIME';
44 CROQGAME 2 = 'CROQUET GAME';
45 CROQUET 0 = 'CROQUET';
46 DAWN 2 = 'DAWN' 'SUNRISE';
47 DAY 2 = 'DAY';
48 DINNER 0 = 'LUNCH';
49 DININGRM 0 = 'DINING ROOM';
50 DOOR 2 = 'DOOR';
51 DRAWR 0 = 'DRAWER';
52 DRAWRS 1 = 'DRAWER';
53 DRAWINGRM 2 = 'DRAWING ROOM';
54 DRHUME 0 = 'DR. HUME' 'DR. BARTHOLOMEW HUME' 'HUME';
55 ENCORE 0 = 'ENCORE';
56 EVENING 2 = 'EVENING';
57 EVERYONE 0 = 'EVERYONE';
58 FAIL 0 = 'FAIL';
59 FASHION 2 = 'FASHION';
60 FEAR 0 = 'FEAR';
61 FLOWERS 3 = 'FLOWER';
62 FOOD 2 = 'FOOD';
63 FOOTPRINT 0 = 'FOOTPRINT';
64 FRNENTS 1 = 'FINGERPRINT';
65 FRIDAY 0 = 'FRIDAY';
66 GAME 0 = 'GAME';
67 GARDEN 0 = 'GARDEN';
68 GOODNIGHT 0 = 'GOOD NIGHT';
69 GOODTIME 0 = 'FUN';
70 GREED 0 = 'GREED';
71 GREENHHS 0 = 'GREEN HOUSE';
72 GUN 0 = 'GUN' 'Pistol';
73 HAIR 0 = 'HAIR';
74 HALL 2 = 'HALL' 'CORRIDOR';
75 HANDKERCHIEF 0 = 'HANDKERCHIEF';
76 HANDS 1 = 'HAND';
77 HEAD 2 = 'HEAD';
78 HOUSE 0 = 'HOUSE';
79 INFORMATION 2 = 'INFORMATION';
80 INSPECTOR 0 = 'INSPECTOR' 'DETECTIVE';
81 JAIL 2 = 'JAIL';
82 JAMES 0 = 'JAMES';
83 JAW 2 = 'JAW' 'CHIN';
84 JEALOUSY 0 = 'JEALOUSY';
85 JEWELS 1 = 'JEWEL' 'JEWELRY';
86 JOHNBUX 0 = 'JOHN' 'JOHN BUXLEY';
87 JOKE 0 = 'JOKE' 'FUNNY STORY';
88 KITCHEN 2 = 'KITCHEN';
89 KNIFE 0 = 'KNIFE' 'TAXI DRIVER';
90 LADYBUXLEY 0 = 'LADY BUXLEY';
91 LADYJANE 0 = 'JANE' 'LADY JANE';
92 LIVINGRM 2 = 'LADY BUXLEY'S BEDROOM';
93 LIFEC 0 = 'LIFE';
94 LIBRARY 0 = 'LIBRARY';
95 LORD 0 = 'LORD EDWARD' 'EDWARD';
96 LOVER 0 = 'LOVE';
97 MAID 0 = 'MAID' 'HEATHER'.
MARION 0 = 'MARION';
MEN 3 = 'MAN';
MILK 0 = 'MILK';
MISTAKE 0 = 'MISTAKE' 'ERROR';
MONEY 2 = 'MONEY';
MOTIVE 2 = 'MOTIVE';
MURDER 2 = '*' 'MURDER';
MURDERER 2 = 'MURDER' 'KILL';
MUSIC 0 = 'MUSIC';
NECK 2 = 'NECK';
NEPHEW 0 = 'NEPHEW';
NIGHTGOWN 0 = 'NIGHTGOWN';
NOONE 0 = 'NO ONE' 'N O B D Y';
NOSE 2 = 'NOSE';
NOTE 0 = 'NOTE';
NOVEL 0 = 'NOVEL';
NURSE 0 = 'FLORENCE';
ONCE 0 = 'ONCE';
OTHERS 3 = 'OTHER';
PAPERWT 0 = 'PAPERWEIGHT';
PARLOR 0 = 'PARLOR';
PARTNER 2 = 'PARTNER';
PARTY 0 = 'PARTY';
PIANO 2 = 'PIANO';
PILLOW 0 = 'PILLOW' 'CUSHION';
PLACE 0 = 'PLACE';
PLAN 2 = 'PLAN';
PLAYER 2 = 'PLAY';
POISON 2 = 'POISON';
POLICE 3 = 'POLICEMAN' 'COP';
POLITICS 0 = 'POLITICS';
PORT 0 = 'PORT';
QUESTNS 1 = 'QUESTION';
REVENGE 0 = 'REVENGE';
RONALD 0 = 'RONALD';
ROOM 2 = 'ROOM';
SATURDAY 0 = 'SATURDAY';
SECRET 0 = 'SECRET PASSAGE';
SERVANTS 3 = 'SERVANT';
SHEERY 0 = 'SHEERY';
SHIRT 0 = 'SHIRT';
SHOE 0 = 'SHOE';
SKY 2 = 'KY';
SMOTHERING 0 = 'SHMOTHER';
SOMEONE 0 = 'SOMEONE';
STAIN 0 = 'STAIN';
STAIRS 1 = 'STAIR';
STOMACH 2 = 'STOMACH' 'BELLY';
STRAND 0 = 'STRAND';
STRANDOFAIR 0 = ;
STUDY 2 = 'STUDY';
SUN 2 = 'SUN';
SUNDAY 0 = 'SUNDAY';
SUPPER 0 = 'SUPPER' 'DINNER';
TEA 2 = 'TEA';
TEATIME 0 = 'TEA TIME';
TENNIS 0 = 'TENNIS';
TENNISCOURT 2 = 'TENNIS COURT';
THEY 3 = 'THEY';
THREAD 0 = 'THREAD';
TIME 2 = 'TIME';
TRASH 0 = 'TRASH' 'JUNK';
TRUTH 2 = 'TRUTH';
VASE 0 = 'VASE';
VODKA 0 = 'VODKA';
WALK 0 = 'WALK';
WEATHER 2 = 'WEATHER';
WEEKEND 0 = 'WEEKEND';
WHAT 0 = 'WHAT';
WHISKEY 0 = 'WHISKEY';
WHO 0 = 'WHO';
WINDOW 0 = 'WINDOW';
WOMEN 3 = 'WOMAN';
YARD 2 = 'YARD';
AFFAIR 2 = 'AFFAIR';
AFFECTION 0 = 'AFFECTION';
BAR 2 = 'BAR';
BARMAN 2 = 'BARMAN';
BEATLES 3 = 'BEATLE';
CANTEN 0 = 'PUB';
CARDS 3 = 'CARD';
CHANGE 0 = 'CHANGE';
CLUE 0 = 'CLUB';
COOKIES 1 = 'COOKIE';
CORNER 0 = 'CORNER';
DETAILS 1 = 'DETAIL';
DRINK 0 = 'DRINK';
DRINKS 1 = 'DRINK';
FRIEND 0 = 'FRIEND';
HOTEL 0 = 'HOTEL';
INTERMISSION 0 = 'INTERMISSION';
INVITATION 0 = 'INVITATION';
IT 0 = 'IT';
MONDAY 0 = 'MONDAY';
MOVIE 0 = 'MOVIE';
MORNING 2 = 'MORNING';
PARK 0 = 'PARK';
PASSION 0 = 'PASSION';
ROCKS 3 = 'ROCK';
SODA 0 = 'SODA';
SOMETHING 0 = 'SOMETHING';
SONG 0 = 'SONG';
TELEPHONE 0 = 'TELEPHONE';
THEATRE 0 = 'THEATER';
THURSDAY 0 = 'THURSDAY';
TUESDAY 0 = 'TUESDAY';
WEDNESDAY 0 = 'WEDNESDAY';
% **************** RELATIONS ****************

***

206 %
207 % RELATIONS;
208 %
209 GO: NA (10) 20 = 'THAT';
210 ACCOST 130 = 'ACCOST';
211 ACCUSE 130 = 'ACCUSE';
212 AFFECTION (3) 130 = 'HATE'/-2.5'/DISLIKE'/-0.5'/LIKE'/.5'/LOVE';
213 AGREE 130 = 'AGREE';
214 AGREEWITH 130 = ;
215 ANNOUNCE 130 = 'ANNOUNCE';
216 ARGUE 130 = 'ARGUE';
217 ARGUWITH 130 = ;
218 ARREST 130 = 'ARREST';
219 ARRIVE 130 = 'ARRIVE';
220 ASK 130 = 'ASK';
221 ASKFOR 130 = ;
222 ATTACK 130 = 'ATTACK';
223 AWAKE 130 = 'AWAKE';
224 AWAKEEN 130 = 'AWAKE';
225 BEAT 130 = 'BEAT';
226 BETRAY 130 = 'BETRAY';
227 BLACKMAIL 130 = 'BLACKMAIL';
228 BLEED 130 = 'BLEED';
229 BREAK 130 = 'BREAK';
230 CALL 130 = 'CALL';
231 CALM 130 = 'CALM';
232 CARESS 130 = 'CARESS';
233 CARRY 130 = 'CARRY';
234 CATCH 130 = 'CATCH';
235 CHEAT 130 = 'CHEAT';
236 CHOKE 130 = 'CHOKE';
237 COLLAPSE 130 = 'COLLAPSE';
238 COMMIT 130 = 'COMMIT';
239 COMPLIMENT 130 = 'COMPLIMENT';
240 CONFESS 130 = 'CONFESS';
241 CONGRATULATE 130 = 'CONGRATULATE';
242 CONVINCE 130 = 'CONVINCE';
243 COUNT QA (8) 20 = ;
244 COVER 130 = 'COVER';
245 COVERWITH 120 = ;
246 CRY 130 = 'CRY';
247 CURSE 130 = 'CURSE';
248 DECIDE 130 = 'DECIDE';
249 DECEIVE 130 = 'DECEIVE';
250 DENY 130 = 'DENY';
251 DESPISE 130 = 'DESCISE';
252 DIE 130 = 'DIE';
253 DISCUSS 130 = 'DISCUSS';
254 DRAW 130 = 'DRAW';
255 DRINK 130 = 'DRINK';
256 EAT 130 = 'EAT';
257 ENJOY 130 = 'ENJOY';
258 ENTER 130 = 'ENTER';
259 EXAMINE 130 = 'EXAMINE';
260 FAINT 120 = 'FAINT';
261 FALL 130 = 'FALL';
262 FALLDWN 130 = ;
263 FEEL 130 = 'FEEL';
264 FEELNO 130 = 'FEEL';
265 FEELWELL 130 = ;
266 FIND 130 = 'FIND';
267 FIRE 130 = 'FIRE';
268 FLATTER 130 = 'FLATTER';
269 FLIRT 130 = 'FLIRT';
270 FLIRTWITH 130 = ;
271 FOLLOW 130 = 'FOLLOW';
272 FORGIVE 130 = 'FORGIVE';
273 FUCK 130 = 'SCREW' 'SEDUCE';
274 GET 130 = 'GET';
275 GETDRESS 130 = ;
276 GETUP 130 = ;
277 GIVE 130 = 'GIVE';
278 GO A 31 = 'GO';
279 GOFOR 130 = ;
280 GOSSIP 130 = 'GOSSIP';
281 GOSSIPNO 130 = 'GOSSIP';
282 GOTO 130 = ;
283 GRAB 130 = 'GRAB';
284 GRABFOR 130 = ;
285 GRATE 130 = 'GRATE';
286 GREET 130 = 'GREET';
287 GROOM 130 = 'GROOM';
288 HAVE 130 = 'HAVE';
289 HEADNO 130 = 'HEAD';
290 HEADFOR 130 = ;
291 HEAR 130 = 'HEAR';
292 HIDE 130 = 'HIDE';
293 HIDDEN 130 = 'HIDE';
294 HIT 130 = 'HIT';
295 IGNORE 130 = 'IGNORE';
296 INHERIT 130 = 'INHERIT';
297 INSULT 130 = 'INSULT';
298 IS 130 = 'BE';
299 JOIN 130 = 'JOIN';
300 KEEP 131 = 'KEEP';
301 KICK 130 = 'KICK';
302 KILL 130 = 'KILL';
303 KILLEDBY 130 = ;
304 KISS 130 = 'KISS';
305 KNOW 110 = 'KNOW';
306 LAST 130 = 'LAST';
307 LAUGH 130 = 'LAUGH';
308 LEAVE 130 = 'LEAVE';
309 LOOK 130 = 'LOOK';
310 LOOKFOR 130 = ;
311 LOOKTHROUGH 130 = ;
312 LOOKWHILE 130 = ;
MAKE 130 = "MAKE";
MENTION 130 = "MENTION";
MEET 130 = "MEET";
MOVE A 30 = "MOVE";
OPEN 130 = "OPEN";
OVERHEAR 130 = "OVERHEAR";
OWN 130 = "OWN";
PANIC A 30 = "PANIC";
PAY 130 = "PAY";
PLANNO A 30 = "PLAN";
PLAY 130 = "PLAY";
POINT 130 = "POINT";
POISONS 130 = "POISON";
POS 130 = "BE";
PREPARE 130 = "PREPARE";
PRETEND A 30 = "PRETEND";
PUSH 130 = "PUSH";
QUESTION I 130 = "QUESTION";
QUIT A 30 = "QUIT";
READ 130 = "READ";
RELATEDTO 130 = ;
REMOVE 130 = "REMOVE";
RESUME 130 = "RESUME";
RETURN 130 = "RETURN";
RETURNING 130 = ;
RIP 130 = "RIP";
RIPFROM 130 = ;
RISE A 30 = "RISE";
RUN A 30 = "RUN";
SAY 130 = "SAY";
SAYTO 130 = ;
SCOFF A 30 = "SCOFF";
SCRATCH 130 = "SCRATCH";
SCREAM A 30 = "SCREAM";
SEARCH 130 = "SEARCH";
SEDUCE 130 = "SEDUCE";
SEE 130 = "SEE";
SERVE 130 = "SERVE";
SHOOT 130 = "SHOOT";
SHOOTAT 130 = ;
SINK A 30 = "SINK";
SIT A 30 = "SIT";
SLASH 130 = "SLASH";
SMASH 130 = "SMASH";
SMILE A 30 = "SMILE";
SMILEAT 130 = ;
SMOK 130 = "SMOKE";
SMOTHER 130 = "SMOTHER";
SNEAK A 30 = "SNEAK";
SNORE A 30 = "SNORE";
SOLVE 130 = "SOLVE";
STAB 130 = "STAB";
STAGGER A 30 = "STAGGER";
START 130 = "START";
STARTNO A 31 = "START";
STEAL 130 = "STEAL";
STOP 130 = "STOP";
STOPNO A 31 = "STOP";
STRUGGLE A 10 = "STRUGGLE";
STRUGLWITH 130 = ;
SUGGEST 130 = "SUGGEST";
SURPRISE 130 = "SURPRISE";
Suspect 130 = "SUSPECT";
TAKE 130 = "TAKE";
TALK A 30 = "TALK";
TALKABOUT 140 = ;
TALKWITH 130 = ;
TELL 130 = "TELL";
THINK 130 = "THINK";
THREATEN 130 = "THREATEN";
THROW 130 = "THROW";
THROWAWAY 130 = ;
TOUCH 130 = "TOUCH";
TRIP 130 = "TRIP";
TRY A 30 = "TRY";
UNDRESS A 30 = "UNDRESS";
WAIT A 30 = "WAIT";
WAITFOR 130 = ;
WALKNO A 30 = "WALK";
WALKIN 130 = ;
WANT 130 = "WANT";
WANTNO A 30 = "WANT";
WASH A 30 = "WASH";
WAVE 130 = "WAVE";
WEAR 130 = "WEAR";
WHYKILL Q15 130 = ;
Wisper A 30 = "WHISPER";
WISPERTO 130 = ;
WRITE 130 = "WRITE";
YELL A 30 = "YELL";
YELLAT 130 = ;
YAWN A 30 = "YAWN";
BRING 130 = "BRING";
COME 130 = "COME";
COMEWITH 130 = ;
FORCSTA QA(13) 20 = ;
GOZIP 130 = ;
HAPPENED A 30 = "HAPPEN";
INTRODUCE 130 = "INTRODUCE";
INVITE 130 = "INVITE";
LIKE 130 = "LIKE";
NOTICE 130 = "NOTICE";
NUMBER QA(6) 20 = ;
OFFER 130 = "OFFER";
PHONE 130 = "PHONE";
RECALL 130 = "RECALL";
RECEIVE 130 = "RECEIVE";
REMEMBER 130 = "REMEMBER";
7.5. Network and Simulation Rule Plot Specification

The specification of the network includes the assignment of all initial conditions: numerical attributes, lexical triples, semantic triples, and a listing of relations which are logically mutually exclusive for automatic maintenance of logical consistency. This initialization of starting conditions is part of the first time frame of the simulation. Comments on the significance of groups of rules appear indented between them. The following is a small sample of the total data.
TALKING = GUESTS;
TEMP = ;
WAKE = GUESTS INVITED;
WANTED () = ;
WINNER = ;
PEOPLE = WAKE SERVANT;

% % *********************** NET-
WORK % ***********************
% INITIALIZE PERSONALITY CHARACTERISTICS NOT TO
% BE DESCRIBED IN OUTPUT.
$NETWORK:
LADYBUX COURAGE = 2,
LADYBUX VIOLENT = 1,
JOHNBUX IQ = 100,
JOHNBUX COURAGE = -1,
DRHUME WEALTH = -2,
DRHUME VIOLENT = 3,
DRHUME AFFECTION = -1 LORDED,
DRHUME AFFECTION = -1 RONALD,
DRHUME AFFECTION = -1 LADYBUX,
LORDED IQ = 100,
LORDED COURAGE = 1,
LORDED MARRIED,
LORDED AFFECTION = 1 DRHUME,
LADYJANE WEALTH = 3,
LADYJANE IQ = 100,
LADYJANE VIOLENT = -1,
LADYJANE MARRIED,
RONALD IQ = 110,
RONALD VIOLENT = -1,
RONALD JEALOUS = 1, RONALD MARRIED,
CATHY IQ = 100,
CATHY WEALTH = 2,
CATHY MARRIED,
JAMES COURAGE = 2,
JAMES MARRIED,
MARIAN COURAGE = 2,
MARIAN MARRIED,
BUTLER VIOLENT = -1,
NURSE IQ = 100,
MAID COURAGE = -2,
COOK IQ = 100,
COOK COURAGE = 2,
SUN FORCAST = 15;
% DEFINE COMPOUND RELATIONS IN TERMS OF
% INDIVIDUAL RELATIONS.
% *LEXT (GO FOR) TO GOFOR;
*LEXT (MAD AT) TO MADAT;
*LEXT (GET UP) TO GETUP;
*LEXT (GAME OF CROQUET) TO CROOGAME;
$LOOP:
$RULE:
P.PICK(RENDEVOUS);
*DISABLE STARTWALK.
*REMOVE P FROM RENDEVOUS,
*MOVE P TO REMDM;
*ADD P TO INTERRUPT;
*ADD WANTED(P) TO INTERRUPT;
*ENABLE REND IN 20M;
$ENDGROUP
$RULE:
T($ENDGROUP)
*ENABLE AFTERN IN 15M;
$RULE:
$ENDLOOP;
CLOCK EQ 15H;
*ENABLE NIGHT IN 15M;

GROUP AFTERN BEGINS AN AFTERNOON TRYST.

$GROUP AFTERN:

1H/OFF;
P.RENDM;
W.WANTED(P);
C.CHASER(P);
*DISABLE AFTERN,
*MOVE GREENHS TO MROOM,
*MOVE WINDOW TO WEAPON,
*MOVE HOUSE TO MOTIVE,
*INSERT(W DECIDE)(DECIDE GOFOR WALK),
W.SMILEAT P,
*INSERT (P SEE THAT)(W GOTO GARDEN),
P.FOLLOW W,
*INSERT (C SEE THAT)(P FOLLOW W),
*INSERT (C THINK THAT) (P AFFECTION = 3 W),
W.WALKIN GARDEN,
C.FOLLOW P,
P.MEET W;

GROUP NIGHT BEGINS A NIGHT-TIME TRYST.

$GROUP NIGHT:

1H/OFF;
P.RENDM;
W.WANTED(P);
C.CHASER(P);
*DISABLE NIGHT,
*MOVE LIBRARY TO MROOM,
*MOVE DOOR TO WEAPON,
*MOVE BED TO MOTIVE,
P.AWAKE,
P.GETUP,
*INSERT (P THINK THAT)(SPOUSE(P) ASLEEP);
(P MARRIED);
*INSERT (P PLANS)(PLANO MEET W),
P ENTER HALL,
W GETUP,
W GOTO HALL,
C KNOW PLAN,
*INSERT (C DECIDE)(DECIDE FOLLOW THEY),
ULST XX,
*INSERT (C DECIDE)(DECIDE FOLLOW P),
*INSERT (C DECIDE)(DECIDE FOLLOW W),
ENDLOOP;
ENDLOOP;
ENDGROUP;
% GROUP REND CONTROLS THE ACTUAL TRYST ITSELF.
% THERE ALWAYS AN OBSERVER INVOLVED: HIS
% (OR HER) ACTIONS DEPEND ON HIS RELATIONSHIP TO
% THE OTHER TWO AND ON HIS OWN PERSONALITY.
GROUP REND: IH/OFF;
RULE: *DISABLE REND;
$LOOP: P.RENDM;
$LOOP: W.WANTED(P);
$LOOP: C.CHASER(P);
RULE: *REMOVE P FROM INTERRUPT,
*REMOVE C FROM INTERRUPT,
*REMOVE W FROM INTERRUPT,
RULE: P KISS W;
: .5;
RULE: W CARESS P;
: .5;
RULE: W KISS P;
: .5;
RULE: ULST XX,
ULST XX,
GOTO MROOM,
GOTO MROOM,
FOLLOW P,
FOLLOW W,
LIST XX,
THEY GOTO MROOM,
FOLLOW THEY,
W UNDRESS,
P F**K W;
RULE: S.SPOUSE(P);
RULE: *INSERT (C SEE THAT)(P F**K W),
: .5;
RULE: P COMMIT ADULTERY,
*ADD P TO POSVICTM(S),
*ADD S TO POSKILLR,
S WHYKILL = 2 P,
*ADD W TO POSVICTM(S),
S WHYKILL = 3 W,
10,-10:
(P MARRIED);
ENDLOOP;
$LOOP: S.SPOUSE(W);
$RULE: W COMMIT ADULTRY,
*ADD S TO POSKILLR,
*ADD W TO POSVICTM(S),
S WHYKILL = 2 W,
*ADD P TO POSVICTM(S),
S WHYKILL = 3 P;
ENDLOOP;
$SWITCH: T(L1);
10,-10:
(C EQL SPOUSE(P) OR (C EQL SPOUSE(W)));
RULE: *ADD P TO POSKILLR,
P WHYKILL = 5 C,
*ADD C TO POSVICTM(P),
*INSERT (C DECIDE)(DECIDE BLACKMAIL P),
*INSERT (P LEAVE MROOM)(LEAVE WITH W),
C ACCOST P,
C BLACKMAIL P;
RULE: T(L5)
*INSERT (C THREATEN)(THREATEN TELL SPOUSE(P));
10,-10:
(P MARRIED);
RULE: *INSERT(C THREATEN)(THREATEN TELL SPOUSE(W));
RULE L5:
MADAT C;
: .5;
RULE: *INSERT(P THREATEN)(THREATEN KILL C):
2508 (P VIOLENT)/9 + .5;
2509 $RULE: P AFRAID;
2510 *INSERT (P AGREE)(AGREE PAY C);
2511 (P WEALTH)/8 + .5;
2512 $RULE: P AFFECTION = -3 C;
2513 %
2514 $RULE L1:
2515 C ENRAGED;
2516 (C JEALOUS)/7 + .6;
2517 $RULE: F(L2)
2518 C ENTER MROOM,
2519 C YELLAT P;
2520 (C VIOLENT)/8 + .5;
2521 $RULE: C CRY;
2522 10,-10:
2523 (C EQL FEMALE);
2524 $RULE: T(L5)
2525 *INSERT (C THREATEN)(THREATEN KILL P);
2526 (C VIOLENT)/7 + .5;
2527 $RULE: W EMBARASD;
2528 .7;
2529 $RULE: W CRY;
2530 .5;
2531 $RULE: (L1)
2532 *INSERT(SPOUSE(C) ASK C)(ASK FORGIVE SPOUSE(C));
2533 7;
$RULE L2:
2534 C LOOKTHRU WEAPON,
2535 MX QO = 1,
2536 C HIDDEN;
2537 $RULE: C MADAT SPOUSE(C);
2538 (C JEALOUS)/8 + .5;
\$RULE:
\$RULE:
\$RULE:
\$RULE:
\$RULE L3:
\$RULE:
\$RULE:
\$RULE:
\$RULE:
\$RULE:
\$RULE:
\$GROUP DOKILL:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$SWITCH:
\$RULE:
\$RULE: 1H/OFF;
\$SWITCH: T(LI);
10, -10:  NUM(POSKILLR) EQ 0;
\$LOOP: K.PICK(POSKILLR);
\$LOOP: V.PICK(POSKILLR);
\$RULE:  *ENABLE FINDING IN 1H,
   *DISABLE DOKILL,
   *REMOVE V FROM RETIRED,
   *REMOVE V FROM PEOPLE,
   *REMOVE V FROM SUESTS,
   *REMOVE V FROM SERVANT,

   *REMOVE V FROM MALE,
   *REMOVE V FROM FEMALE,
   *MOVE K TO KILLER,
   *MOVE V TO VICTIM;

   T(KSA);

   10, -10:  (K WHYKILL V) EQ 2;
   T(KSL);

   10, -10:  (K WHYKILL V) EQ 3;
   T(KB);

   10, -10:  (K WHYKILL V) EQ 5;
   T(KR);

   10, -10:  (K WHYKILL V) EQ 6;
   T(KBP);

   10, -10:  T(KLB);

   10, -10:  (K WHYKILL V) EQ 9;
   T(KLB);

   10, -10:  (K WHYKILL V) EQ 9;

   10, -10:  (ENDGROUP)

   *PRINT '****ERROR: NO MOTIVE',
   *PRINT K,
   *PRINT V,
   *END;

   STABBING YOUR SPOUSE FOR ADULTRY.

   *INSERT (K KNOW THAT)(V COMMIT ADULTRY),
   K ENRAGED;

   *INSERT (K MADAT V)(MADAT VERY),
   *INSERT (K DECIDE)(DECIDE STAB V),
   DAY IS SUNDAY,
   TIME IS DAWN,

   *INSERT (V A WAKE)(AWAKE EARLY),
   *INSERT (V DECIDE)(DECIDE GOFOR WALK),
   *INSERT (V GETUP)(GETUP QUIETLY),
   *INSERT (V THINK THAT)(K ASLEEP);

   ULST XX;

   V OEDRESS.
   LST XX;

   V GOTO GARDEN,
   K FOLLOW V;

   V SEE K;

   *INSERT (K HAVE KNIFE)(KNIFE LONG);

   *INSERT (K WAVE KNIFE)(WAVE WILDLY),
   K STAB V,
   MX QQ = 1,
   V SCREAM;

   *INSERT (KNIFE SINK)(SINK DEEP);

   *INSERT (V STRUGGLE)(STRUGGLE WEAKLY);

   *INSERT (V HIT K);

   .6;
7.6. Sample Murder Mystery Texts

We offer a 2100 word story, complete with semantic deep structure, generated in under 19 seconds. We also offer selected murder scenes from other runs that used different random number sequences and/or different character trait specification for Dr. Hume. (In some runs he was made very lustful and evil.) The change stack listing does show all triple linkages that are tabulated by the system.

7.6.1. A 2100 Word Murder Mystery Story

The following material is a short sample of the actual computer output, plus the full text of an actual story, minus deep structure, and edited for capitalization and spacing within paragraphs.
CHANGE STACK FOR TIME 19W3D10H

1:  (LADYBUX WEALTH) = 3.0000
2:  (MX QQ) = 2.0000
3:  (LADYBUX GOOD) = 3.0000
4:  (LADYBUX IQ) = 125.0000
5:  (LADYBUX SINGLE) SET AT 19W3D10H
6:  (MX QQ) = 2.0000
7:  (LADYBUX ATTRACTI) = -2.0000
8:  (LADYBUX SEXDRIVE) = 4.0000
9:  (JOHNBUX IS NEPHEW) SET AT 19W3D10H
10: (NEPHEW POS LADYBUX) SET AT 19W3D10H
11: (JOHNBUX GOOD) = -3.0000
12: (MX QQ) = 2.0000
13: (JOHNBUX WEALTH) = -2.0000
14: (JOHNBUX VIOLENT) = 1.0000
15: (JOHNBUX SINGLE) SET AT 19W3D10H
16: (MX QQ) = 2.0000
17: (JOHNBUX HANDSOME) = 3.0000
18: (JOHNBUX SEXDRIVE) = 3.0000
19: (JOHNBUX AFFECTIO LORDEO) = -2.0000
20: (JOHNBUX AFFECTIO DRHUME) = -2.0000
21: (DRHUME GOOD) = -3.0000
22: (MX QQ) = 2.0000
23: (DRHUME IQ) = 150.0000
24: (DRHUME COURAGE) = 3.0000
25: (DRHUME SEXDRIVE) = 4.0000
26: (DRHUME SINGLE) SET AT 19W3D10H
27: (MX QQ) = 1.0000
28: (DRHUME HANDSOME) = 1.0000
29: (LORDED WEALTH) = 3.0000
30: (MX QQ) = 2.0000
31: (LORDED GOOD) = 2.0000
32: (LORDED VIOLENT) = -1.0000
33: (LORDED HANDSOME) = -1.0000
34: (MX QQ) = 1.0000
35: (LORDED SEXDRIVE) = 2.0000
36: (LORDED MARRIED) SET AT 19W3D10H
37: (MARRIED TO LADYJANE) SET AT 19W3D10H
38: (LORDED AFFECTIO LADYJANE) = 1.0000
39: (LORDED JEALOUS) = -1.0000
40: (LORDED AFFECTIO JOHNBUX) = -1.0000
41: (LADYJANE AFFECTIO LORDED) = 1.0000
42: (MX QQ) = 2.0000
43: (LADYJANE ATTRACTI) = 1.0000
44: (LADYJANE JEALOUS) = 1.0000

WONDERFUL SMART LADY BUXLEY WAS RICH.
UGLY OVERSEXED LADY BUXLEY WAS SINGLE.
JOHN WAS LADY BUXLEY’s NEPHEW.
IMPOVERISHED IRRITABLE JOHN WAS EVIL.
HANDSOME OVERSEXED JOHN BUXLEY WAS SINGLE.
JOHN HATED EDWARD.
JOHN BUXLEY HATED DR. BARTHOLOMEW HUME.

BRILLIANT BRAVE HUME WAS EVIL.
HUME WAS OVERSEXED.
HANDSOME DR. BARTHOLOMEW HUME WAS SINGLE.
KIND EASY GOING EDWARD WAS RICH.
OVERSEXED LORD EDWARD WAS UGLY.
LORD EDWARD WAS MARRIED TO LADY JANE.
EDWARD LIKED LADY JANE.
EDWARD WAS NOT JEALOUS.
LORD EDWARD DISLIKED JOHN.
PRETTY JEALOUS JANE LIKED LORD EDWARD.

CHANGE STACK FOR TIME 19W3D10H1M

1:  (RONALD GOOD) = 2.0000
2:  (MX QQ) = 1.0000
3:  (RONALD WEALTH) = 2.0000
4:  (RONALD MARRIED) SET AT 19W3D10H1M
5:  (MARRIED TO CATHY) SET AT 19W3D10H1M
6:  (MX QQ) = 1.0000
7:  (RONALD SEXDRIVE) = 1.0000
8:  (RONALD AFFECTIO CATHY) = 3.0000
9:  (MX QQ) = 1.0000
10: (RONALD HANDSOME) = 1.0000
11: (RONALD AFFECTIO DRHUME) = 1.0000
12: (RONALD AFFECTIO JAMES) = -1.0000
13: (CATHY GOOD) = 2.0000
14: (MX QQ) = 2.0000
15: (CATHY VIOLENT) = -2.0000
16: (CATHY SEXDRIVE) = 1.0000
17: (CATHY AFFECTIO RONALD) = 3.0000
18: (MX QQ) = 2.0000
19: (CATHY ATTRACTI) = 2.0000
20: (CATHY JEALOUS) = 1.0000
21: (JAMES IS PARTNER2) SET AT 19W3D10H1M
22: (PARTNER2 POS RONALD) SET AT 19W3D10H1M
23: (JAMES AFFECTIO RONALD) = -3.0000
24: (JAMES IQ) = 80.0000
25: (MX QQ) = 2.0000
26: (JAMES GOOD) = -3.0000
27: (JAMES VIOLENT) = 3.0000
28: (JAMES MARRIED) SET AT 19W3D10H1M
29: (MARRIED TO MARION) SET AT 19W3D10H1M
30: (MX QQ) = 2.0000
31: (JAMES SEXDRIVE) = -3.0000
32: (JAMES HANDSOME) = -3.0000
33: (JAMES AFFECTED MARION) = -1.0000
34: (MX QQ) = 2.0000

CHANGE STACK FOR TIME 20W2D1H
CHANGE STACK FOR TIME 20W2D1H15M
CHANGE STACK FOR TIME 20W2D1H25M
CHANGE STACK FOR TIME 20W2D1H35M
1: NOT (DRHUME PLAY BRIDGE) SET AT 20W1D2H15SM
2: NOT (JOHNBUX PLAY BRIDGE) SET AT 20W1D2H15SM
3: NOT (LADYBUX PLAY BRIDGE) SET AT 20W1D2H15SM
4: NOT (LADYJANE PLAY BRIDGE) SET AT 20W1D2H15SM
5: NOT (THEY PLAY BRIDGE) SET AT 20W1D2H15SM
6: (CARDGAME OVER) SET AT 20W2D1H35M

THE CARD GAME WAS OVER.

CHANGE STACK FOR TIME 20W2D2H
CHANGE STACK FOR TIME 20W2D2H15M
1: (JOHNBUX AWEAKE) SET AT 20W2D2H15M
2: (JOHNBUX GETUP) SET AT 20W2D2H15M
3: (JOHNBUX PLANNO) SET AT 20W2D2H15M
4: (PLANNO MEET MARION) SET AT 20W2D2H15M
5: (JOHNBUX ENTER HALL) SET AT 20W2D2H15M
6: (MARION GETUP) SET AT 20W2D2H15M
7: NOT (MARION GOTO PARLOR) SET AT 20W1D2H015M
8: (MARION GOTO HALL) SET AT 20W2D2H15M
9: (JAMES KNOW PLAN) SET AT 20W2D2H15M
10: (JAMES DECIDE) SET AT 20W2D2H15M
11: (DECIDE FOLLOW THEY) SET AT 20W2D2H15M
12: (ULST XX) SET AT 20W2D2H15M
13: (JAMES DECIDE) SET AT 20W2D2H15M
14: (DECIDE FOLLOW JOHNBUX) SET AT 20W2D2H15M
15: (JAMES DECIDE) SET AT 20W2D2H15M
16: (DECIDE FOLLOW MARION) SET AT 20W2D2H15M
17: (ULST XX) SET AT 20W2D2H15M

JOHN AWOKE.
JOHNBUXLEY GOT UP.
JOHN PLANNED TO MEET MARION.
JOHN ENTERED THE CORRIDOR.
MARION GOT UP.
MARION WENT TO THE HALL.
JAMES KNEW THE PLAN.
JAMES DECIDED TO FOLLOW THEM.

CHANGE STACK FOR TIME 20W2D2H20M
1: (JOHNBUX KISS MARION) SET AT 20W2D2H20M
2: (MARION KISS JOHNBUX) SET AT 20W2D2H20M
3: (ULST XX) SET AT 20W2D2H20M
4: NOT (JOHNBUX GOTO PARLOR) SET AT 20W1D2D0H15M
5: (JOHNBUX GOTO LIBRARY) SET AT 20W2D2H20M
6: NOT (MARION GOTO HALL) SET AT 20W2D2H15M
7: (MARION GOTO LIBRARY) SET AT 20W2D2H20M
8: (JAMES FOLLOW JOHNBUX) SET AT 20W2D2H20M
9: (JAMES FOLLOW MARION) SET AT 20W2D2H20M
10: (ULST XX) SET AT 20W2D2H20M
11: NOT (THEY GOTO GREENHS) SET AT 20W1D1H15H20M
12: (THEY GOTO LIBRARY) SET AT 20W2D2H20M
13: (JAMES FOLLOW THEY) SET AT 20W2D2H20M
14: (MARION UNDRESS) SET AT 20W2D2H20M
15: (JOHNBUX FUCK MARION) SET AT 20W2D2H20M
16: (MARION COMMIT ADULTERY) SET AT 20W2D2H20M
17: (JAMES ENRAGES) SET AT 20W2D2H20M
18: (JAMES ENTER LIBRARY) SET AT 20W2D2H20M
19: (JAMES YELLAT JOHNBUX) SET AT 20W2D2H20M
20: (JAMES THREATEN) SET AT 20W2D2H20M
21: (THREATEN KILL JOHNBUX) SET AT 20W2D2H20M
22: (MARION EMBARRASO) SET AT 20W2D2H20M
23: (MARION CRY) SET AT 20W2D2H20M
24: (EVERYONE GOTO BED) SET AT 20W2D2H20M
25: (ULST XX) SET AT 20W2D2H20M
26: (JAMES GOTO BED) SET AT 20W2D2H20M
27: (JOHNBUX GOTO BED) SET AT 20W2D2H20M
28: (MARION GOTO BED) SET AT 20W2D2H20M
29: (ULST XX) SET AT 20W2D2H20M

JOHNBUXLEY KISSED MARION.
MARION KISSED JOHN.
THEY WENT TO THE LIBRARY.
JAMES FOLLOWED THEM.
MARION UNDRESSED.
JOHNBUXLEY SCREWED MARION.
MARION COMMITED ADULTERY.
JAMES WAS ENRAGED.
JAMES ENTERED THE LIBRARY.
JAMES YELLED AT JOHN.
JAMES THREATENED TO KILL JOHNBUXLEY.
MARION WAS EMBARRASSED.
MARION CRIED.
EVERYONE WENT TO BED.

CHANGE STACK FOR TIME 20W2D3H
CHANGE STACK FOR TIME 20W2D4H
CHANGE STACK FOR TIME 20W2D5H
CHANGE STACK FOR TIME 20W2D6H
1: (JAMES RICH) SET AT 20W2D6H
2: (RICH VERY) SET AT 20W2D6H
3: (BUTLER WEALTH) = - 10000
4: (BUTLER WANT MO: IF AT 20W2D6H
5: (BUTLER RELATED JAMIES) SET AT 20W2D6H
6: (BUTLER DECIDE) SET AT 20W2D6H
7: (DECIDE POISONS JAMES) SET AT 20W2D6H
S. Klein et al.

MURDER MYSTERY 1 by Novel Writer Simulation Program

(Edited only for capitalization and spacing within paragraphs. Original paragraphing, reflecting sequential time frames, is preserved)

Wonderful smart Lady Buxley was rich. Ugly oversexed Lady Buxley was single. John was Lady Buxley’s nephew. Impoverished irritable John was evil. Handsome oversexed John Buxley was single. John hated Edward. John Buxley hated Dr. Bartholomew Hume. Brilliant Hume was evil. Hume was oversexed. Handsome Dr. Bartholomew Hume was single. Kind easy going Edward was rich. Oversexed Lord Edward was ugly. Lord Edward married to Lady Jane. Edward liked Lady Jane. Edward was not jealous. Lord Edward disliked John. Pretty jealous Jane liked Lord Edward.

Well to do Ronald was kind. Lusty Ronald was married to Cathy. Handsome Ronald loved Catherine. Ronald liked Hume. Ronald disliked James. Easy going lusty Cathy was kind. Beautiful jealous Catherine loved Ronald. James was Ronald’s partner. James hated Ronald. Evil violent James was dumb. Impotent ugly James was married to Marion. Well to do Jealous James disliked Marion. James disliked Dr. Bartholomew Hume. Unpleasant violent Marion was smart. Beautiful Marion was impoverished. Jealous oversexed Marion hated James. Marion disliked Florence.

Florence was Lady Buxley’s companion. Wonderful Florence was easy going. Beautiful oversexed Florence was single. The smart unpleasant butler was lusty. Poor brave butler was single. The dumb maid was good. Pretty poor Heather was single. Ugly violent cook was single. The cook was poor.

The day was Tuesday. The weather was rainy. Marion was in the park. Dr. Bartholomew Hume ran into Marion. Hume talked with Marion. Marion flirted with Hume. Hume invited Marion. Dr. Hume liked Marion. Marion liked Dr. Bartholomew Hume. Marion was with Dr. Bartholomew Hume in the hotel. Marion was near Hume. Dr. Hume caressed Marion with passion. Hume was Marion’s lover. Lady Jane following them saw the affair. Jane blackmailed Marion. Marion was impoverished. Jane was rich.

Marion phoned Jane in the morning. Marion invited Jane to go to a theater. Jane agreed. Jane got dressed for the evening. They met them in the theater. Jane introduced Lord Edward during an intermission to Marion.

The day was Wednesday. The weather was windy. Lady Jane was in the tennis court. John ran into Lady Jane. John talked with Jane. Lady Jane flirted with John Buxley. John Buxley invited Lady Jane. John liked Lady Jane. Lady Jane liked John. John Buxley was with Jane in a movie. John was near Lady Jane. Jane caressed John Buxley with passion. Lady Jane was John’s lover. Cathy following them saw the affair. Cathy blackmailed Lady Jane. Jane was well to do. Lady Catherine was rich.

Lady Catherine invited Jane to play bridge. Lady Catherine told Marion to come with Lady Buxley. Jane asked them to sit down. Lady Jane brought the
cards. Jane offered drinks. Lady Buxley asked for whiskey on the rocks. The others had coffee with cookies. Jane shuffled the cards. Lady Jane started a game. Marion casually signaled Lady Buxley with hands. Jane noticed it. Lady Jane suspected that they cheated. Jane watched them closely. Marion won the game with Lady Buxley. Jane was upset with Catherine. Lady Jane disliked Marion.

The day was Thursday. The weather was rainy. A small pub was on a corner. John Buxley was in the pub. John Buxley asked for whiskey on the rocks. John got a drink from the barman. John talked with Hume near the bar. Hume sang the Beatles's song. John Buxley was drunk. James said that Marion committed adultery. James thought that James was drunk. James was depressed. James left the pub. Edward said that Lady Jane committed adultery. John Buxley thought Lord Edward was drunk. Lord Edward was depressed. Lord Edward left the pub.

The day was Friday.

Lady Buxley had a big house. Lady Buxley's house had a pretty fragrant garden. A green house was in the garden. The garden was near the tennis court. The house had a big bright dining room. The house also had a pleasant parlor. A cool dark musty library was near the parlor. The time was evening. Lady Buxley gave a party. The party lasted for a weekend.

Lady Buxley talked with Florence
Marion arrived with James.
Catherine arrived with Ronald.
Edward arrived with Jane.
Dr. Hume arrived. Dr. Bartholomew Hume joined a conversation.

Catherine talked with Dr. Bartholomew Hume. Dr. Bartholomew Hume flirted with Lady Catherine. Dr. Bartholomew Hume said that Lady Catherine was beautiful. Dr. Hume wanted to seduce Catherine. Hume told a joke. Catherine laughed.


The servants went to bed.


John Buxley talked with Jane. John Buxley casually mentioned politics. Lady Jane discussed politics with John Buxley. Lady Jane said that the weather was nice.


Everyone went to bed.

The day was Saturday. The sun rose. The servants got up. The cook went to the kitchen. The cook prepared a breakfast. Clive followed the cook. Clive seduced Maggie in the kitchen.

The day was beautiful. They got up. They got dressed. They went down to the breakfast.


The breakfast was over. James talked with Lady Buxley. James casually mentioned a music. Lady Buxley discussed the music with James.

Everyone went to the parlor.


Dr. Hume asked Lord Edward to play chess. Edward agreed. Lord Edward went to the study with Dr. Hume. They played chess. Hume was a good player. Lord Edward played chess well.


Lady Buxley talked with Florence.

The cook went to the kitchen. Maggie prepared lunch.

Ronald talked with Lady Buxley.

Florence talked with Hume. Florence casually mentioned fashion. Dr. Bartholomew Hume hated the conversations about fashion.

Lunch was over. The men went to the parlor. The men smoked cigars. The women went to the drawing room. The women drank whiskey.


Marion talked with John Buxley. John Buxley flirted with Marion. John Buxley gently touched Marion. Marion smiled at John. John Buxley wanted to seduce Marion. Marion wanted to seduce John Buxley. James saw that Marion talked with John. James was mad at John. James overhearing Marion was angry. Marion saw that James was upset. Marion talked with James.

The butler announced tea.

Everyone went to the garden. The butler served tea. The day was cool. The sky was cloudy. The garden was nice. The flowers were pretty. Marion complimented Lady Buxley.

Ronald talked with Marion.

Tea time was over.

Everyone went to the parlor.

The cook went to the kitchen. Maggie prepared dinner.

Dr. Hume asked Edward to play tennis. Edward agreed. Lord Edward went to the tennis court with Dr. Hume. They played tennis. Dr. Hume was the good player. Edward played tennis well.

The butler announced dinner.

Dr. Bartholomew Hume stopped playing tennis. Edward stopped playing tennis.

Everyone went to the dining room. Everyone sat down. The butler served the food. Supper started.

Marion talked with Florence. Florence argued with Marion. Marion said that Florence was idiotic.

Florence talked with Lady Buxley.

Supper was over. The men went to the parlor. The men smoked fat smelly stogies. The men drank sherry. The women went to the drawing room. The women gossiped drunk coffee.

Everyone went to the parlor.

Marion talked with Jane.

James went to the library. James read the good paperback. Edward asked Ronald to play tennis. Ronald agreed. Ronald went to the tennis court with Lord Edward. They played tennis.

John suggested the game of bridge. Lady Buxley agreed. Dr. Bartholomew Hume. Jane agreed. They played bridge.

The servants went to bed. Everyone went to bed.

James stopped reading the book.


John Buxley cheated at bridge.

John cheated at bridge.

The card game was over.


James was very rich. Clive was impoverished. Clive wanted the money. The butler was related to James. The butler decided to poison James. Clive thought that Clive inherited the money. Clive knew that James drank a milk. Clive poisoned the milk. James drank the milk. James went to bed. James died. The others thought that James was asleep. Clive removed the fingerprints. The butler returned the bottle.

Ronald awakened. Ronald got up. Ronald thought that the day was beautiful. Ronald found James. Ronald saw that James was dead. Ronald yelled. The others awakened. The others ran to Ronald. The others saw James. Everyone talked. Heather called the policemen. Hume examined the body. Dr. Bartholomew Hume said that James was killed by poison.

John talked with Edward about the murder.
Edward talked with Maggie about the murder. Maggie was upset about the murder.

The cops arrived. The cops were idiotic. A detective examined the corpse. The policemen looked for hints in the bathroom. Dr. Bartholomew Hume also looked. Edward tried to calm Marion.

The policemen questioned Dr. Bartholomew Hume. The detective asked questions. The policemen searched the garden. The policemen tried to find clues. Marion cried.

Dr. Bartholomew Hume searched stairs. Hume looked for hints. Dr. Hume questioned Lady Buxley. Dr. Hume knew that Lady Buxley told the truth. Florence talked with Heather about the murder. Marion cried.

The policemen questioned Ronald. The inspector suspected Ronald. The inspector asked the stupid questions. The policemen searched the parlor. The policemen tried to find hints. Florence was upset.

Dr. Bartholomew Hume searched the dining room. Dr. Bartholomew Hume looked for hints.

The cops questioned Heather. The detective asked the stupid questions. Dr. Hume questioned Heather. Dr. Hume knew that Heather told the truth. The cops searched the tennis court. Clive talked with Ronald about the murder. The butler said that James was kind. The cook talked about the murder.

Dr. Bartholomew Hume searched the bathroom. Dr. Hume looked for clues. Marion cried.

Dr. Hume questioned Florence. Hume knew that Florence told the truth. Dr. Bartholomew Hume got information from Florence. The cops searched the bathroom. The cops found a thread. The thread was misleading clue. Lady Buxley talked with John about the murder. Lady Buxley said that James was kind. Dr. Hume was upset.

Dr. Bartholomew Hume searched the library. The cops questioned John Buxley. The detective asked the stupid questions. Hume questioned the cook. Dr. Bartholomew Hume knew that Maggie told the truth. Hume got information from the cook.

Hume went to the bathroom. Dr. Hume found the bottle Hume knew the murderer. Hume asked everyone to go to the parlor. Dr. Bartholomew Hume said that the murderer was in the room. Everyone was surprised. Everyone talked. Dr. Bartholomew Hume said that James was killed by poison. Hume said that the butler killed James. Everyone was shocked. The butler drew a pistol. Clive headed for the door. Dr. Bartholomew hume followed Clive. The butler shot at Hume. Dr. Bartholomew Hume grabbed a paperweight. Dr. Bartholomew Hume threw the paperweight at Clive. The paperweight hit Clive in the head. Clive fell. Dr. Bartholomew Hume took the gun. The policemen took Clive. Ronald congratulated Hume. Clever Dr. Hume solved the crime.

7.6.2. Murder and Solution from Story 2

JAMES KNEW THAT HUME SCREWED MARION.
JAMES HATED DR. BARTHOLOMEW HUME.
JAMES WANTED A REVENGEB.
JAMES DECIDED TO KILL DR. HUME.
JAMES WROTE A NOTE.
DR. HUME GOT THE NOTE FROM JAMES.
HUME MET JAMES.
THE DAY WAS SUNDAY.
THE TIME WAS THE DAWN.
JAMES GOT UP.
JAMES WENT TO THE LIBRARY.
DR. BARTHOLOMEW HUME WENT TO THE LIBRARY.
HUME THOUGHT THAT JAMES WAS UNAWARE.
JAMES SAID THAT DR. BARTHOLOMEW HUME WAS EVIL.
JAMES POINTED A PISTOL AT DR. BARTHOLOMEW HUME.
DR. HUME SAW THE PISTOL.
HUME ATTACKED JAMES.
DR. BARTHOLOMEW HUME HIT JAMES IN THE BELLY.
DR. BARTHOLOMEW HUME TRIED TO GRAB THE PISTOL.
JAMES HIT HUME.
JAMES STRUGGLED WITH DR. BARTHOLOMEW HUME.
JAMES KEPT THE PISTOL.
JAMES HIT DR. BARTHOLOMEW HUME.
HUME STAGGERED BACK.
DR. BARTHOLOMEW HUME DIED.
JAMES HID THE GUN.
JAMES LOOKED FOR THE NOTE.
THE NOTE WAS GONE.
JAMES RETURNED TO THE BEDROOM.

LADY JANE AWAKENED.
LADY JANE GOT UP.
JANE THOUGHT THAT THE DAY WAS BEAUTIFUL.
JANE FOUND DR. BARTHOLOMEW HUME.
LADY JANE SAW THAT HUME WAS DEAD.
LADY JANE SCREAMED LOUD.
LADY JANE FAINTED.
THE OTHERS AWAKENED.
THE OTHERS RAN TO LADY JANE.
THE OTHERS SAW DR. BARTHOLOMEW HUME.
EVERYONE TALKED.
EDWARD CALLED THE COPS.
FLORENCE EXAMINED THE CORPSE.
FLORENCE SAID THAT DR. BARTHOLOMEW HUME WAS KILLED BY THE GUN.
THE POLICEMEN ARRIVED.
THE COPS WERE IDIOTIC.
A DETECTIVE EXAMINED THE CORPSE.
THE COPS LOOKED FOR CLUES IN THE LIBRARY.
FLORENCE ALSO LOOKED.
FLORENCE TALKED WITH THE COOK ABOUT THE MURDER.
THE COOK WAS UPSET ABOUT THE MURDER.
JAMES SAID THAT RONALD KILLED DR. HUME.
RONALD DENIED THE ACCUSATION.
RONALD SAID THAT JAMES WAS STUPID.

THE COPS QUESTIONED FLORENCE.
THE DETECTIVE SUSPECTED FLORENCE.
THE INSPECTOR ASKED QUESTIONS.
LADY CATHERINE TALKED ABOUT THE MURDER.

FLORENCE SEARCHED THE PARLOR.
FLORENCE LOOKED FOR HINTS.
FLORENCE QUESTIONED THE BUTLER.
FLORENCE GOT INFORMATION FROM CLIVE.

FLORENCE SEARCHED THE LIBRARY.
FLORENCE LOOKED FOR HINTS.
THE COPS QUESTIONED LADY JANE.

FLORENCE SEARCHED THE LIBRARY.
FLORENCE FOUND ASHES.
THE ASHES WERE VALUABLE CLUE.
THE POLICEMEN QUESTIONED RONALD.
THE INSPECTOR ASKED THE QUESTIONS.
JAMES TALKED ABOUT THE MURDER.

FLORENCE QUESTIONED MARION.
FLORENCE KNEW THAT MARION TOLD THE TRUTH.
FLORENCE GOT INFORMATION FROM MARION.

THE COPS QUESTIONED HEATHER.
THE INSPECTOR ASKED THE QUESTIONS.
THE COPS SEARCHED THE DRAWING ROOM.
THE POLICEMEN FOUND A THREAD.
THE THREAD WAS MISLEADING CLUE.
CATHERINE TALKED WITH THE BUTLER ABOUT THE MURDER.
CATHY SAID THAT DR. BARTHOLOMEW HUME WAS KIND.
THE BUTLER AGREED.
CLIVE WAS UPSET ABOUT THE MURDER.

FLORENCE WENT TO THE LIBRARY.
FLORENCE FOUND THE NOTE.
FLORENCE KNEW THE KILLER.
FLORENCE ASKED EVERYONE TO GO TO THE PARLOR.
FLORENCE SAID THAT THE MURDERER WAS IN THE ROOM.
EVERYONE WAS SURPRISED.
EVERYONE TALKED.
FLORENCE SAID THAT DR. HUME WAS KILLED BY THE PISTOL.
FLORENCE SAID THAT JAMES KILLED DR. BARTHOLOMEW HUME.
EVERYONE WAS SHOCKED.
JAMES DREW THE GUN.
JAMES HEADED FOR THE DOOR.
FLORENCE TRIpped JAMES.

JAMES FELL.
FLORENCE STRUGGLED WITH JAMES.
THE GUN FIRED.
FLORENCE GOT THE GUN.
THE COPS TOOK JAMES TO THE JAIL.
THE POLICEMEN CONGRATULATED FLORENCE.
CLEVER FLORENCE SOLVED THE CRIME.

7.6.3. Murder Scene from Story 3

DR. BARTHOLOMEW HUME BLACKMAILED EDWARD.
EDWARD WAS AFRAID OF DR. HUME.
LORD EDWARD DECIDED TO KILL DR. BARTHOLOMEW HUME.
THE DAY WAS SUNDAY.
THE TIME WAS THE SUNRISE.
LORD EDWARD GOT UP.
LORD EDWARD WENT TO THE DARK CORRIDOR.
LORD EDWARD HID.
EDWARD HAD A CANDLE HOLDER.
DR. BARTHOLOMEW HUME AWAKENED EARLY.
DR. BARTHOLOMEW HUME WAS USUALLY EARLY.
DR. HUME WENT FOR THE WALK.
EDWARD WATTED FOR HUME.
LORD EDWARD SURPRISED HUME.
EDWARD HIT DR. BARTHOLOMEW HUME WITH THE CANDLE HOLDER.
DR. BARTHOLOMEW HUME GROANED WEAKLY.
DR. HUME DIED.
EDWARD RETURNED TO THE BEDROOM.

7.6.4. Murder Scene from Story 4

LORD EDWARD KNEW THAT LADY JANE COMMITED ADULTERY.
LORD EDWARD WAS ENRAGED.
EDWARD DECIDED TO STAB JANE.
THE DAY WAS SUNDAY.
THE TIME WAS THE SUNRISE.
JANE AWAKENED EARLY.
LADY JANE DECIDED TO GO FOR THE WALK.
JANE GOT UP QUIETLY.
JANE THOUGHT THAT EDWARD WAS ASLEEP.
JANE GOT DRESSED.
JANE WENT TO THE GARDEN.
EDWARD FOLLOWED LADY JANE.
JANE SAW EDWARD.
LORD EDWARD HAD A LONG DAGGER.
EDWARD WAVED THE DAGGER WILDLY.
LORD EDWARD STABBED JANE SCREAMING.
THE KNIFE SANK DEEP.
JANE STRUGGLED WEAKLY.
JANE HIT EDWARD.
LORD EDWARD SLASHED JANE AGAIN.
EDWARD SAID THAT LADY JANE BETRAYED LORD EDWARD.
JANE DYING COVERED WITH THE BLOOD.
LORD EDWARD HID THE KNIFE.
EDWARD RETURNED TO THE BEDROOM.
LORD EDWARD WASHED OFF THE BLOOD.

8.0. References

Bach, E.
1968

Coles, S. L.
1968

Fillmore, C. J.
1968

Green, C. C., Raphael, B.
1968

Heidorn, G. E.
1972
Natural language inputs to a simulation programming system, Report NPS-55HD72101A, Naval Postgraduate School (Monterey California).

Katz, J.
1972

Kellogg, C. H.
1968

Klein, S.
1965a
“Automatic paraphrasing in essay format”, in: Mechanical Translation 8, Nos. 2 and 3 combined.

Klein, S.
1965b
“Control of style with a generative grammar”, in: Language 41, No. 4.

Klein, S.
1965c

Klein, S.
1973

Klein, S.
1967

Klein, S.
1972
Computer simulation of language contact models, UWCS Tech Report 167; also in: Proceedings SECOL 8 Conference, Oct. 26–29, 1972 (Georgetown Univ.).

Klein, S., Davis, B., Fabens, W., Herriot, R., Katke, W., Kuppin, M. A., Towser, A. 1967a
AUTOLING: an automated linguistic fieldworker (Second International Conference on Computational Linguistics, August 1967, Grenoble).

Klein, S., Fabens, W., Herriot, R., Katke, W., Kuppin, M. A., Towser, A. 1968
The AUTOLING system, UWCS Tech Report 43. (Univ. of Wisconsin in Computer Sc. Dept.)

Klein, S., Dennison, T. A.
1971

Klein, S., Kuppin, M. A.
1970

Klein, S., Kuppin, M. A., Meives, K.
1965
“Monte carlo simulation of language change in Tikopia and Maori”, in: Preprints of Papers Presented at the 1969 International Conference on Computational Linguistics (Stockholm: KVAL).}

Klein, S., Lieman, S. L., Lindstrom, G. D.
1966

Klein, S., Oakley, J. D., Suurballe, D. A., Ziesemer, R. A.
1971

Lakoff, G.
1969

McCawley, J. D.
1968


Mel'čuk, I. A. Žolkovskij, A. K.
1970
"Toward a functioning 'meaning-text' model of language", in: *Linguistics 57*.

Petöfi, J. S.
1973

Quillian, R.
1966

Schank, R. C.
1969
*A conceptual dependency representation for a computer-oriented semantics*, (Stanford Univ.: Artificial Intelligence Memo (AIM) 83).

Schank, R. C.
1972

Schank, R. C., Rieger, C. J.
1973
*Inference and the computer understanding of natural language* (Stanford Univ.: Artificial Intelligence Memo (AIM) 197).

Simmons, R. F.
1970
*Some relations between predicate calculus and semantic net representations of discourse* (Austin: Univ. of Texas, Computer Science Dept, preprint).

Simmons, R. F.
1972
"Generating English discourse from semantic networks", in: *Communications of the ACM* 14, No. 10.

Simmons, R. F.
*Compiling semantic networks from English sentences* (in prep.).