A Prototype Natural Language Understanding Legal Consultant

L. Karl Branting

AI TR87-49 February 1987
Contents

1 Introduction 2
2 The Law of Assault and Battery 3
3 Frame Instantiation 4

4 Question Answering 8
   4.1 Simple yes/no questions 9
   4.2 Who questions 9
   4.3 Yes/no questions regarding legal status 10
   4.4 How questions 11

5 The consultation 12
6 The Grammar 12
7 The Taxonomic Tree 13
8 Discussion 13
9 Examples 14
10 Appendices 19
Abstract

Because of the intimate connection between legal reasoning and natural language understanding, a robust natural language interface is important for a useful automated legal reasoning system. This project attempts to demonstrate how existing techniques of natural language understanding and discourse analysis could be used to construct a system able to interpret and resolve legal problems and express their solutions. The program uses frame instantiation both in text understanding and in legal analysis.

1 Introduction

Legal reasoning is a challenging domain for artificial intelligence research. An important source of difficulty in constructing automated legal reasoning systems (ALRS) is that legal doctrines, authorities, problems, arguments, and judgments are all expressed in natural language. An ALRS that does not accept problems and provide output expressed in natural language will necessarily require extensive pre- and post-processing, and its power will be limited by the expressive power of the natural language interface. This suggests that an effective theory of legal discourse may be essential to any useful ALRS, regardless of the formalism used for legal reasoning itself.

This project attempts to demonstrate how existing techniques of natural language understanding and discourse analysis could be used to construct a system able to interpret legal problems and express their solutions. Because constructing a working ALRS of any real depth is a very large-scale project unfeasible for a semester project, the goal here is simply to build a prototype that suggests directions for future research. The law of assault and battery was selected as the domain of the program because the legal doctrines are comparatively simple and because the facts determining whether an assault or battery has occurred can be expressed concisely in terms of physical actions, contacts, intent, apprehension, and consent.

The program itself consists of four parts. First is a grammar capable of parsing simple sentences expressing events that can give rise to intentional torts. The second part is a taxonomic tree that expresses the relationships among the objects and actions described by the sentences. The third part takes the SSR's produced by the grammar and attempts to instantiate frames for "interpersonal transactions", characteristic sequences of events that may constitute a tort. The fourth part consists of question answering procedures,
including some simple rules for determining whether a tort has occurred.

The program was written in HCPRVR running in ELISP.

2 The Law of Assault and Battery

A brief description of the law of assault and battery is helpful to understanding the design of the program. Actions in tort are cases in which an individual, the plaintiff, complains to a court of a wrong committed by another, the defendant, that has resulted in financial loss, physical injury, or other legally recognized harm to the plaintiff. Torts are civil actions, which means that the remedy the court can provide is generally limited to financial compensation.

Assault and battery are termed intentional torts in the law, as distinguished from unintentional torts such as negligence, because intent is an element necessary for the existence of either assault or battery. In the context of intentional torts, intent is defined as follows:

A person intends a result when he acts for the purpose of accomplishing it or believes that the result is substantially certain to follow from his act [Prosser 41].

Battery is the remedy for “intentional and unpermitted physical contact with the plaintiff’s person” [Kionka 77]. An assault is an action that causes in the plaintiff a reasonable apprehension of “immediate harmful or offensive contact with the plaintiff’s person” [Prosser 41].

An act giving rise to either assault or battery must have been intended by the defendant to cause contact or apprehension in the defendant or a third person. However, if the plaintiff consented to the contact or apprehension, or if the defendant was privileged to act as he did, then the defendant is not liable. For example, participants in contact sports are deemed to have consented to contacts consistent with the rules of the game. Self-defense, parental discipline, and authority of law are situations involving privilege [Kionka 77].

To summarize, the elements of battery are contact, intent, and absence of consent or privilege, and the elements of assault are apprehension, intent, and absence of consent or privilege. For simplicity, the program assumes that privilege is absent and that liability turns upon the remaining issues exclusively.
3 Frame Instantiation

The goal of the text-understanding portion of the program is to combine into a coherent whole sentences that describe a single event that could give rise to an assault or battery. This goal is a special case of the coherence analysis described in [Alterman 82]. However, my program assumes that there is only a small set of ways that sentences can combine to describe an event of the relevant kind, so it makes use of just three of Alterman’s coherence relations: SEQ, PREC, and COORD.

The program assumes that events that can give rise to assault or battery involve some subset of the following elements:

- intent on the part of the defendant
- an act of moving some object by the defendant
- a contact between the defendant or a movable object and the plaintiff
- apprehension by the plaintiff
- consent by the plaintiff

A consistent subset of these elements constitutes what I call an interpersonal transaction ("IPT"), an event that may be a candidate for an assault or battery. The program maintains a list of partially or wholly instantiated IPT’s. As each new sentence is received, the program searches the IPT list in reverse chronological order until an IPT is found to which the new sentence can be added. If no such IPT is found, the new sentence becomes the most recent IPT.

The procedure responsible for instantiating a new SSR is BLDSCH (for “BuiLD SCHEMA”), set forth along with the other instantiation procedures in Appendix C. BLDSCH extracts each IPT in turn from the IPT list and calls RELATE with the SSR and the IPT as arguments. RELATE attempts to find a schema that relates the SSR and the IPT. If RELATE succeeds, it returns a new IPT consisting of the combination of the SSR and the old IPT with pronominal references resolved, and BLDSCH substitutes the new IPT for the old in the IPT list.

The program contains six schemata falling into four categories. The first is the act of moving an object (a "move_obj") by one person followed by a contact between the object and another person. For example, given the SSR’s of the following sentences:
John threw a rock at Mary.
It hit her.

a call by RELATE to this schema would result in the following structure:

(THROW SEQ (HIT AGT (ROCK NBR SING)
  TNS PAST
  AE (MARY NBR SING))
  AGT (JOHN NBR SING)
  TNS PAST
  AE (ROCK NBR SING)
  OBJ (MARY NBR SING))

in which the two SSR's are related by the SEQ coherence relation.

A second class of schemata, which contains two members, involves the connection of an
action to its intent. The first schema involves an action sentence and an intention sentence, the AE of which is a verb infinitive. For example, the sentences:

John threw a rock at Mary.
He tried to hit her.

would result in the following structure:

(THROW PREC (TRY AGT (JOHN NBR SING)
  TNS PAST
  AE (HIT TNS INF AE (MARY NBR SING))
  AGT (JOHN NBR SING)
  TNS PAST
  AE (ROCK NBR SING)
  OBJ (MARY NBR SING))

The second intent schema applies when there are two action sentences, one of the action verbs is a generalization of the other, and the more general action sentence contains the arc/value pair “VMOD (INTENTIONALLY)”. For example, the sentences:

John threw a rock at Mary.
He acted intentionally.
result in the the structure:

(THROW PREC (ACT AGT (JOHN NBR SING)
TNS PAST
VMOD (INTENTIONALLY))
AGT (JOHN NBR SING)
TNS (PAST)
AE (ROCK NBR SING)
OBJ (MARY))

since "THROW" is an "ACT".

The third category of schemata involves apprehension. The first schema applies when a move_obj causes an apprehension, that is, when one sentence is a move_obj sentence, the second sentence is a "cause_app" sentence (cause_apps are words like scare, frighten, alarm, etc.), and the AE of the first sentence and the AGT of the second sentence are movable objects ("mob_objs") that can be matched. An example is:

John threw a rock at Mary.
It scared her.

which is represented:

(THROW SEQ (SCARE AGT (ROCK NBR SING)
TNS PAST
AE (MARY NBR SING))
AGT (JOHN NBR SING)
TNS PAST
AE (ROCK NBR SING)
OBJ (MARY NBR SING))

The second apprehension schema occurs where one sentence is a move_obj sentence the OBJ of which is a person, and the other sentence is a BE sentence, the AE of which is the same person, that contains the arc/value pair MOD_APP, where APP is an apprehension. An example is:

John threw a rock at Mary.
She was frightened.
represented as:

\[ \text{(THROW SEQ (BE AE (MARY NBR SING)) TNS PAST MOD (FRIGHTENED)) AGT (JOHN NBR SING) TNS PAST AE (ROCK NBR SING) OBJ (MARY NBR SING))} \]

The last category of schema involves consent. It relates an action sentence to a consent sentence if the AGT of the consent sentence matches the AE or OBJ of the action sentence. For example:

John kissed Mary.
She consented.

would be combined to form:

\[ \text{(KISS COORD (CONSENT AGT (MARY NBR SING) TNS PAST) AGT (JOHN NBR SING) TNS PAST AE (MARY NBR SING))} \]

Because the schemata are heavily constrained, consistent structures will result even if sentences from different IPT's are presented out of order or interwoven. This is useful for analyzing more complex situations like fights in which either party could be considered the plaintiff or the defendant. An example of a plausible situation involving interwoven sentences forming distinct IPT's is the following:

Alex swung a stick at Ellen.
Ellen threw a rock at Alex.
It hit him.
It frightened her.
He intended to hit her.
This sequence of sentences produces the following IPT list:

```
(THROW SEQ (HIT AGT (ROCK NBR SING)
  TNS PAST
  AE (ALEX NBR SING))
 AGT (ELLEN NBR SING)
 TNS PAST
 AE (ROCK DET A NBR SING)
 OBJ (ALEX NBR SING))
(SWING PREC (INTEND AGT (ALEX NBR SING)
  TNS PAST
  AE (HIT TNS INF
       AE (ELLEN NBR SING)))
 SEQ (FRIGHTEN AGT (STICK NBR SING)
  TNS PAST
  AE (ELLEN NBR SING))
 AGT (ALEX NBR SING)
 TNS PAST
 AE (STICK DET A NBR SING)
 OBJ (ELLEN NBR SING)))
```

The first IPT relates Ellen's throwing a rock at Alex to the rock hitting him. The second IPT relates Alex's swinging a stick at Ellen to his intent and to Ellen's apprehension.

4 Question Answering

The program provides several different procedures for answering questions from a list of instantiated IPT's. Most questions can be answered by simply matching the query against the IPT's. However, questions concerning legal liability require, in addition, that the matching IPT satisfy the elements of the tort being inquired of. The rules for question answering are set forth in Appendix D.
4.1 Simple yes/no questions

A sentence that begins with "did" or with a verb is recognized as a yes/no question. Procedure ANS_YN attempts to find an SSR that is a component of some IPT that satisfies the query. An SSR satisfies a query if the root of the SSR stands in the ISA relationship with the root of the query, i.e., if it is an instance of the root, and if for each arc/value pair in the query there is a corresponding arc/value pair in the SSR for which the arc is identical and the value is an instance of that in the query. The procedure SATISFY is responsible for determining whether a given SSR satisfies the query.

For example, the query "Did John contact Mary?" is parsed to

(CONTACT QOBJ DID
  AGT (JOHN NBR SING)
  TNS PAST
  AE (MARY NBR SING)).

The procedure ANS_W deletes the QOBJ DID pair and calls ANS_YN with the resulting SSR.

If the IPT includes:

(HIT AGT (JOHN NBR SING)
  TNS PAST
  AE (MARY NBR SING)),

the SSR for the sentence "John hit Mary", SATISFY will succeed because HIT is an instance of contact, and the remaining arc/value pairs in the query are present in the IPT.

If a query sentence begins with a verb, the verb is moved to its correct location, the sentence is parsed, and ANS_YN attempts to find an SSR in the IPT list that satisfies the resulting sentence.

4.2 Who questions

The program's grammar can accommodate who questions concerning the AGT of a sentence (e.g., "Who threw a rock at MARY?") or the object of the preposition "at" in a move_obj sentence (e.g., Who did Mary throw a rock at?). The resulting SSR contains the arc/value pair QOBJ AGT or QOBJ OBJ respectively. The procedure ANS_W attempts to find an
SSR in the IPT that matches the remaining portions of the query SSR and returns the value of the AGT or OBJ arc from the IPT SSR.

For example, "Who threw a rock at Mary?" is parsed to:

(THROW QOBJ AGT
  TNS PAST
  AE (ROCK NBR SING)
  OBJ (MARY NBR SING)).

If:

(THROW AGT (JOHN NBR SING)
  TNS PAST
  AE (ROCK NBR SING)
  OBJ (MARY NBR SING))

is in some IPT, "John" will be returned as the value of ANS_W since John is the AGT of the matching IPT sentence.

4.3 Yes/no questions regarding legal status

A special question answering procedure is invoked for verb-inversion yes/no questions whose SSR is of the following form:

(BE <defendant> LSTATUS (LIABLE *TO <plaintiff>
  *FOR <assault or battery>))

If ANS_YN receives a question of this form, it calls the procedure TORT, which takes as arguments the tort inquired of, the plaintiff, and the defendant. TORT extracts each IPT in turn from the IPT list and invokes SATTORT, which determines whether the given IPT satisfies each of the elements of the tort for the given plaintiff and defendant.

There are two SATTORT schemata for battery: one consisting of a contact between the plaintiff and the defendant, and one consisting of a move_obj by the plaintiff of some object and a contact by the object with the defendant. The sole assault schema consists of
a move_obj by the plaintiff of an object and apprehension of the object by the defendant. All three schemata require that intent be satisfied and that consent be absent.

For each element of the tort, SATTORT calls the procedure QSAT, which determines whether a particular element of a tort (intent, apprehension, or consent) is present. In general, SSR's of more than one form may satisfy a given element, so each call to QSAT will entail looking for any of several possibilities. If none is found, QSAT issues a question to the user. If the answer to the question is "Yes", the IPT is updated to reflect this new information.

For example, QSAT for intent checks first whether the IPT satisfies the SSR "(INTEND AGT (DF))". If that fails, QSAT tests whether the IPT satisfies the SSR "(_ACT VMOD (INTENTIONALLY))", where _ACT is the act giving rise to the possible tort, i.e., a contact or a move_obj. If that fails, QSAT asks the user whether the defendant acted intentionally, and, if the answer is yes, amends the IPT by adding: PREC (ACT AGT (DF NBR SING) VMOD (INTENTIONALLY) TNS PAST).

One complication to the approach of asking the user whether an element is satisfied is that there must be some minimum matching between SATTORT and an IPT before QSAT's can start being invoked. Otherwise, the system will always start with the first IPT, no matter how inappropriate, and begin asking the user whether each element in turn is satisfied. Ideally, the IPT's should be ranked by degree of match with the given tort before any questions are asked.

My program takes the simpler (but less satisfactory) approach of only trying IPT's with a move_obj or contact that matches the tort in question. Since an IPT is never considered a candidate unless it satisfies the move_obj or contact for the tort in question, no QSAT is necessary for the contact or move_obj elements of a tort.

4.4 How questions

If the answer to a yes/no question regarding legal status is yes, then the variable CNTXT is assigned the value of the IPT that satisfies the given tort. Questions of the form "How is element satisfied?", where element is contact, apprehension, or intent, are answered by the procedure ANS_W, which finds the SSR of CNTXT that satisfies that element, converts the SSR back into sentence form, and outputs the sentence as the answer.

The procedures used by ANS_W in the case of how questions include TRANS, which converts the element into its corresponding verb form; ROOTMEMBER and PRUNE,
which isolate a single SSR from CNTXT; and ROOTSAT, which determines whether a
given SSR satisfies the verb form of the given element.

5 The Consultation

The procedures that control the course of a consultation are CNSLT and PROC, set forth
in Appendix D. CNSLT issues the prompt "OK", calls GETSNT, a lisp procedure that
reads a line of text terminating in a back-slash or question mark, and calls PROC with the
line of text as its argument. If the text consists of the word "END", then PROC succeeds
and the consultation ends. If the text consists of the word "NEW", then the IPT list is
set to NIL and CNSLT is called again.

If the text is neither "END" nor "NEW", PROC calls the appropriate question an-
swering procedure if the sentence can be identified as a question. Otherwise, the sentence
is assumed to be an assertion and is parsed and passed to BLDSCH to be added to the
IPT list. In either case, the last clause of PROC is a new call to CNSLT.

The idea behind having PROC and CNSLT call one another is to create in effect a
listen-loop within HCPRVR. Of course, the goal stack gradually gets deeper during the
course of a consultation as CNSLT is repeatedly called, but this is unlikely to create a
problem in a consultation of normal proportions.

6 The Grammar

The program uses an ordinary top-down grammar, set forth in Appendix A. Its main
usual feature is that the semantic information from which SEF's are computed has been
transferred to the taxonomic tree. The motive for making this change is to localize all
semantic information in a single place.

A typical SEF is:

```lisp
(((SEF NRT VRT AGT)
  <
  (ISA NRT PERSON)
  (ISA VRT ACTION))
```
The program must search the taxonomic tree to determine whether NRT is a PERSON and VRT is an ACTION in order to conclude that the proper arc is AGT. It is not clear to me whether the advantages of centralizing semantic information outweigh the computational costs of constantly searching through the taxonomic tree.

7 The Taxonomic Tree

The program makes use of a taxonomic hierarchy of both objects and acts. The taxonomy is based directly upon the hierarchy set forth in [Meldman 75]. It is clearly too simple to sustain anything but superficial reasoning because there is no inheritance—and, in fact, no feature information other than position in the taxonomy—and there is only a single taxonomic axis. Development of deeper legal reasoning systems will depend heavily upon creating richer knowledge representation structures.

For purposes of experimentation, I added an associated action to the entry "GUN":

\[
\text{((GUN (ACTION FIRE BULLET)))}
\]

A rule was added to BLDSCH that associated a sentence having as its root a move_obj "_ACTION" and AE "_INSTR" with a sentence having as its root a contact "_CONTACT" and AGT "_OBJECT" if "_INSTR" has as its associated action "(ACTION _ACTION _OBJECT)". In the case of guns and bullets, this means that:

\[
\text{John fired a gun at Mary.}
\]
\[
\text{The bullet hit her.}
\]

will be linked into a single IPT. A complete ALRS would require a very rich supply of similar semantic information.

8 Discussion

My program uses essentially the same technique for discourse analysis as for legal analysis. Text is analyzed by instantiating a structured object representing an occurrence that might be an instance of assault or battery. Legal analysis is performed by testing each structured object to see whether all the requirements for, and none of the defenses to, a tort are present.
As compared with the system described in [Alterman 82], the language understanding portion of my system has comparatively little information associated with each lexical item and a great deal of domain-specific information embodied in the instantiation rules. As currently implemented, this approach seems appropriate only for a very application-specific system, since accommodating new object/event sequences requires new rules rather than simply new lexical entries.

On the other hand, it seems clear that humans use extensive domain-specific knowledge to guide their understanding of text. Simulating this human ability will require a more flexible system for embodying domain specific knowledge into the process of discourse analysis than does my system, which requires a fresh set of rules for each new domain.

Legal reasoning through frame instantiation was apparently first suggested in [Meldman 75]. He described how an ALRS for the law of assault and battery might be constructed using a frame-based language then under development at MIT. McCarty [McCarty 77] designed a frame instantiation system for a narrow area of the law of corporate taxation.

More recently, [McCarty & Sriharan 82], [Gardner 84], and others have argued that the techniques of frame instantiation and of rule-based systems are unlikely to achieve much depth because legal concepts are “open-textured”, that is, there is an inherent indeterminacy to the extension of such concepts. It is likely that ALRS’s will be unable to rival human legal reasoning capabilities until a better model of open-textured concepts is developed. However, frame instantiation and rule-based reasoning will probably play a role even in ALRS’s able to reason about such concepts.

9 Examples

Appendix E contains an interactive session involving nine case examples. Since the GET-SNT function issues "*" as a prompt, all lines beginning with "*" were typed by the user. For each example, the assertions made to the program are set forth below together with a short explanation of the program’s behavior.

1. Input:

   Aaron swung a knife at Alex.
   He tried to hit him.
   Alex was frightened by the weapon.
This example requires the program to connect a move_obj sentence, an intent sentence, and an apprehension sentence. Aaron is liable for assault because trying to hit Alex satisfies the intent requirement, Alex’s fear of the knife satisfies the apprehension requirement, and the user responds to the program’s query by saying Alex did not consent to Aaron’s actions. Aaron is not liable for battery because there was no contact.

2. Input:

Mildred shoved Ellen.
She meant to hit her.

This example requires that a contact sentence and an intent sentence be linked. Mildred is not liable for assault but is liable for battery because shoving Ellen satisfies the contact requirement of battery, that Mildred “meant to hit” Ellen satisfies the intent requirement, and the user states that there was no consent.

3. Input:

Mary tried to hit Aaron with a rock.
She threw it at him.
It scared him.

In this example, an intent sentence, a move_obj sentence, and an apprehension sentence must be connected. The three sentences satisfy the intent, act, and apprehension requirements of assault, so user’s statement that consent was absent establishes the assault. The program answers questions about how intent and apprehension are satisfied, and about who threw the rock and at whom the rock was thrown.

4. Input:
Alex swung a stick at Ellen.
Ellen threw a rock at Alex.
It hit him.
It frightened her.
He intended to hit her.

This is a case of two interwoven related event sequences. The program constructs a separate IPT for Alex’s actions towards Ellen and for Ellen’s actions toward Alex, and is able to answer questions appropriately about each.

5. Input:

John intentionally threw a rock at Mary.
It hit her.

Here, intent is satisfied by a VMOD (INTENTIONALLY) pair rather than through an intention sentence.

6. Input:

Mildred fired a gun at Aaron.
The bullet hit him.

This tests the ability of the program to connect firing a gun with a bullet hitting the OBJ of the firing.

7. Input:

Mildred tripped Jill.
She intentionally tripped her.

This input tests the ability of the program to connect parallel sentences, one of which specifies the intent of the AGT of the other.

8. Input:
John threw the book at Ellen.

This tests the ability of the program to ask the user about all the elements of a tort and incorporate the answers into the IPT.

9. Input:

John kissed Mary.
She consented.

This example tests the ability of the program to link a contact sentence with a consent sentence. Battery was not committed by John because of Mary’s consent. The program unnecessarily asks whether John acted intentionally because it tests for the elements of a tort before considering defenses.
References


10 Appendices

A  Grammar for Prototype Legal Consultant

(VARIABLES PA X Y U V V1 V2 W W1 N2 VRT VRT1 X1 X2 X3 Y1 Y2 Y3
     NRT NRT1 R R1 R2 ARC ARC1 CNTX DF PL)

(SETOGRAMMAR '(
    (; SENTENCE RULES)
    (; FIRST, THE SPECIAL CASE OF QUESTION WORDS)
    (S (WHO.X) (VRT QOBJ AGT.V))
        <
        (VP X VRT NIL (VRT.V)))

    ((S (DID.X) (VRT QOBJ DID ARC W.V1))
        <
        (NP X NRT R W)
        (VP R VRT NIL (VRT.V1))
        (SEP NRT VRT ARC))

    ((S (WHO DID.X) (VRT QOBJ AE.Y))
        <
        (S X (VRT.Y))
        (UNLESS* (ARCMEMBER AE Y)))

    ((S (WHO DID.X) (VRT QOBJ ARC1 ARC W.V1))
        <
        (NP X NRT R W)
        (VP R VRT (R1) (VRT.V1))
        (PREP R1 U ARC1)
        (SEP NRT VRT ARC))

    ((S (HOW X.X1) (BE QOBJ HOW AE W ARC V1 ARC1 (Y1.W1)))
        <
        (VERB X (BE ARC V1))
        (NP X1 NRT (R.R1) W)
        (ADJ R (Y1) ARC1)
        (NCOMP R1 U NIL W1))

    (; PREDICATE ADJECTIVE SENTENCE RULE )


19
((S X (BE AE W ARC V1 ARC1 (Y1.W1)))
  <
  (NP X NRT (X1.R1) W)
  (VERB X1 (BE ARC V1))
  (ADJ R (Y1) ARC1)
  (ADVPH R1 NIL W1))

(; OTHER SENTENCES)
((S X (VRT ARC W.V1))
  <
  (NP X NRT R W)
  (VP R VRT NIL (VRT.V1))
  (SEP NRT VRT ARC))

(; VERBAL PHRASES EG THAT MODIFY PRED ADJ'S)
((ADVPH NIL NIL NIL))
((ADVPH (X.Y) R1 (ARC W.W1))
  <
  (PREP X ARC U)
  (NP Y NRT R W)
  (ADVPH R R1 W1))

(; NOUN PHRASE RULES )
((NP (X.Y) NRT R (NRT DET X.Y))
  <
  (DET X)
  (NP1 Y NRT R (NRT.Y)))

((NP X NRT R V)
  <
  (NP1 X NRT R V))

(; NP1 RULES )
((NP1 (X.Y) NRT R (NRT ARC V W *AND W1))
  <
  (NOUN X (NRT ARC V))
  (NCOMP Y NRT (AND.R1) W)
  (NP R1 NRT1 R W1))

((NP1 (X.Y) NRT R (U ARC X1.V))
  <
  (ADJ X X1 FA)
((NP (X.Y) VRT R (VRT TNS INF.W)))
<
(VP X VRT R (X1.Y1))
(SEF PERSON VRT AGT))

((NP (X.Y) VRT R (VRT TNS PRPRT.W)))
<
(VP X VRT R (X1.Y1))
(SEF PERSON VRT AGT))

((NP (TO X.Y) NRT R (AE* (VRT TNS INF).W)))
<
(VP X VRT R (X1.Y1))
(SEF PERSON VRT AGT))
<
(VERB X (VRT U X3))
(VCOMPS Y VRT R W))

(((VP (X.Y) VRT R (VRT U X3 V V1 *AND W)))
<
(VERB X (VRT U X3))
(VCOMPS Y VRT (AND.R1) (V V1))
(VCOMPS R1 VRT1 R W))

(((VP (X.Y) VRT R (VRT VMOD X1.Y1)))
<
(ADV X X1 ARC)
(VP Y VRT R (VRT.Y1)))

(; VCOMPS RULES ;)

(((VCOMPS NIL VRT NIL NIL))
(((VCOMPS (TO X.Y) VRT R (AE (_VRT TWS INF.W))))
<
(VERB X (_VRT TWS PRES))
(VCOMPS Y _VRT R W))

(((VCOMPS X VRT R (ARC V)))
<
(WP X NRT R V)
(SEF VRT NRT ARC))

(((VCOMPS X VRT R (AE V Y V1)))
<
(WP X NRT (X1.R1) V)
(PREP X1 Y Y1)
(SEF VRT NRT AE)
(WP R1 NRT1 R V1))

(((VCOMPS (X1.Y) VRT R (ARC (V PRP X1.1)))
<
(PREP X1 ARC1 ARC)
(WP Y NRT R (V.V1)))

(((VCOMPS (X1.Y) VRT R (ARC (V ARC1 W.V1)))
<
(ADV X1 X2 ARC)
(NP Y NRT R1 W)
(VP R1 VRT R (V.V1))
(SEF NRT VRT ARC1))

(; VERB MODIFYING ADVERBS)
((VCOMP X VRT R (VMOD Y2.V)) <
  (SETV* (Y.Y1) (REVERSE 'X))
  (ADV Y Y2 ARC)
  (SETV* X1 (REVERSE 'Y1))
  (VCOMP X1 VRT R V))

(; NOUNS)

((NOUN ASSAULT (ASSAULT NBR SING)))
((NOUN BATTERY (BATTERY NBR SING)))
((NOUN HE (HE NBR SING)))
((NOUN HIM (HIM NBR SING)))
((NOUN SHE (SHE NBR SING)))
((NOUN IT (IT NBR SING)))
((NOUN HER (HER NBR SING)))
((NOUN JOHN (JOHN NBR SING)))
((NOUN MARY (MARY NBR SING)))
((NOUN AARON (AARON NBR SING)))
((NOUN JILL (JILL NBR SING)))
((NOUN ALEX (ALEX NBR SING)))
((NOUN ELLEN (ELLEN NBR SING)))
((NOUN MILDRED (MILDRED NBR SING)))
((NOUN STICK (STICK NBR SING)))
((NOUN GUN (GUN NBR SING)))
((NOUN BULLET (BULLET NBR SING)))
((NOUN KNIFE (KNIFE NBR SING)))
((NOUN HAT (HAT NBR SING)))
((NOUN ROCK (ROCK NBR SING)))
((NOUN CAR (CAR NBR SING)))
((NOUN BOOK (BOOK NBR SING)))
((NOUN ARM (ARM NBR SING)))
((NOUN INTENT (INTENT NBR SING)))
((NOUN APPREHENSION (APPREHENSION NBR SING)))
((NOUN CONTACT (CONTACT NBR SING)))
((NOUN WEAPON (WEAPON NBR SING)))

(; VERBS)
((VERB STRIKE (STRIKE TWS PRES)))
((VERB STRUCK (STRIKE TWS PAST)))
((VERB CONTACT (CONTACT TWS PRES)))
((VERB CONTACTED (CONTACT TWS PAST)))
((VERB ASSAULT (ASSAULT TWS PRES)))
((VERB ASSAULTED (ASSAULT TWS PAST)))
((VERB HIT (HIT TWS PAST)))
((VERB HIT (HIT TWS PRES)))
((VERB MISSED (MISS TWS PAST)))
((VERB MISS (MISS TWS PRES)))
((VERB FIRED (FIRE TWS PRES)))
((VERB FIRED (FIRE TWS PAST)))
((VERB SHOT (SHOOT TWS PAST)))
((VERB TRIPPED (TRIP TWS PAST)))
((VERB THREW (THROW TWS PAST)))
((VERB THROW (THROW TWS PRES)))
((VERB SHOVED (SHOVE TWS PAST)))
((VERB SHOVED (SHOVE TWS PRES)))
((VERB KISSED (KISS TWS PAST)))
((VERB KISS (KISS TWS PRES)))
((VERB TOLD (TELL TWS PAST)))
((VERB HAD (HAVE TWS PAST)))
((VERB IS (BE TWS PRES)))
((VERB WAS (BE TWS PAST)))
((VERB INTEND (INTEND TWS PRES)))
((VERB INTENDED (INTEND TWS PAST)))
((VERB WANT (WANT TWS PRES)))
((VERB MEAN (MEAN TWS PRES)))
((VERB MEANT (MEAN TWS PAST)))
((VERB WANTED (WANT TWS PAST)))
((VERB CONSENTED (CONSENT TWS PAST)))
((VERB CONSENT (CONSENT TWS PRES)))
((VERB AGREE (AGREE TWS PRES)))
((VERB AGREED (AGREE TWS PAST)))
((VERB TRIED (TRY TWS PAST)))
((VERB APPREHEND (APPREHEND TWS PRES)))
((VERB SCARED (SCARE TWS PAST)))
((VERB FRIGHTENED (FRIGHTEN TWS PAST)))
((VERB SCARE (SCARE TWS PRES)))
((VERB FRIGHTEN (FRIGHTEN TWS PRES)))
((VERB ACT (ACT TWS PRES)))
((VERB ACTED (ACT TWS PAST)))
((VERB SWING (SWING TWS PRES)))
((VERB SWUNG (SWING TWS PAST)))
(; VERB AUXILIARIES)

((AUX HAD))
((AUX DID))

(; SEF's)
((SEF NRT IS AE))

((SEF NRT VRT AGT) < 
 (ISA NRT PERSON) 
 (ISA VRT ACTION))

((SEF NRT VRT AGT) < 
 (ISA NRT MOBL_OBJ) 
 (ISA VRT CONTACT))

((SEF NRT VRT AGT) < 
 (ISA NRT MOBL_OBJ) 
 (ISA VRT CAUSE_APP))

((SEF VRT NRT AE) < 
 (ISA VRT CAUSE_APP) 
 (ISA NRT PERSON))

((SEF NRT VRT AGT) < 
 (ISA VRT INTEND) 
 (ISA NRT PERSON))

((SEF VRT NRT AE) < 
 (ISA VRT ACTION))

((SEF QUAL X MOD))

((SEF TIME THING MOD))

(; ADJECTIVES)

((ADJ SATISFIED (SATISFIED) MOD))
((ADJ SCARED (SCARED) MOD))
((ADJ FRIGHTENED (FRIGHTENED) MOD))
((ADJ LIABLE (LIABLE) LSTATUS))
(; PREPOSITIONS)

  ((PREP OF SUBJ *OF))
  ((PREP ABOUT SUBJ *ABOUT))
  ((PREP IN LOC LOC))
  ((PREP WITH *WITH INST))
  ((PREP TO *TO OBJ))
  ((PREP FOR *FOR *FOR))
  ((PREP AT OBJ OBJ))
  ((PREP BY INSTR INSTR))

(; ADVERBS)

  ((ADV INTENTIONALLY (INTENTIONALLY) VMOD))
  ((ADV (HOW) *HOW))

(; DETERMINERS)

  ((DET A))
  ((DET THE))

))

(AXIOMS GRAMMAR)

(SETQ LIMIT 5000)
B  Taxonomic Tree

(SETQ TAXON '(
   (; Identity)
   ((ISA _A _A))

   (; Transitivity rules )
   ((ISA _A _B)
    <
    (ISSA _A _B))

   ((ISA _A _B)
    <
    (ISSA _A _C)
    (ISA _C _B))

   (; allow objects to be indexed through name rather than ISSA )
   ((ISSA _A _B)
    <
    (_A ISSA _B))

   (; Object hierarchy )
   ((MOBL_OBJ ISSA OBJECT))

   (; MOVABLE OBJECTS )
   ((TOOL ISSA MOBL_OBJ))
   ((WEAPON ISSA MOBL_OBJ))
   ((CLOTHING_ART ISSA MOBL_OBJ))
   ((STICK ISSA MOBL_OBJ))
   ((ROCK ISSA MOBL_OBJ))

   (; VEHICLES )
   ((CAR ISSA VEHICLE ))

   (; TOOLS )
   )
   )

27
((BOOK ISSA TOOL))

(; WEAPONS   )

((GUN ISSA WEAPON))
((GUN (ACTION FIRE BULLET)))
((ROCK ISSA WEAPON))
((KNIFE ISSA WEAPON))
((STICK ISSA WEAPON))
((BULLET ISSA WEAPON))

(; BODY PARTS)

((ARM ISSA BODY_PART))
((LEG ISSA BODY_PART))

(; MENTAL STATES)

((INTEND ISSA MENTAL_STATE))
((APPROACH ISSA MENTAL_STATE))
((CONSENT ISSA MENTAL_STATE))

(; CONSENTS)
((AGREE ISSA CONSENT))

(; INTENDS)
((TRY ISSA INTEND))
((ATTEMPT ISSA INTEND))
((WANT ISSA INTEND))
((MEAN ISSA INTEND))

(; APPREHENDS)
((FRIGHTENED ISSA APPREHEND))
((SCARED ISSA APPREHEND))
((CAUSE_APP ISSA APPREHEND))
((INTEND ISSA ACTION))
((ACT ISSA ACTION))

(; CAUSE_APP, IE, CAUSE APPREHENSION)
((FRIGHTEN ISSA CAUSE_APP))
((SCARE ISSA CAUSE_APP))

(; ACTS )
((CONTACT ISSA ACT))
((MOVE_OBJC ISSA ACT))
((MOVE_SELF ISSA ACT))
((CONSENT ISSA ACT))
((CAUSE_APP ISSA ACT))
(; CONTACTS )

((SHOVE ISSA CONTACT))
((TOUCH ISSA CONTACT))
((GRAB ISSA CONTACT))
((STRIKE ISSA CONTACT))
((MISS ISSA CONTACT))
((SHOOT ISSA CONTACT))
((TRIP ISSA CONTACT))
((MISS ISSA CONTACT))

(; MOVE_OBJC )

((CARRY ISSA MOVE_OBJC))
((STEAL ISSA MOVE_OBJC))
((THROW ISSA MOVE_OBJC))
((SWING ISSA MOVE_OBJC))
((FIRE ISSA MOVE_OBJC))
((AIM ISSA MOVE_OBJC))

(; COMMUNICATIONS )

((TELL ISSA COMMUN))
((SHOW ISSA COMMUN))

(; STRIKES )

((PUNCH ISSA STIKE))
((HIT ISSA STRIKE))

(; PERSONS )
((MAN ISSA PERSON))
((WOMAN ISSA PERSON))
((SHE ISSA PERSON))
((HER ISSA PERSON))
((HE ISSA PERSON))
((HIM ISSA PERSON))
((JOHN ISSA MAN))

29
((MARY ISSA WOMAN))
((AARON ISSA MAN))
((ALEX ISSA MAN))
((ELLEN ISSA WOMAN))
((JILL ISSA WOMAN))
((MILDRED ISSA WOMAN))

(; SPECIAL CASES)
((IT ISSA X)
<
(UNLESS* (X ISSA PERSON)))
))

(AXIOMS TAXON)
C Rules for Instantiation of Interpersonal Transaction Frames

(VARIABLES ARC X X1 X2 Y Y1 Y2 U W Z ARC)

(SETPQ RULES ')

(;; BUILD A SCHEMA FROM THE GIVEN SSR)

(;; IF IPT, THE SCHEMA LIST IS EMPTY, MAKE SSR SOLE MEMBER)
((BLDSCH _SSR)
 <
 (SETV* NIL IPT)
 (SETV* _DUMMY (SETQ IPT (LIST '_SSR))))

(;; IF IPT IS NON-EMPTY, ATTEMPT TO ADD THE SSR TO AN EXISTING
SCHEMA)
((BLDSCH _SSR)
 <
 (SETV* _IPT IPT)
 (MEMBER _T _IPT)
 (RELATE _SSR _T _NEW)
 (REPLACEA (_NEW _T) _IPT2 _IPT)
 (SETV* _DUMMY (SETQ IPT '_IPT2)))

(;; IF SSR CANNOT BE RELATED TO EXISTING SCHEMA, LET IT BE A NEW ONE)
((BLDSCH _SSR)
 <
 (SETV* _DUMMY (SETQ IPT (CONS '_SSR IPT))))

(;; TWO EVENT/STATES ARE RELATED IF THEY ARE CONNECTED BY A SCHEMA)

((RELATE X (Y.Y1) (Y ARC X1.Y2))
 <
 (SCHEMA ARC (Y.Y1) X _SUB2 _SUB1)
 (REPLACE _SUB1 X X1)
 (REPLACE _SUB2 Y1 Y2))

((RELATE (X.X1) Y (X ARC Y1.X2))
 <
 (SCHEMA ARC (X.X1) Y _SUB1 _SUB2)
 (REPLACE _SUB1 X1 X2)
(REPLACE _SUB2 Y Y1))

( ; SCHEMAS)

( ; MOVE_OBJ RESULTING IN A CONTACT)
((SCHEMA SEQ (X. X1) (Y. Y1) _SUB1 _SUB2)
 <
 (ISA X MOVE_OBJ)
 (ISA Y CONTACT)
 (ARCMEMBER AE _INSTR1 X1)
 (ARCMEMBER AGT _INSTR2 Y1)
 (ISA _INSTR1 MOBL_OBJ)
 (MATCH_INSTR X _INSTR1 _INSTR2 _S11 _S12)
 (ARCMEMBER OBJ Z X1)
 (ARCMEMBER AE W Y1)
 (CORRESP Z W _S11 _S12 _SUB1 _SUB2))

( ; INTENT OF AN ACTION)
((SCHEMA PREC (X. X1) (Y. Y1) _SUB1 _SUB2)
 <
 (ISA X ACTION)
 (ISA Y INTEND)
 (ARCMEMBER AGT Z X1)
 (ARCMEMBER AGT W Y1)
 (MATCH Z W _S11 _S12)
 (MEMBER _ARC (OBJ AE))
 (ARCMEMBER _ARC V X1)
 (ISA V PERSON)
 (PAIRMEMBER AE (_A THIS INF. _Z) Y1)
 (MEMBER _ARC1 (AE OBJ))
 (ARCMEMBER _ARC1 V1 _Z)
 (CORRESP V V1 _S11 _S12 _SUB1 _SUB2))

((SCHEMA PREC (X. X1) (Y. Y1) _SUB1 _SUB2)
 <
 (ISA X ACTION)
 (ISA Y ACTION)
 (ISA X Y)
 (ARCMEMBER AGT Z X1)
 (ARCMEMBER AGT W Y1)
 (MATCH Z W _SUB1 _SUB2)
 (ARCMEMBER VMOD INTENTIONALLY Y1))

( ; MOVE_OBJ THAT CAUSES AN APPREHENSION)
((SCHEMA SEQ (X X1) (Y Y1) _SUB1 _SUB2)
 <
 (ISA X MOVE_OBJ)
 (ISA Y CAUSE_APP)
 (ARCMEMBER AE _INSTR1 X1)
 (ARCMEMBER AGT _INSTR2 Y1)
 (ISA _INSTR1 MOBL_OBJ)
 (MATCH_INSTR X _INSTR1 _INSTR2 _S11 _S12)
 (ARCMEMBER OBJ Z X1)
 (ARCMEMBER AE W Y1)
 (CORRESP Z W _S11 _S12 _SUB1 _SUB2))

(; MOVE_OBJ WITH OBJ PL FOLLOWED BY AN APPREHENSION BY PL)
((SCHEMA SEQ (X X1) (BE Y1) _SUB1 _SUB2)
 <
 (ISA X MOVE_OBJ)
 (ARCMEMBER OBJ Z X1)
 (ARCMEMBER AE W Y1)
 (MATCH Z W _SUB1 _SUB2)
 (ARCMEMBER MOD _A Y1)
 (ISA _A APPREHEND))

(; CONSENT TO AN ACTION)
(; CASE 1 -- CONSENT BY AE OR OBJ OF ACTION)
((SCHEMA COORD (X X1) (Y Y1) _SUB1 _SUB2)
 <
 (ISA X ACTION)
 (ISA Y CONSENT)
 (ARCMEMBER AGT Z Y1)
 (MEMBER _ARC (AE OBJ))
 (ARCMEMBER _ARC W X1)
 (MATCH W Z _SUB1 _SUB2))

(; CASE 2 -- CONSENT BY AE OF RELATED ACTION)
((SCHEMA COORD (X X1) (Y Y1) _SUB1 _SUB2)
 <
 (ISA X ACTION)
 (ISA Y CONSENT)
 (ARCMEMBER AGT Z Y1)
 (ARCMEMBER SEQ _R X1)
 (MEMBER _ARC (AE OBJ))
 (ARCMEMBER _ARC W _R)
 (MATCH Z W _SUB1 _SUB2))

((CORRESP Z W _S11 _S12 _SUB1 _SUB2)

33
<
(MATCH Z W _S21 _S22)
(APPEND _S11 _S21 _SUB1)
(APPEND _S12 _S22 _SUB2))

(; MATCH FUNCTION)

((MATCH X (Y POSS Y1.Y2) _S1 _S2)
 <
 (MATCH X Y1 _S1 _S2))
((MATCH (Y POSS Y1.Y2) X _S1 _S2)
 <
 (MATCH Y1 X))
((MATCH HER X (X HER) NIL)
 <
 (ISA X WOMAN))
((MATCH X HER NIL (X HER) )
 <
 (ISA X WOMAN))
((MATCH SHE X (X SHE) NIL)
 <
 (ISA X WOMAN))
((MATCH X SHE NIL (X SHE))
 <
 (ISA X WOMAN))
((MATCH BE X (X BE) NIL)
 <
 (ISA S MAN))
((MATCH X BE NIL (X BE))
 <
 (ISA X MAN))
((MATCH HIM X (X HIM) NIL)
 <
 (ISA X MAN))
((MATCH X HIM NIL (X HIM))
 <
 (ISA X MAN))
((MATCH X IT NIL (X IT))
 <
 (UNLESS* (ISA X PERSON)))
((MATCH IT X (X IT) NIL)
 <
 (UNLESS* (ISA X PERSON)))
((MATCH X X NIL NIL))  
((MATCH X Y NIL NIL)  
  (ISA X Y))  
((MATCH X Y NIL NIL)  
  (ISA Y X))

(; SPECIAL MATCH FUNCTION FOR INSTRUMENTS, EG, GUNS ETC)  
((MATCH_INSTR _A X Y _SUB1 _SUB2)  
  (MATCH X Y _SUB1 _SUB2))

((MATCH_INSTR _A X Y NIL NIL)  
  (X (ACTION _A Y)))

(; LIST FUNCTIONS)  
((MEMBER X (X.Y)))  
((MEMBER X (Y.Z))  
  (MEMBER X Z))

((REPLACEA (X X) Y Y))  
((REPLACEA (X Y) (Y.X1) (X.X1)))  
((REPLACEA (X Y) (W.W1) (W.Z))  
  (REPLACEA (X Y) W1 Z))

((REPLACE () X X))  
((REPLACE (X Y) ((Y.Y1).Z) ((X.Y1).Z)))  
((REPLACE X (Y) (Y1))  
  (REPLACE X Y Y1))

((REPLACE (X Y) (W.Z) (W.U))  
  (REPLACE (X Y) Z U))  
((REPLACE (X Y.Y1) U W)  
  (REPLACE (X Y) U Z)  
  (REPLACE Y1 Z W))

((REPLACE X ((Y.Y2).Z) ((Y1.Y2).Z))
<
(REPLACE X Y Y1))
(((REPLACE X (Y Y1 Y2) (Y Z Y2))
<
(REPLACE X Y1 Z))

(((ARC MEMBER X Y (X Y1 Z)))
(((ARC MEMBER X Y (U W Z))
<
(RE ARC MEMBER X Y Z))
(((APPEND () X X))
(((APPEND (X Y) W (X Z)))
<
(APPEND Y W Z))

)))
(AXIOMS RULES)
(SETQ IPT NIL)
D Rules for Question Answering

(SETQ RULES '(

(; IN A CONSULTATION, SENTENCES ARE INPUT SEPARATELY; DECLARATORY
SENNENCES ARE ASSERTED; QUESTIONS ARE ANSWERED; END TERMINATES
CONSULTATION)
((CNSLT)
 <
 (RPRINT OK)
 (RTERPRI)(RTERPRI)
 (SETV* X (GETSMT))
 (PROC X))

(; TERMINATE INPUT)
((PROC (END)))

(; CREATE NEW IPT)
((PROC (NEW))
 <
 (SETV* _DUMMY1 (SETQ IPT NIL))
 (CNSLT))

(; IF SENTENCE BEGINS WITH QUERY WORD, IT IS A NON-YES/NO QUESTION)
((PROC (X.X1))
 <
 (MEMBER X (DID HOW WHO))
 (S (X.X1) Y)
 (ANS_W Y)
 (CNSLT))

(; IF S BEGINS NEITHER WITH QTERM NOR VERB, IT IS DECLARATIVE)
((PROC X)
 <
 (S X Y)
 (BLDSCH Y)
 (CNSLT))

(; RULES FOR ANSWERING LEGAL QUESTIONS)
((ANS_YN (BE.X))
 <
(PAIREDMEMBER LSTATUS (LIABLE X1) X)
(PAIREXIST *FOR (_TORT X2) X1)
(ARCHMEMBER AE DF X)
(ARCHMEMBER TO PL X1)
(TORT _TORT PL DF))

((TORT _TORT PL DF)
 <
 (SETV* _IPT IPT)
 (MEMBER _T _IPT)
 (SATTORT _TORT PL DF _T _T1)
 (REPLACEA (_T1 _T) _IPT _IPT2)
 (SETV* _DUMMY1 (SETQ IPT '_IPT2))
 (SETV* _DUMMY2 (SETQ CNTXT '_T1))
 (PUTSVT* (YES, DF IS LIABLE TO PL FOR _TORT)))
((TORT _TORT PL DF)
 <
 (PUTSVT* (NO, DF IS NOT LIABLE FOR _TORT)))

(; HOW QUESTIONS IE HOW LEGAL STATUS IS SATISFIED)
((ANS_W (BE QOBJ HOW X))
 <
 (ARCHMEMBER AE _LSTATE X)
 (MEMBER _LSTATE (CONTACT INTENT APPREHENSION))
 (ARCHMEMBER MOD SATISFIED X)
 (TRANS _LSTATE _VFORM)
 (SETV* _CNTAX CNTXT)
 (ROOTMEMBER (Y1 Y1) _CNTAX)
 (PRUNE Y1 Y2)
 (ROOTSAT _VFORM (Y2))
 (OUTPUT (Y2)))
((ROOTSAT Y (X1 X1))
 <
 (ISA X Y))
((ROOTSAT INTEND (X1 X1))
 <
 (ARCHMEMBER VMOD INTENTIONAL X1))
((ROOTSAT APPREHEND (BE X))
 <
 (ARCHMEMBER MOD X1 X)
 (ISA X1 APPREHEND))
((OUTPUT X)
 <
 (S Y X))
(PUTSYM Y))
((OUTPUT X)
 <
 (PUTSYM X))
((TRANS CONTACT CONTACT))
((TRANS INTENT INTEND))
((TRANS APPREHENSION APPREHEND))
((PRUNE NIL NIL))
((PRUNE (ARC X Y) Z)
 <
 (MEMBER ARC (PREC SEQ COORD))
 (PRUNE Y Z))
((PRUNE (ARC X Y) (ARC X Z))
 <
 (PRUNE Y Z))

(; YES/NO QUESTIONS)
((PROC (X Y W))
 <
 (S (Y X W) X1)
 (ANS_YN X1)
 (CNSLT))

(; FIND A SINGLE SSR IN STRUCTURE WHICH SATISFIES QUERY)
((ANS_YN X)
 <
 (SETV* _IPT IPT)
 (MEMBER _T _IPT)
 (ROOTMEMBER Y _T)
 (SATISFY Y X)
 (RPRINT YES))

((ANS_YN X)
 <
 (RPRINT NO))

(; NOW-YES/NO QUESTIONS)
(; DID QUESTIONS)
((ANS_W (X QOBJ DID. X1))
 <
 (ANS_YN (X X1)))

(; WHO QUESTIONS)
((ANS_W (X QOBJ ARC. X1))}
(MEMBER ABC (AGT AE OBJ))
(SETV* _IPT IPT)
(MEMBER _T _IPT)
(ROUTMEMBER Y _T)
(SATISFY Y (X ARC (W.W1).X1))
(PUTSN* W))
((ANS_W X)
<
(PUTSN* (I DON'T KNOW)))
((ROUTMEMBER X X))
(((ROUTMEMBER X (Y.Y1))
<
(PAIRMEMBER W X Y1)
(MEMBER W (SEQ PREC COORD)))

(;; RESEMBLES ARCMEMBER, BUT RETURNS ENTIRE LIST INSTEAD OF JUST HEAD)
((PAIRMEMBER X Y (X.Y.Z)))
((PAIRMEMBER X Y (U W.Z))
<
(PAIRMEMBER X Y Z))
((SATISFY (X.X1) (Y.Y1))
<
(ISA X Y)
(SATPAIR X1 Y1))
((SATPAIR X NIL))
((SATPAIR X (Y Y1.Z))
<
(MEMBER Y (AUX TNS ))
(SATPAIR X Z))
((SATPAIR X (Y (Y1.Y2).Z))
<
(ARCMEMBER Y X1 X)
(ISA X1 Y1)
(SATPAIR X Z))
(;; UNPARSABLE SENTENCE ERROR)
((PROC X)
<
(PUTSN* (UNABLE TO PARSE INPUT -- TRY AGAIN))
(CNSLT)))

(;; TWO SCHEMAS FOR BATTERY)
((SATTTORT BATTERY PL DF _T _T1)
<
(SATISFY X (CONTACT AGT (DF) AE (PL)))
(ISA DF PERSON)
(QSAT INTEND CONTACT PL DF _T _T1) !
(UNLESS* (QSAT CONSENT PL DF _T1 _T2))
((SATORT BATTERY PL DF _T _T2)
<
(STOREMembre X _T)
(QSAT INTEND CONTACT PL DF _T _T1) !
(QSAT CONTACT _INSTR PL _T _T1) !
(QSAT INTEND MOVE_OBJ PL DF _T1 _T2) !
(UNLESS* (QSAT CONSENT PL DF _T2 _T3)))

(;; SCHEMA FOR ASSAULT)
((SATORT ASSAULT PL DF _T _T2)
<
(STOREMembre X _T)
(QSAT INTEND MOVE_OBJ AGT (DF) AE (_INSTR))) !
(QSAT APPREHEND _INSTR PL _T _T1)
(QSAT INTEND MOVE_OBJ PL DF _T1 _T2) !
(UNLESS* (QSAT CONSENT PL DF _T2 _T3)))

(;; ATTEMPT TO SATISFY ELEMENTS OF TORT FROM IPT OR BY ASKING)
((QSAT INTEND _ACT PL DF X X)
<
(STOREMembre X1 X)
(QSAT INTEND AGT (DF)))
((QSAT INTEND _ACT PL DF X X)
<
(STOREMembre X1 X)
(QSAT INTEND AGT (DF))
((QSAT INTEND _ACT PL DF (X.X1) (X PRED (ACT AGT (DF NBR SING)
VMOD (INTENTIONALLY) TMS PAST).X1))
<
(PUTMETA (DID DF ACT INTENTIONALLY TOWARD PL ?))
(INTERPRI)
(SETV* _ANS (GETSNT)) !
(EQ* _ANS (YES)))

((QSAT CONSENT PL DF X X)
<
(STOREMembre X1 X)
(QSAT INTEND AGT (PL)))
((QSAT CONSENT PL DF (X.X1)
(x coord (consent agt (pl nbr sing) tns past).x1))

<
(putsnt* (did pl consent to the actions of df ?))
(rterpri)
(setv* _ans (getsnt)) !
(eq* _ans (yes))

((qsat contact _instr pl _t _t)
 <
 (rootmember x _t)
 (satisfy x (contact agt (_instr) ae (pl))))
 ((qsat contact _instr1 pl _t _t)
 <
 (rootmember (x.x1) _t)
 (_instr1 (action x _instr2))
 (rootmember y _t)
 (satisfy y (contact agt (_instr2) ae (pl))))
 ((qsat contact _instr pl (x.x1)
   (x seq (contact agt (_instr det the nbr sing)
     tns past ae (pl nbr sing)).x1))

<
(putsnt* (did the _instr contact pl ?))
(rterpri)
(setv* _ans (getsnt)) !
(eq* _ans (yes))

((qsat apprehend _instr pl x x)
 <
 (rootmember x1 x)
 (satisfy x1 (be ae (pl nbr sing) tns past
     mod (apprehend instr (_instr)))))

((qsat apprehend _instr pl x x)
 <
 (rootmember x1 x)
 (satisfy x1 (apprehend agt (_instr) ae (pl))))
 ((qsat apprehend _instr pl (x.x1)
   (x seq (frighten agt (_instr det the nbr sing)
     tns past ae (pl nbr sing)).x1))

<
(putsnt* (did the _instr cause apprehension in pl ?))
(rterpri)
(setv* _ans (getsnt)) !
(eq* _ans (yes))
(PUTSN'T* S)
<
(SETV* _DUMMY (PUTSN'T 'S)))
})

(AXIOMS RULES)
(SETQ CNTXT NIL)
(SETQ LIMIT 5000)

(; RETURNS SENTENCE INPUT FROM KEYBOARD AS LIST)
(DE GETSN'T ()
 (PROG (A B)
     LOOP (SETQ A (READ))
 (COND ((OR (EQUAL A '"')(EQUAL A '?')) (RETURN B)))
 (SETQ B (REVERSE (CONS A (REVERSE B))))
 (GO LOOP)))

(; PRINTS ATOMS IN LIST)
(DE PUTSN'T (S)
 (PROG ()
     (SETQ L S)
     (TERPRI NIL)
     LOOP (COND ((NULL L) (RETURN NIL))
     ((ATOM L) (PRIN L) (RETURN NIL))
     (PRIN (CAR L))
     (PRINC '/ ')
     (SETQ L (CDR L))
     (GO LOOP)))

43
E  Sample Consultation

[PHOTO: Recording initiated Sat 21-Feb-87 4:50PM]

[Link from CS.BRANTING, TTY52]

Tops-20 Command processor 5.1(223700)
@ELISP
Elisp, 10 14 85

The variable MSG shows how to begin and MSGi shows how to save your files.

*(LOAD-AX GRAMMAR)(LOAD-AX TAXON)

Axioms loaded
GRAMMAR

Axioms loaded
TAXON

*(LOAD-AX INSTANT)(LOAD-AX ANSWER)

Axioms loaded
INSTANT

Axioms loaded
ANSWER

*(?CNSLT))

OK

*AARON SWUNG A KNIFE AT ALEX \n
OK

*HE TRIED TO HIT HIM \n
OK
*ALEX WAS FRIGHTENED BY THE WEAPON \ 
OK

*IS AARON LIABLE TO ALEX FOR ASSAULT ?
DID ALEX CONSENT TO THE ACTIONS OF AARON ? *NO \ 
YES, AARON IS LIABLE TO ALEX FOR ASSAULT OK

*HOW IS INTENT SATISFIED ?
AARON TRIED TO HIT ALEX OK

*IS AARON LIABLE TO ALEX FOR BATTERY ?
DID THE KNIFE CONTACT ALEX ? *NO \ 
NO, AARON IS NOT LIABLE FOR BATTERY OK

*NEW \ 
OK

*MILDRED SHOVED ELLEN \ 
OK

*SHE MEANT TO HIT HER \ 
OK

*IS MILDRED LIABLE TO ELLEN FOR ASSAULT ?
NO, MILDRED IS NOT LIABLE FOR ASSAULT OK

*IS MILDRED LIABLE TO ELLEN FOR BATTERY ?
DID ELLEN CONSENT TO THE ACTIONS OF MILDRED?
*NO \\

YES, MILDRED IS LIABLE TO ELLEN FOR BATTERY
OK

*HOW IS CONTACT SATISFIED?

MILDRED SHOVED ELLEN
OK

*HOW IS INTENT SATISFIED?

MILDRED MEANT TO HIT ELLEN
OK

*NEW \\
OK

*MARY TRIED TO HIT AARON WITH A ROCK \\
OK

*SHE THREW IT AT HIM \\
OK

*IT SCARED HIM \\
OK

*IS MARY LIABLE TO AARON FOR ASSAULT?

DID AARON CONSENT TO THE ACTIONS OF MARY?
*NO \\

YES, MARY IS LIABLE TO AARON FOR ASSAULT
OK

*HOW IS APPREHENSION SATISFIED?

IT SCARED AARON
OK

*HOW IS INTENT SATISFIED?

CONTINUE? *Y

MARY TRIED TO HIT AARON WITH A ROCK
OK

*WHO THREW A ROCK AT AARON?

MARY
OK

*WHO DID MARY THROW A ROCK AT?

AARON
OK

*NEW

OK

*ALEX SWUNG A STICK AT ELLEN

OK

*ELLEN THREW A ROCK AT ALEX

OK

*IT HIT HIM

OK

*IT FRIGHTENED HER

CONTINUE? *Y

OK

*HE INTENDED TO HIT HER

OK
*WHO DID ELLEN THROW A STICK AT?

I DON'T KNOW
OK

*WHO DID ELLEN THROW A ROCK AT?

ALEX
OK

*WHO DID ALEX SWING A STICK AT?

CONTINUE? *Y

ELLEN
OK

*WHO THREW A ROCK AT ALEX?

ELLEN
OK

*IS ALEX LIABLE TO ELLEN FOR ASSAULT?

DID ELLEN CONSENT TO THE ACTIONS OF ALEX?

*NO \

YES, ALEX IS LIABLE TO ELLEN FOR ASSAULT
OK

*IS ALEX LIABLE TO ELLEN FOR BATTERY?

DID THE STICK CONTACT ELLEN?

*NO \

NO, ALEX IS NOT LIABLE FOR BATTERY
OK

*IS ELLEN LIABLE TO ALEX FOR BATTERY?

DID ELLEN ACT INTENTIONALLY TOWARD ALEX?

*YES \

48
DID ALEX CONSENT TO THE ACTIONS OF ELLEN?
*NO

YES, ELLEN IS LIABLE TO ALEX FOR BATTERY
OK

*IS ELLEN LIABLE TO ALEX FOR ASSAULT?

DID THE ROCK CAUSE APPREHENSION IN ALEX?
*NO

NO, ELLEN IS NOT LIABLE FOR ASSAULT
OK

*NEW

OK

*JOHN INTENTIONALLY THREW A ROCK AT MARY
OK

*IT HIT HER

OK

*IS JOHN LIABLE TO MARY FOR BATTERY?

DID MARY CONSENT TO THE ACTIONS OF JOHN?
*NO

YES, JOHN IS LIABLE TO MARY FOR BATTERY
OK

*HOW IS CONTACT SATISFIED?

ROCK HIT MARY
OK

*HOW IS INTENT SATISFIED?

JOHN INTENTIONALLY THREW A ROCK AT MARY
OK
*NEW *

OK

*MILDRED FIRED A GUN AT AARON *

OK

*THE BULLET HIT HIM *

CONTINUE? *Y

OK

*IS MILDRED LIABLE TO AARON FOR BATTERY ?

DID MILDRED ACT INTENTIONALLY TOWARD AARON ? *

*YES *

DID AARON CONSENT TO THE ACTIONS OF MILDRED ? *

*NO *

YES, MILDRED IS LIABLE TO AARON FOR BATTERY
OK

*HOW IS CONTACT SATISFIED ?

THE BULLET HIT AARON
OK

*HOW IS INTENT SATISFIED ?

MILDRED INTENTIONALLY ACTED
OK

*WHO DID THE BULLET HIT ?

AARON
OK

*NEW *

OK

OK
*MILDRED TRIPPED JILL \n
OK

*SHE INTENTIONALLY TRIPPED HER \n
OK

*IS MILDRED LIABLE TO JILL FOR BATTERY ?

DID JILL CONSENT TO THE ACTIONS OF MILDRED ?
*NO \n
YES, MILDRED IS LIABLE TO JILL FOR BATTERY

OK

*HOW IS INTENT SATISFIED ?

MILDRED INTENTIONALLY TRIPPED HER

OK

*HOW IS CONTACT SATISFIED ?

MILDRED TRIPPED JILL

OK

*NEW \n
OK

*JOHN THREW THE BOOK AT ELLEN \n
OK

*IS JOHN LIABLE TO ELLEN FOR ASSAULT ?

DID THE BOOK CAUSE APPREHENSION IN ELLEN ?
*YES \n
DID JOHN ACT INTENTIONALLY TOWARD ELLEN ?
*YES \n
DID ELLEN CONSENT TO THE ACTIONS OF JOHN ?
*NO \n
51
YES, JOHN IS LIABLE TO ELLEN FOR ASSAULT
OK

*HOW IS APPREHENSION SATISFIED ?

THE BOOK FRIGHTENED ELLEN
OK

*HOW IS INTENT SATISFIED ?

JOHN INTENTIONALLY ACTED
OK

*IS JOHN LIABLE TO ELLEN FOR BATTERY ?

DID THE BOOK CONTACT ELLEN ?
*NO \ 

NO, JOHN IS NOT LIABLE FOR BATTERY
OK

*NEW \ 

OK

*JOHN KISSED MARY \ 

OK

*SHE CONSENTED \ 

OK

*IS JOHN LIABLE TO MARY FOR BATTERY ?

CONTINUE? *Y

DID JOHN ACT INTENTIONALLY TOWARD MARY ?
*YES \ 

NO, JOHN IS NOT LIABLE FOR BATTERY
OK

52
ANOTHER? *(EXIT)
20POP

[PHOTO: Recording terminated Sat 21-Feb-87 5:10PM]