A PRELIMINARY STUDY IN COMPUTER-AIDED LEGAL ANALYSIS

Jeffrey A. Meldman

November 1975

PROJECT MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge, Massachusetts 02139

This empty page was substituted for a blank page in the original document.

A PRELIMINARY STUDY IN COMPUTER-AIDED LEGAL ANALYSIS*

Abstract

This paper describes the prototype for a computer system that can perform a simple kind of legal analysis. The system user, who is presumed to be a lawyer, describes to the system a hypothetical set of facts. The system determines the extent to which these facts fall within certain legal doctrines (by syllogism), or near to these doctrines (by analogy). During this process, the system may ask the user for additional facts. The system then tells the user of its determinations and of the logic behind its conclusions, supporting these conclusions with reference to judicial decisions and other legal authority. The prototype system communicates with the user in a computer language (called Preliminary Study Language) designed to be translatable into and out of English by natural-language processing techniques, based on case grammar, that are currently being developed in other research.

As the basis for this analysis, structural machine models are built to represent legally-relevant human activity and doctrines of law. The primitive components in these structures represent simple things and relations (like persons, firearms, hitting, near, etc.) in the everyday world of human affairs. These things and relations are classified hierarchically into categories. They are assembled into facts comprising two things and the relation between them. Facts, in turn, are assembled into more complicated structures called situations, which are represented in terms of component elements, or in terms of alternative types, or both. These situational structures are used to represent the hypothetical facts being analysed as well as the factual content of legal doctrines. The factual situations of specific cases provide examples and counter-examples that behave as alternative types of the situational components of more general legal doctrine. The prototype system contains representations for doctrine involving civil battery and assault.

Analysis is performed by decomposing the situations that represent legal doctrines according to their elements and their types. When this decomposition reaches the level of things and relations, these things and relations, together with their situational structure, are matched against the things and relations contained in the hypothetical facts. The matching of individual things and relations is accomplished by reference to their hierarchical categorization.

*This report reproduces a thesis of the same title submitted to the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology on August 29, 1975 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Acnowledgements

The author is grateful to Professor Edward Fredkin for his supervision of this thesis, and to Professors William A. Martin and Patrick H. Winston for their assistance in its development. He is grateful also for the advice and assistance given him by Bob Baron, Robert P. Bigelow, Gretchen Brown, Andy Bjendorf, Cathy Fowler, Professor Sanford Fox, Ross Gale, Lowell Hawkinson, Beth Hoemke, Rand Krumland, Michael Marean, Dean Robert H. Rines, Dottie Scanlon, and Bill Swartout.

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under contract number N00014-75-C-0661 (original number N00014-70-A-0362-0006).

Table of Contents

Chapter 1 Introduction 8

1.1 Overview of the Study 8

1.2 Examples of Analysis 12

Chapter 2 Machine Models and Legal Analysis 20

2.1 The Process of Legal Analysis 20

2.2 Modeling Factual Situations 29

2.21 Machine Models of Human Activity 29
2.22 Toward a Level of Simplified Complexity 32

2.3 Modeling Legal Doctrine and Legal Analysis 39

2.31 The Development of Machine Models in Law 39
2.32 The Nature of the Model Proposed in this Study 49

Chapter 3 Representations of Human Activity 54

3.1 An Architecture Based on Things and Relations 54

3.11 Things and Relations 58
3.12 The Kind Hierarchy 61
3.13 Kinds and Instances 65
3.14 Semantic Relations 66
3.15 The Absence of Temporal Relations 69

3.2 Facts and Situations 70

3.21 Facts 70
3.22 Situations 72
3.23 The Situational Structure of the System's Knowledge 76

3.4 A More Comprehensive Kind Hierarchy 79

Chapter 4 Implementation in PSL 90

4.1 Representing Facts and Situations in PSL 90

4.2 The Scope of the Instance Names 96

4.3 Summary of PSL Notation 99

4.4 A More Complicated Example 102

Chapter 5 Representations of Legal Doctrine 105

5.1 The Torts of Battery and Assault 105

5.2 Generalized Representations of Battery and Assault 107

5.21 Battery 108
5.22 The Use of Co-Descriptive Models 116
5.23 Assault 120
5.24 Summary of Generalized Representations 125

5.3 Representations of Specific Cases 132

5.31 Specific Facts and Categorized Holdings 132
5.32 Summary of Case Representations 142

Chapter 6 Machine Procedures for Legal Analysis 147

6.1 Machine Procedures in PSL 147

6.2 The Loading Procedures 149

6.21 Building Permanent Knowledge 149
6.22 Inserting a Factual Situation for Analysis 151

6.3 The Instantiation Procedures 154

6.31 Instantiating a Fact 155
6.32 Instantiating a Situation 160
6.33 Instantiation by Query 163
6.34 The General Process of Instantiation 166
6.35 The Instantiation Record 172
6.36 The Algorithms for Instantiation 174

6.4 The Discussion Procedure 180

Chapter 7 Examples of Analysis, Explained 184

Chapter 8 Concluding Discussion 203

References 211

Biographical Note 215

Table of Figures

Figure 3-1	55
Figure 3-2	55
Figure 3-3	55
Figure 3-4	59
Figure 3-5	62
Figure 3-6	62
Figure 3-7	68
Figure 3-8	71
Figure 3-9	71
Figure 3-10	73
Figure 3-11	73
Figure 3-12	75
Figure 3-13	78
Figure 5-1	106
Figure 5-2	111
Figure 6-1	153
Figure 6-2	175
Figure 6-3	176
Figure 6-4	177
Figure 6-5	178
Figure 6-6	179

Table of Summaries

A More Comprehensive Kind Hierarchy	(section 3.3)	79
Summary of PSL Notation	(section 4.3)	99
Summary of Generalized Representations	(section 5.24)	125
Summary of Case Representations	(section 5.32)	142
The Algorithms for Instantiation	(section 6.36)	174

Reality is too complex for verbal transmission. Logic changes reality into a different form, and thus it reaches out into the world.

. . . The Alpha 60 Computer

(Jean-Luc Godard, Alphaville)

The life of the law has not been logic; it has been experience.

. . . Oliver Wendell Holmes

Chapter 1 Introduction

1.1 Overview of the Study

This paper describes the prototype for a computer system that can perform a simple kind of legal analysis. The user of the system is presumed to be a lawyer. In an analysis session, the user sits at a computer terminal and types a description of a hypothetical factual situation, using a standard alphanumeric keyboard. The system explores its internal representations of various legal doctrines, and it determines the extent to which the hypothetical facts fall within, or near to, these doctrines. Often, the system asks the user to supply additional facts that it needs in order to make these determinations. The system then tells the user of its determinations, and it explains to the user the logic according to which its conclusions were reached. Whenever possible, it supports its conclusions with references to judicial decisions and to other authoritative assertions of law. The system communicates to the user on the typewriter console or on a video display device like a cathode ray tube (which is similar to a television screen).

In order to effect this kind of computer analysis, it is necessary that we be able to construct explicit machine representations for the specific factual situations that are to be analyzed. It is also necessary that we provide the machine with similar representations for more generalized situations in terms of which legal doctrines can be expressed.

Finally it is necessary that we design machine procedures for matching the specific facts being analyzed to the more general facts contained in the doctrines.

Existing methods for representing legal doctrine in terms of explicit formalisms, such as those based upon Boolean algebra, are inadequate for the task of matching specific facts to more general facts. This is because the elementary components represented in such models are too large and complicated to permit any useful mechanism for categorization. In this study, we propose the use of structural representations whose primitive elements represent simple things and relations from the everyday world of human activity. The prototype system contains representations for several hundred things and relations, like persons, firearms, hats, hitting, believing, near, far, et cetera. All of these things and relations are classified hierarchically into categories. These simple representations can be assembled into complicated structures that are used to represent larger factual situations.

Such structures are used to represent the hypothetical facts presented by the user for analysis. Larger, similar structures are used to represent the more generalized facts contained in legal doctrine. The prototype system contains representations for doctrine in the area of civil battery and assault.

The system's procedures are able to decompose these representations of legal doctrine into smaller composite and alternative sub-structures. This decomposition can be continued all the way to the level of primitive things and relations. At this level, the specific facts being analyzed can be matched to the more general facts contained in the doctrine by reference to the hierarchical arrangement of things and relations.

The task of constructing useful machine representations of factual situations, and the task of matching specific facts to more general facts, both present significant probelms that have not yet been satisfactorally solved. In this preliminary study, some of these problems will be explored in detail. Others will be bypassed, however, due to certain simplications built into our system. For example, elementary things and relations are represented simply in terms of their locations on a hierarchy. Other features and characteristics of these things and relations are ignored. Similarly, we will be primarily concerned with techniques for assembling and decomposing complicated situations. We will be less concerned with techniques for matching individual things and relations, relying simply on the classification hierarchy for this purpose.

As a result of these, and other, simplications, the prototype system can perform analysis only on rather carefully drawn factual situations. It is hoped that this method of analysis will provide a starting point for more sophisticated analysis of a similar nature, and that this study will provide a framework for understanding the problems that must be faced in more advanced systems. Many of the problems that lie outside the scope of

of this study are already the subjects of current research in the fields of artificial intelligence and automatic programming.

The prototype system described in this study does not understand English. It communicates in an artificial language that we call PSL (Preliminary Study Language). This language is designed to be comprehendible by a computer modeling system, called OWL, which is under current development. The OWL system can translate statements expressed in a language like PSL directly into data structures and machine procedures. Ultimately, the OWL system is expected to be able to perform similar translations directly from statements in English, at which time the use of an artificial language will become unnecessary.

Finally, it must be explained that the prototype system is not yet realized, that is, it is not programmed, debugged, and running on a computer. However, the data structures and machine procedures for the prototype are described in this study with sufficient specificity and detail so that implementation of the system ought to be relatively straightforward.

1.2 Examples of Analysis

The nature of the prototype system perhaps can be introduced best by a few examples of the dialog that might ensue between the user and the system during sessions of analysis. The dialog will be presented here in English, but it should be remembered that the prototype actually communicates in PSL statements. We will examine the PSL language in later chapters.

In Chapter 7, after we have explored the structure and operation of the system, we will return to these examples and trace the details of the analysis involved.

Analysis Session 1

USER: Aaron Aardvark purposely kicked Zachary Zetz in the leg.

SYSTEM: Did Zachary Zetz consent to being kicked?

USER: No.

SYSTEM: Was Zachary Zetz apprehensive that he would be kicked?

USER: Assume not.

SYSTEM: Aaron Aardvark appears to be liable to Zachery Zetz for a battery. Aaron Aardvark appears not to be liable to Zachary Zetz for an assault.

USER: Why is there liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and the lack consent. These components appear to be satisfied by the facts at hand.

USER: How is contact satisfied?

SYSTEM: The decision in Foe v. Moe is applicable. In that case, Joe Moe punched Fred Foe in the nose. It was held that when one person strikes the anatomy of another, there is contact as required for a battery. This appears to cover the facts at hand, in which Aaron Aardvark kicked Zachary Zetz in the leg.

USER: How is intent satisifed?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or the apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for a battery. This appears to cover the facts at hand, in which Aaron Aardvark kicked Zachary Zetz purposely.

USER: How is lack of consent satisfied?

SYSTEM: When a person consents to being contacted, there is consent as will avoid a battery. This appears to be contradicted by the facts at hand, in which Zachary Zetz did not consent to being kicked.

USER: Why is there no liability for an assault?

SYSTEM: According to the doctrine adopted in Jones v. Smith, an action in assault is established by two components: apprehension and intent. Apprehension appears to be contradicted by the facts at hand.

USER: How is apprehension contradicted?

SYSTEM: When a person is apprehensive of being contacted, there is apprehension as required for an assault. This appears to be contradicted by the facts at hand, in which, according to your assumption, Zachary Zetz was not apprehensive that he would be kicked.

Analysis Session 2

USER: Fred Dobbs and Benjie Hooray are playing in a hockey game. Fred raises his hockey stick. Fred believes that this will cause Benjie to be apprehensive that he will be hit. Benjie is not apprehensive. Fred hits Benjie in the leg with his hockey stick, but not purposely.

SYSTEM: Was the hit that Benjie Hooray received a part of the hockey game?

USER: Assume so.

SYSTEM: Fred Dobbs appears not to be liable to Benjie Hooray for a battery. Fred Dobbs appears not to be liable to Benjie Hooray for an assault.

USER: Why is there no liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and the lack of consent. Lack of consent appears to be contradicted by the facts at hand.

USER: How is lack of consent contradicted?

SYSTEM: One form of consent involves inferred consent. The decision in Quoe v. Poe is applicable. In that case, the parties were fighting in a boxing match. Perry Poe punched Quentin Quoe in the jaw. It was held that when two persons participate in an athletic competition, consent as will avoid a battery may be inferred for any physical contact that is part of the competition. This appears to cover the facts at hand, in which Fred Dobbs and Benjie Hooray were playing in a hockey game, and in which, according to your assumption, the hit that Benjie Hooray received was a part of the hockey game.

USER: Why is there no liability for an assault?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in assault is established by two components: apprehension and intent. Apprehension appears to be contradicted by the facts at hand.

USER: How is apprehension contradicted?

SYSTEM: When a person is apprehensive of being contacted, there is apprehension as required for an assault. This apperas to be contradicted by the facts at hand, in which Benjie Hooray was not apprehensive that he would be hit.

Analysis Session 3

USER: With the purpose of frightening Gordon Good, Howard Hood visibly points a saturday-night special at him and grabs the umbrella that he is holding. The saturday-night special is not loaded.

SYSTEM: What is a saturday-night special?

USER: A saturday-night special is a kind of pistol.

SYSTEM: Did Gordon Good consent to the grabbing of his umbrella?

USER: No.

SYSTEM: Did Gordon Good know that the saturday-night special was unloaded?

USER: No.

SYSTEM: Howard Hood appears to be liable to Gordon Good for a battery. Howard Hood appears to be liable to Gordon Good for an assault.

USER: Why is there liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and lack of consent. These components appear to be satisfied by the facts at hand.

USER: How is contact satisfied?

SYSTEM: The decision in Roe v. Doe provides an analogy. In that case, John Doe knocked off the hat that Richard Roe was wearing. It was held that when a person strikes an article of clothing on the person of another, there is contact as required for a battery. This comes close to covering the facts at hand, in which Howard Hood grabbed the umbrella that Gordon Good was holding.

USER: How is intent satisfied?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or the apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for a battery. This appears to cover the facts at hand, in which Howard Hood grabbed the umbrella held by Gordon Good with the purpose of making him apprehensive of a physical contact.

USER: How is lack of consent satisfied?

SYSTEM: When a person consents to being contacted, there is consent as will avoid a battery. This appears to be contradicted by the facts at hand, in which Gordon Good did not consent to the grabbing of his umbrella.

USER: Why is there liability for an assault?

SYSTEM: According to the doctrine adopted in Jones v. Smith, an action in assault is established by two components: apprehension and intent. These components appear to be satisfied by the facts at hand.

USER: How is apprehension satisfied?

SYSTEM: One form of apprehension involves a threat with a gun. The decision in Zoe v. Woe is applicable. In that case, Winnie Woe visibly pointed a rifle at Zeke Zoe. The rifle was not loaded, but Zeke Zoe did not know that. It was held that where a person visibly points an unloaded firearm at another person, the other person's apprehension as required for

an assault is established if the other person does not know that the fire-arm is unloaded. This appears to cover the facts at hand, in which Howard Hood visibly pointed a saturday-night special at Gordon Good, and in which the saturday-night special was unloaded, and in which Gordon Good did not know that it was unloaded.

USER: How is intent satisfied?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or an apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for an assault. This appears to cover the facts at hand, in which Howard Hood pointed the saturday-night special at Gordon Good with the purpose of making him apprehensive of a physical contact.

Chapter 2 Machine Models and Legal Analysis

2.1 The Process of Legal Analysis

A client consults an attorney. The attorney may be asked for advice about the legal consequences of some contemplated activity. The attorney may have to draft a contract to protect the client's interests over some range of possible future situations. Or perhaps the client already has become involved in a predicament. He or she may wish to bring legal action to recover for losses or injuries caused by another, or wish to be defended in such a lawsuit--or in a criminal proceeding.

In providing counsel in such situations, the attorney uses many forms of reasoning and skill. One of these is a fundamental technique that we will call legal analysis. By this term, we mean: the logical derivation of a legal conclusion from a particular factual situation in the light of some body of legal doctrine.

Before we can understand the nature of this process, we must be careful to distinguish it from the more complex process of legal reasoning. This term is used generally to describe the process by which judges decide cases (and, therefore, a part of the attorney's overall reasoning, too). In his classic treatise on the judicial process, Benjamin Cardozo separates four major paths along which the force of legal reasoning exerts itself. These are: logical progression, historical development, custom, and social justice [1]. Regarding the force of logic,

he explains:

> In putting it first, I do not mean to rate it most important. On the contrary, it is often sacrificed to others. I have put it first because it has, I think, a certain presumption in its favor. . . . Lacking [some consideration of history or custom or justice], I must be logical.
>
> * * *
>
> You may call the process one of analogy or of logic or of philosophy as you please. Its essence in any event is the derivation of a consequence from a rule or a principle or a precedent which, accepted as a datum, contains implicitly within itself the germ of the conclusion. . . . The method tapers down from the syllogism at one end to mere analogy at the other [2].

What we are calling legal analysis corresponds substantially only to this first, logical component of legal reasoning. We will be concerned both with the logic of syllogism and with the logic of analogy.

Legal analysis is performed on a particular set of facts against a background of legal doctrine. Cardozo is referring to such doctrine when he speaks of "a rule or a principle or a precedent."

Legal doctrine is embodied in different forms, such as in the <u>statutes</u> and the <u>constitutions</u> of our state and federal governments. In common-law systems such as ours, the characteristic embodiments of legal doctrine are the judicial decisions in individual <u>cases</u>. The doctrines of prior cases, which are called "the common law," serve as <u>precedents</u> for future, similar cases, as we will discuss presently. The common law is our most important form of legal doctrine. Substantial areas of legal doctrine (especially of state law) exist only in cases--there are no statutory or constitutional provisions that pertain. Even in areas of law covered by statutes or constitutions, these are always subject to the courts'

interpretation in individual cases, and these interpretations are binding as precedents for future, similar cases.

Legal doctrine can differ in its degree of specificity or generality. We use the word specific to indicate doctrine that is expressed in terms of relatively narrow categories of persons and activities. Here is an example from a 1956 North Carolina decision:

> In short, where an internal operation is indicated, a surgeon may lawfully perform, and it is his duty to perform, such operation as good surgery demands, even when it means an extension of the operation further than was originally contemplated [3].

We describe a doctrine as more general when it speaks in broader categories, for example:

> Where the instrumentality which produced an injury is within the exclusive possession and control of the person charged with negligence, and such person has exclusive knowledge of the care exercised in the control and management of that instrumentality, evidence of circumstances which show that the accident would not ordinarily have occurred without neglect of some duty owed to the plaintiff is sufficient to justify an inference of negligence [4].

Doctrines at this level of generality are often associated with shorthand labels so that their basic concepts can be referred to more easily. The above doctrine, for example, is usually called res ipsa loquitur (the thing speaks for itself).

Even higher levels of generality are typified by such doctrines as: "Wherever there is a wrong there is a remedy to redress it," or "Whoever seeks a remedy in equity must come into court with clean hands." As can be seen, such doctrines are far removed from specific factual situations.

We will use the word doctrine to refer to expressions of law at every level of specificity or generality. On the other hand, many jurists use

this word, along with words like rule and principle, to refer only to relatively general expressions. Thus, one often reads of the "doctrine of res ipsa loquitur," or of the "clean hands doctrine." The doctrine regarding the lawful extension of surgery, being less general and less well known, usually would not be treated with such distinction.

It should be noted also that important differences exist among the legal doctrines of different jurisdictions. In the United States, there are 50 separate state jurisdictions in addition to the federal jurisdiction, which itself comprises 11 separate sub-jurisdictions, called "circuits." It is not always clear which jurisdiction's doctrines ought to apply in a particular situation. In such cases, recourse is taken to additional layers of state and federal doctrine whose sole function is to resolve such conflicts.

Legal doctrine contained in cases, statutes, and constitutions is often called primary legal authority because it is the direct verbal embodiment of the law. Another kind of legal doctrine are the restatements of law compiled by legal scholars in the form of treatises, law review articles, and legal encyclopedias. This is called secondary authorrity, and it is given far less weight (usually none) in the actual process of deciding a case. But secondary authority plays a definite role in the process of legal analysis. Particularly in areas of common-law doctrine, an attorney is faced initially with too large and too disorganized a body of doctrines against which to analyze a particular set of facts. Instead, the attorney must perform an initial analysis in terms of the more organized, coherent, condensed (and more generalized) body of restated

doctrines, which are more easily accessed and recalled.

With the differences among the various kinds of doctrine in mind, let us consider the actual operation of legal analysis. For common-law doctrine, the process amounts to the invocation of one basic rule--the rule of precedent, or stare decisis (stand by the decisions), which we will state like this:

> The legal consequence of a particular factual situation (in a given jurisdiction) must be the same as was the result of any previous case (in that jurisdiction) that involved the same factual situation.

It should be apparent that this doctrine has the highest possible degree of generality. It applies to all factual situations, regardless of their nature. At the same time, if the doctrine is interpreted in its strictest possible sense, it is virtually useless because of the small likelihood that the particular set of facts at hand is going to be exactly "the same" as the facts in some prior case. If the rule is to be meaningful, the word "same" must be understood in the sense of "same kind of" or "similar." Certain differences between the facts at hand and the facts in the prior case must be ignored. Which ones? And how large may the differences be? Having to answer these questions is what makes the application of so simple-looking a rule so difficult.

In practice, this problem is approached through two mechanisms, both involving generalization. In the first place, the holding of a case (that is, the particular piece of doctrine for which the case stands as authority) is almost always intended and understood in terms more general than the specific persons, objects, activities, et cetera, that were actually

involved in the case. Judges understand well the role that their written opinions play as precedents for future cases, and they tend to write in terms of categories rather than individualities.

Consider the holding in the North Carolina case quoted on page 22. We cited it as an example of a relatively specific piece of doctrine; nonetheless, it contains a certain amount of generalization. The judge speaks not in terms of the individual defendant, but in terms of "surgeons," a category of persons that includes the defendant in this case as well as other surgeons who might be future defendants in similar cases. And the holding is not intended to be restricted to operations exactly like the one in this case, nor even to operations of a particular kind (this had been an appendectomy), but to the entire category of "internal operations."

What is the appropriate scope of such categories? This is determined by the reasoning that underlies the decision. In this case, Chief Justice Barnhill's stated reasoning included such factors as the known difficulty of presurgical diagnosis of internal ailments, the unavailability of obtaining further consent due to general anesthesia, and the desire to encourage surgeons who may be tempted to shirk from duty for fear of a lawsuits. (Remember that the process of judicial reasoning includes many other intellectual activities besides what we are calling logical analysis.) Clearly if this reasoning is valid for this individual defendant performing this particular operation, it is also valid for any surgeon performing any internal operation. Thus, he stated his holding in these terms. Ultimately, the holding in any case is determined not by the words

of the judge who wrote the decision, but by future judges who interpret the decision. When a particular holding is thought to be unreasonably broad, which sometimes happens, it is interpreted more narrowly.

The terminology of generalized categories, then, is the first mechanism by which the rule of precedent can be meaningfully used. It eliminates the need to match the facts at hand exactly to the facts of a previously decided case. It requires only that the facts at hand fit into the categories in terms of which a previous deicsion was written. Where this can be done, the result of the prior case determines the legal consequence of the facts at hand. The resulting logic is like that of a classic syllogism: All men are mortal; Socrates is a man; therefore, Socrates is mortal.

The second mechanism for generalization goes beyond the logic of syllogism into the logic of analogy. This method is invoked when the facts at hand fall near, but not within, the scope of a prior holding. Once again the underlying reasoning of the prior case is the key. If that reasoning appears to apply with equal force to the facts at hand, then (absent other precedent, of course) the result in the prior case is controlling. Note that this is not a reassessment of the scope of the prior holding. It is an argument based on similarity, not on inclusion.

For example, consider the situation in which a dentist, while extracting a tooth from a patient under general anesthesia, discovers the necessity of extracting a second tooth, and does so without obtaining additional consent. These facts fall outside the categories "surgeon" and "internal operation" used in Barnhill's opinion. Yet it can be argued

that much of Barnhill's reasoning applies equally well to these facts. On the other hand, consider the case of a garage mechanic who performs additional automobile repairs without the consent of the customer. Barnhill's reasoning is mostly irrelevant here, and the logic of analogy fails.

Through repeated use of analogy, the reasoning in individual cases gradually becomes extended to categories much broader than can be dictated by the facts of any single case. At some point, a perceptive judge may become aware of this growth, and reformulate a doctrine, or combine several doctrines, in appropriately broader terms. The more general common-law doctrines (like res ipsa loquitur) usually evolve in this manner.

When categories of fact become more general (e.g., "exclusive possession and control") it becomes more difficult to recognize whether individual facts do or not not fit into the categories. The solution to this problem sometimes lies within the purview of legal analysis. There might, for example, be previous similar cases in which the judge ruled one way or the other on this point. Often, however, such questions are left to the intuitive reasoning of the fact-finder (e.g., the jury) in a trial.

Legal analysis applied to statutes, to constitutions, and to the restated doctrines in secondary sources, operates much in the same manner. Such doctrine tends to be written in terms of broader categories than those usually found in the holdings of judicial cases. One of the ways in which courts "interpret" statutes and constitutions is to make decisions

about which facts do, and which facts do not, fit into stated categories.

There are other ways in which doctrines interact with each other (e.g., when the same factual situation clearly falls within the scope of two conflicting holdings), but we will not examine these here.

By now the essence of this process of analysis should be recognized: It is the fitting of a particular factual situation into, or sometimes near to, the categorized situations as expressed in legal doctrine, either by simple intuition or by the similar application of additional doctrines.

In order to enlist the assistance of the computer in performing a task like this, it will be necessary to construct machine representations, or models, of the components in the process. We will have to build models for representing factual situations, for representing legal doctrine, and for representing the fact-fitting process itself. Some preliminary asepcts of building these models are examined in the following sections.

2.2 Modeling Factual Situations

2.21 Machine Models of Human Activity

As human beings, we generally are able to engage successfully in social activity. This reflects the existence within each of us of an "operational" model that guides behavior. This model is partially inborn and partially acquired, but it is almost entirely implicit. A considerable portion of human intellectual effort throughout history has been spent in trying to transform pieces of this implicit model into more explicit formalisms. As artists, philosophers, and scientists, working in the frameworks of different cultures, and using different conceptualizations and methodologies, we have generated a vast assortment of images, stories, visualizations, generalizations, and metaphors--all models of human activity and interaction.

Some of these models have been basically normative; they have been used to prescribe human activity. Religious and legal codes are examples. Other models are basically descriptive; these are often used, in conjunction with formulated theories, to explain and to better our understanding of human activity. Freud's hydraulic metaphor for psychic energy is an example.

For the purpose either of prescribing or of explaining, a model's being explicit has important advantages. Formal expression facilitates communication, generalization, deduction, and extension, each of which can amplify greatly the normative or descriptive power of the model.

Until recently, it was most common for explicit models of human activity to be expressed in natural human language. At the same time, some models of other worldly phenomena--especially physical phenomena that were readily observed in terms of quantities and measures--came to be expressed as more highly explicit mathematical representations. In the physical sciences, models increasingly took the form of equations, which were taken to represent "laws of nature."

More recently, this use of law-like mathematical representation has been emulated in the behavioral and social sciences, but with less satisfactory results. Mathematical relations, expressed as equations, have been used to describe many narrow slices of human phenomena. In psychophysics, for example, a major goal has been the discovery of mathematical equations that relate quantities of stimulation to quantities of evoked sensation. Mathematical models also have been developed to relate formally defined economic quantities and activities. Techniques exist by which relationships among social phenomena can be posited by analogy to models of physical systems, and can be measured--or even inferred entirely--from statistical data. In this way, mathematical models have been generated for such phenomena as population growth, political outcomes, advertising effectiveness, et cetera.

The extent to which such mathematically explicit models, in the presence or in the absence of explanatory theory, help us to understand human activity is not clear. Even so, these models can sometimes be valuable for the solution of our practical problems. Thus, a mathematical

model that simulates the level of population over time may be useful in the planning of food supply, even if it is not squarely based on a theoretical understanding of the processes involved.

The value of the mathematical model, whether the model is used to solve problems or to increase understanding, or both, is enhanced greatly by the use of computers. The classic advantage in this regard is the computer's ability to perform arithmetic manipulations on data, millions of times faster than people can. A large number of models and modeling techniques take particular advantage of this ability. These "machine models" involve inhumanly many arithmetic manipulations, and their use is feasible only with the aid of the computer.

The development of one particular class of computer languages (called "list processing" languages) has provided the computer with another important advantage. With the use of these languages, the machine is able to perform rapid structural, as well as arithmetic, manipulations on data. This makes the computer a powerful aid for formal models that are not mathematical in the numerical sense. In numerical models, relationships generally are expressed as equations involving variables that take numerical values. More general, structural models permit a variety of different kinds of relationships to be expressed among a variety of different kinds of components. "Family trees" and "tables of organization" are examples of very simple structural models of certain social phenomena. The rules of English grammar and strategies for winning at chess embody structural models that are more complex.

For the purpose of representing factual situations in the computer, we will build a descriptive, structural, machine model of legally-relevent human activity. The model will include explicit representations of persons, of physical objects, of events, and of the different relationships that occur among persons, objects, and events. We will use the model for representing: (1) a factual situation that is being analyzed, (2) facts in previous judicial decisions, (3) the categorized factual content of legal doctrine, and (4) general factual knowledge about the world of human activity--especially the way in which pieces of that world fit intuitively into categories.

2.22 Toward a Level of Simplified Complexity

To a certain extent, every model is a simplification of the subject modeled. This is as true for a complicated model of world dynamics as it is for a simple model airplane. The purpose for which the model is built generally determines which aspects of the subject are fundamental and need to be preserved, and which can be ignored in the name of economy.

Our model for human activity must be sufficiently simple so that we can understand and control its behavior in the machine. On the other hand, to be a useful aid in performing legal analysis, the model must preserve those aspects of human activity that characteristically are involved in legal analysis. Unfortunately, such human activity is complicated, so that even in our simplified model, there must be room for considerable complexity.

Realizing the proper level of "simplified complexity" in a structural model is not a trivial problem. A brief examination of two recent studies might help us put it in perspective.

Terry Winograd has designed a system of machine procedures with which a computer can answer questions, execute commands, and absorb information in a dialog of natural English expressions [5]. The system handles syntactic and semantic structures of great complexity, but the world about which it is able to converse is greatly simplified. It is a world of toy objects--blocks, boxes, pyramids--and a simulated robot hand that can "move" the objects about on a platform. (The computer is part of the world to the extent that it executes moves and answers questions, and the user is represented as the source of commands and questions.) The system contains a detailed procedural model of what can exist and what can happen in this toy world. It "understands" English expressions about the world ultimately by reference to this model. As a result, the linguistic capabilities of the system, while quite impressive, are severely limited with regard to subject matter. Such a system can not easily be extended, for example, to handle conversations about human activity.

In a more recent study of natural language comprehension, Euguene Charniak proposed a model that specifically included concepts of human activity [6]. As a simplified version of human activity, Charniak chose the world of children's stories. Within this realm, he explored in detail the topic of piggy banks. The study clearly demonstrated that in order for the machine to "know" enough about piggy banks to permit discourse

about them, it is necessary that the machine have knowledge about a great many other things in the world of children's stories--and in the world in general as well. Indeed, the world of children's stories appeared not to be significantly less complicated than the world of human activity in its entirety.

For our model, we will attempt to reach a level of simplification that lies somewhere between the levels in these two examples. Clearly we need to be able to represent things and relations that are more varied and numerous than those in a world of geometrical toys. We need a model in which, as in the world of children's stories, things can be "known" in relation to knowledge about many other things. Our model will include this structural feature. However, we will not try to account for everything that even a child might know about any one thing. Many of the things that even a child knows about the world are difficult to represent in a computer because they are based upon sensory experience. In this regard, more abstract concepts, like those in law, are perhaps easier to model because they are based upon artificial, linguistic constructs. As we will see in Chapter 3, our model comprehends a great many interrelated aspects of human activity, but it treats each aspect incompletely. The system is able, however, to ask the human user for more information about the world when necessary. The basic architecture of the model also permits a straightforward "fleshing out" to greater levels of detail as desired.

Our attempt to realize a simplified level of complexity is aided considerably by the focus of our subject matter. The law itself provides important simplifications of human activity, while retaining (and sometimes adding to) its essential complexity.

In one sense, this is because our attention is limited to a subset of human activity--activity that is relevent to the operation of law. But the world of children's stories is limited in a similar way, and this fact alone does not guarantee a significant simplification. What is more important is that the law embodies models of human affairs, which models are already significantly simplified.

The function of law in society sometimes is viewed as threefold: to settle disputes, to provide realistic expectations about contemplated behavior, and to teach the "right" way to behave [7]. For each of these purposes, the law must incorporate behavioral models, both normative and descriptive. For the law to function successfully, these models have had to be workably simple.

Thus, the law imposes discreteness and quantification upon phenomena that may be perceived by general human cognition as continuous and subtle: The defendant is either quilty or not guilty. A lawsuit must be brought as either this kind of action or that. Damages have to be measured in dollars. And the law imposes thresholds that must be crossed before human activity is recognized: Broken social promises, hurt feelings, religious impropriety, none of these can be complained of in court.

For the effective resolution of factual disputes, our legal system relies heavily on simplification. In place of a scientist's or a philosopher's concept of objectivity, there is a straightforward operational model of truth. The parties to a dispute present evidence to a fact-finder (which may be a jury, or a judge acting specifically in this role), the fact-finder makes a decision, and the facts are thereby determined. It might be a matter of weighing conflicting evidence, or of deciding whether or not certain facts fit within certain categories expressed in legal doctrine (e.g., do the facts presented amount to "exclusive possession and control"?). Where the rules of evidence and the substantive doctrines of law turn such issues over to the fact-finder, the decision of the fact-finder is binding. The law does not concern itself with how such decisions are made.

The rules of evidence themselves are replete with simplifying "presumptions." An out-of-court statement heard by a testifying witness, for example, is not admissible as evidence because it is not subject to cross-examination, and therefore it is presumed to be unreliable (mere hearsay). However, if the out-of-court statement is made by a person who is (or believes he is) dying at the time of the statement, then it is admissible on the presumption that dying statements are truthful [8].

If the law did not project such simplifications onto the world of human activity, it would not be able to function. At the same time, if the law did not substantially reflect the characteristic complexity of human activity, it would be of little practical value. We certainly need

not cite examples of how complex a legal issue can become when necessary (and, unfortunately, sometimes when it is not so necessary). Our attempt to reach a useful level of simplified complexity in our modeling is supported implicitly by the law's congruent propensity.

Compared with many other complex aspects of human affairs, including those with which the law concerns itself, legal doctrine itself is relatively discrete, quantified, and well structured. This is due partly to the kinds of simplification already mentioned. But it is due also to the fact that legal doctrine, however it might be inspired, is an artifact of intellectual effort. Its primary embodiment is in the relatively explicit form of natural language. It is learned, argued, explained, and taught in a traditionally disciplinary manner. It has evolved as an enumerable set of distinctly separate areas, such as the law of torts, the law of property, the law of contracts, criminal law, and so on. For the most part, these areas evolved independently of one another. Compared with the structural complexity found within each area of law, there is little structural interaction between them. Within each area, there is further compartmentalization. As we will see in Chapter 5, an action in tort (in which one private party seeks redress for a wrong committed by another) must be brought in a clearly delineated category, like battery or assault. Each such category, in turn, comprises a well defined set of component elements and defenses. Complicated issues are carefully dissected into smaller, discretely manageable pieces. The structure and the

compartmentalization of Anglo-American legal doctrine is particularly rigid because of its strong reliance on precedent.

In our modeling effort, we will take considerable advantage of the structure and compartmentalization inherent in legal doctrine itself.

2.3 Modeling Legal Doctrine and Legal Analysis

2.31 The Development of Machine Models in Law

Up to this point, we have been discussing legal analysis as a process in which factual situations of human activity play an important role. Viewed from another frame of reference, however, legal analysis itself must be recognized as a part of human activity. As was postulated for human activity in general, participants in legal analysis are guided, to a large extent, by internal, implicit models--both inborn and acquired. And, just as for other kinds of human activity, considerable intellectual effort has gone into the transformation of these models into more explicit formalisms. Indeed, there is a voluminous body of learning, called jurisprudence, that is devoted to the descriptive and normative explication of the nature and functioning of law.

Until recently of course, legal models universally were expressed, by legal scholars and others, in the form of natural-language discourse. Issues of logic, of history, of custom, and of social justice and morality--so nicely separated in Cardozo's 1921 treatise--tended to be woven inextricably together [9]. During the Nineteenth Century, however, some legal scholars became increasingly attracted to the logical philosophies of the natural sciences. The result was a school of thought, called <u>analytical jurisprudence</u>, focused primarily on the logic of legal reasoning. This focus opened the door to legal models that were

expressible in terms of more explicit formalisms.

A classic example is the system of analysis proposed by Wesley Hohfeld in 1923 [10]. The system is based on four "elements," called rights, privileges, powers, and immunities, and on their counterparts, called duties, no-rights, liabilities, and disabilities. In Hohfeld's model, legal doctrine is expressed by treating these elements as relations (he called them relatives) among individuals and their actions. Legal analysis is a matter of following a small set of logical rules that operate on the elements. For example, if A has a duty to B to do X, then B has a right to demand that A do X. Then, if A does not do X, A gains a power over B to do Y (e.g., recover damages in a court of law), and B beocmes liable to A with respect to Y.

This kind of approach to legal reasoning was not well received by legal scholars in general. This was due partly to traditional jurists' fears that analytical jurists were trying to reduce the entire art of law to mere logical science. It was due partly to the fact that many analytical jurists (Hohfeld was not among them) were trying to do precisely that, of course, without successs. Hohfeld emphasized that the analytical method was only part of the solution to legal problems. Nevertheless, his work, which was perhaps the most promising, was the last major effort in the school of analytical jurisprudence.

In 1949, Lee Loevinger wrote a now famous law review article severely cirticizing jurisprudence in its entirety [11]. He proposed the creation of new discipline, to be called jurimetrics, in which legal problems would

be investigated according to principles and techniques of modern science. He referred to problems of legal reasoning, legal language, legal evidence, the methodologies of jurors and legislatures, and the efficacy of legal doctrine as a whole. He complained that these problems had been approached for over 2,000 years on the basis of speculation, supposition, and superstition. Traditional jurisprudence, he suggested, "bears the same relation to a modern science of jurimetrics as astrology does to astronomy, alchemy to chemistry, or phrenology to psychology [12]." As might be expected, his proposal was largely ignored--at least for the time being.

During the ten years that followed, Georg von Wright introduced and developed an analytical model called deontic logic [13]. Von Wright, who is a logician and not a lawyer, was aiming toward a mathematical logic to describe the obligations that run between people (sometimes called normative logic). He was not expressly attempting to model legal doctrine or analysis, and yet his system is remarkably similar to Hohfeld's. Von Wright's model, however, is more highly explicit and involves a more powerful calculus of logic. If this system were to be applied in the law, legal doctrine would be expressed mathematically in terms of commands and permissions, relating sources of authority to the acts and forbearances of individuals. These acts and forbearances, in turn would be expressed mathematically in terms of elementary states of affairs and the transitions between states. As a simple example of his method of representation, here is the kind of expression von Wright would use for a

command for someone to open a window [14]:

$$O\{d(pT\sim p)\},$$

where the symbol:	O	means	a command or obligation,
	d	means	an act of someone,
	p	means	the state "the window is closed,"
	T	means	a transition between states,
	$\sim p$	means	the logical negative of p, i.e., "the window is open."

In the 1960's, an interest in logical techniques returned within the legal community. This was spurred mainly by the efforts of Layman Allen, who started a small research journal called Modern Uses of Logic in Law, and by the rapid advances of computer technology. Lawyers were becoming interested in the possible uses of automation in various areas of legal activity. Straightforward methods of data processing were being applied to tasks like law office management [15], court administration [16], document management in litigation [17], tax return preparation [18], title searching [19], police operations [20], and so forth. None of this involved legal analysis per se, but it did focus attention on the more mechanical aspects of the legal process. And this had an important effect. The computer's infamous "demand" for precise instructions and acurate data was making some lawyers aware, perhaps for the first time, that the English language, which is the lawyer's stock in trade, is not necessarily the pinnacle of precision or accuracy. This realization did not imply that the lawyer's natural language needed to be replaced by

something more explicit. However, formalisms more explicit than English were starting to appear useful for those particular legal functions where mechanically explicit behavior was, in fact, intended.

For Layman Allen, legal logic was such a function. He built models of legal doctrine and legal analysis using the simple, but highly explicit, formalisms of symbolic logic and propositional calculus [21]. His object was to eliminate those errors of logic (contradictions, ambiguities, non sequiturs) made unintentionally when statutes and judical opinions are written in ordinary English prose. According to his model, a statement of legal doctrine is restated in the form of two major propositions, one of which is a set of legal consequences, and the other is the set of conditions that imply these consequences. As a simple example, let us restate the doctrine of res ipsa loquitur quoted on page 22. The legal consequence in that piece of doctrine might be stated as the following proposition (which we will call Proposition P):

> Evidence of circumstances which show that the accident would not ordinarily have occurred without neglect of some duty owed to the plaintiff is sufficient to justify an inference of negligence.

The set of conditions that bring about this consequence can be stated as the following proposition (Proposition Q):

> The instrumentality which produced an injury is within the exclusive possession and control of the person charged with negligence, and such person has exclusive knowledge of the care exercised in the control and management of that instrumentality.

The doctrine is then restated in the form:

If Proposition Q, then Proposition P,

or, equivalently:

Proposition Q implies Proposition P.

As is often the case, Propositions P and Q themselves can be stated in terms of smaller propositions, combined by Boolean operators. These operators are represented by words like and, or, not, and implies. Their meaning is similar to that of ordinary English, but their definitions are mathematically precise. In our example, we can identify four smaller propositions within Propositions P and Q:

Proposition A: The instrumentality which produced an injury is within the exclusive possession and control of the person charged with negligence.

Proposition B: Such person has exclusive knowledge of the care exercised in the control and management of that instrumentality.

Proposition C: There is evidence of circumstances which show that the accident would not ordinarily have occurred without neglect of some duty owed to the plaintiff.

Proposition D: An inference of negligence is justified.

Proposition Q is understood to be logically equivalent to the Boolean expression:

(Proposition A and Proposition B).

Similarly, Proposition P is equivalent to:

(Proposition C implies Proposition D).

Therefore, the whole doctrine can be expressed:

(Proposition A and Proposition B)

implies

(Proposition C implies Proposition D),

or, more symbolically:

$$(A \wedge B) \Rightarrow (C \Rightarrow D)$$

where the symbol $\wedge$ means and, and the symbol $\Rightarrow$ means implies. The rules of Boolean logic then dictate, for example, that an inference of negligence is justified whenever Propositions A, B, and C all are true.

This simple example illustrates propositional logic, but it does not demonstrate its advantage over the ordinary logic of the English language. This advantage shows up in more complicated examples involving numerous propositions and logical operations. Large statutes often involve this level of complexity. Here it is not always easy for the mind to keep track of all the pieces at once, or to perform the logical steps without error. Of course, the computer is suited well to the task of keeping track of large numbers of data items and of performing the arithmetic manipulations called for. Allen's model is very useful for performing this kind of logical analysis on complicated doctrine that is represented in the form of propositions [22].

Another computer application, whose development also began around 1960, involves a machine model for a different aspect of legal analysis.

This is the problem of legal research, that is, the process of locating pieces of legal doctrine that may be relevant to facts being analyzed. "Finding the law," as this process is sometimes called, can be the most time-consuming part of legal analysis. A lawyer's analysis ultimately must be based on primary authority--the exact language of constitutions, statutes, and case decisions.

A constitution is relatively brief, and can be searched easily when necessary. A set of statutes, on the other hand, can present a problem. The Massachusetts General Laws (unannotated), for example, fill more than 3,000 pages. Of course, most legislatures attempt to organize their statutes so that doctrines pertaining to related areas of activity are grouped together in the same or adjoining segments. These chapters and sections are also given descriptive titles, and their contents are usually outlined at the beginning of each major segment. These aids help considerably in the finding of relevant doctrine. Even so, the scheme of organization is not always sensible from the researcher's point of view, nor is it consistently adhered to. It is usually necessary to explore numerous approaches in order to find a section or sections that might apply to a particular set of facts.

For case law, if it were not for separate means of assistance, the task of research would be hopeless. The decisions of the Massachusetts Supreme Judicial Court fill over 350 volumes, each of which contains an average of over 200 cases. Like the cases of all jurisdictions, these are printed and bound in simple chronological order. The titles of the cases

indicate nothing but the names of the parties. There is no way in which they could be read or even sampled in search of relevant doctrine.

Indirect methods of case-law research had to be developed long before the advent of computer technology. We already mentioned the restatements of legal doctrine found in legal encyclopedias, treatises, and law review articles. These sources of "secondary authority" are produced and oragnized in terms of major legal topics (like the law of torts), sub-topics (battery, assault), and so on. Besides providing the lawyer with smaller, compartmentalized bodies of doctrine with which to perform preliminary analysis, these materials usually contain references to the cases on which the restatements are based. Such cases often turn out not to be directly relevant, but they are a valuable entry point into the case law itself.

Other means specifically intended to assist case research have been devised. The West Publishing Company has promulgated a widely-used key-number system. The system is based on a hierarchical arrangement of legal issues that runs from major topics at the top all the way to specific holdings of cases at the bottom. Numerical values are assigned to these specific issues such that (theoretically) similar holdings receive the same, or adjacent, numbers. West publishes a digest of case law in which can be found the restated holdings of cases, arranged according to these numbers. West's encyclopedia, Corpus Juris Secundum, is also arranged according to these numbers, and the reprints of cases that West publishes for every jurisdiction have these key numbers inserted

appropriately in the text. Another publisher, Shepard, produces a citation index that lists, for each case within a given jurisdiction, all later cases in which the judge makes some reference to that particular case. By "shepardizing" a case, a researcher can determine what role the holding of that case has played in the reasoning of more recent decisions.

Since the process of legal research appeared largely to be a matter of "data retreival," it was to be expected that the computer would be called on for assistance. Some early attempts were made to computerize the systems of West and Shepard; these systems were neither innovative nor successful. However, a new technique for research, which took particular advantage of the computer's high processing speeds, fared considerable better. This is the method of full-text indexing and logical inquiry, sometimes called "key work in combination" [23]. Its use in law was deveoped by John Horty, who was himself engaged in legal reserach involving the health statutes of the various states. The differing methods by which the states organized their statutes was presenting severe problems--especially for those states where laws related to health were scattered and buried in laws pertaining to other matters. Horty's solution was to load the full text of a statute into a computer file, have the computer remove the insignificant words (like a, the, and), and then have the computer produce an index listing the exact locations in the statute of each of the significant words. With this index in the machine's memory, he could ask the computer to identify or to print out every statutory section containing a particular word, like "health." Or he could specify

a logical combination of words, and ask, for example, for all sections containing both "health" and either "dangerous" or "hazardous." By 1970 this technique was applied to case law as well as to statutory law [24]. Systems of both kinds are now available commercially.

The growing interest in computerized legal research was accompanied by efforts to apply other kinds of mathematical models to law. Reed Lawlor pioneered the use of statistical analysis in the modeling of judicial decision-making [25]. The use of probability theory in the process of fact-finding was explored [26]. Lee Loevinger's 1949 law review article was attracting new attention, and his term jurimetrics was being used to describe much of this activity. In 1966, the title of Modern Uses of Logic in Law was changed to Jurimetrics Journal. It has become the official publication of the American Bar Association's new Section on Science and Technology. In 1972 over twenty law schools offered courses involving these and other quantitative methods [27].

2.32 The Nature of the Model Proposed in this Study

The computer system examined in this study constitutes an operational model of legal analysis. It is based upon the description of logical analysis contained in section 2.1. The characteristic nature of our model can be understood most clearly in relationship to models like those discussed in the previous section.

We are concerned chiefly with a method of representing factual situations and legal doctrine in a manner that facilitates the process of

fitting the facts of a particular situation into, or near to, the factual categorizations expressed in doctrines. For this purpose, we are proposing the use of structural representations. These representations comprise relatively complicated structures assembled from primitive data items that represent relatively simple things and relations in the everyday world.

In contrast, representations like those used by Layman Allen are assembled according to the formalisms of Boolean algebra. In such models, factual situations and legal doctrines often can be decomposed into smaller propositions connected by logical operators. This decomposition cannot be carried down, however, to the level of primitive things like persons or physical objects. For example, recall the propositional representation of the doctrine of res ipsa loquitur described on page 43. We were able to break this doctrine into four logically connected subpropositions, one of which was:

> Proposition C: There is evidence of circumstances which show that the accident would not ordinarily have occurred without neglect of some duty owed to the plaintiff.

Under strict Boolean analysis, this proposition can be represented only as an indivisible chunk. No further Boolean decomposition is possible. There is no way to represent the internal factual pieces of this proposition (pieces like "evidence," "accident," and "plaintiff"), nor the non-Boolean relations among such pieces. This makes it very difficult, if not impossible, to determine by means of a mechanical procedure whether or not some specific factual situation matches the situation that the proposition

represents. Our structural model will be built from very small factual pieces and a large assortment of relations, including Boolean relations. This will permit us to match complicated situations with one another by comparing simple things and relations.

The representations of Wesley Hohfeld include a level of decomposition that is slightly more powerful than that of purely Boolean logic. Recall that Hohfeld expressed legal doctrine in forms like: "A has a duty to B to do X," His model contains a small number of legal relations, like right and duty, in addition to logical relations, like and and or. By using this model, we could break factual situations down at least to the level of persons and their activitites. Such activities, however, must still be represented as indivisible chunks (like "X").

Von Wright's deontic logic allows further decomposition. The activities of individuals can be expressed as (Boolean) combinations of states and the transitions between states. Even so, von Wright's states (e.g., "the window is closed") also are represented as indivisible wholes. They have meaning only in terms of being the logical negatives of other indivisible states (e.g., "the window is open"). Smaller factual pieces (like "window" and "closed") are not actually represented, and they are not available for purposes of matching.

By supplementing the basic set of purely logical relations with more and more specialized relations, we can achieve higher and higher degrees of decomposition. It is not at all obvious how much decomposibility is needed to permit relatively simple matching of component pieces, and we

will not try to solve that problem in this study. Instead, we will explore a highly decomposible, structural model in which matching can be performed at varying levels of decomposition. The matching of the smallest factual peices is accomplished by the arrangement of these pieces into a hierarchy of categories. The matching of larger pieces is accomplished in terms of structures of composite and alternative sub-structures of composite and alternative sub-structures.

Issues of Boolean logic, of factual decomposition, and of categorization, also play important roles in the manual and the automated systems used for legal research. The West key-number system, for example, is based on an underlying model of legal doctrine in which the holdings of all cases are arranged hierarchically into finely distinguishable categories. This system operates on the assumption that the researcher already has, or will come to have, a corresponding hierarchical model in his or her mind. As proponents of the computerized full-text research systems like to point out, this assumption is often erroneous. Some researchers find the finer classifications in West's hierarchy to be arbitrary, confusing, and sometimes incorrect. The categorization of case examples in a system like ours, however, is used by the system itself, not by the user. The user does not have to learn (or agree with) the details of the hierarchy.

In automated full-text logical inquiry systems, case decisions are represented by the set of all significant words that were used in writing the decision. The researcher must be able to predict which words a judge is likely to have used in those cases, and only those cases, that are relevant to the situation being researched. This model involves a high degree of decomposition; the elementary pieces in the model are individual words. But the structural relations among these pieces are not included in the model. The researcher can ask the system to indicate cases containing a logical cominbation of the presences and absences of certain words, but other relations among these words are ignored. As a consequence, the system produces a high percentage of irrelevant cases. In the model proposed in this study, the relational structure of the factual components is explicitly represented. Current full-text retrieval systems also do not include mechanisms for recognizing words according to categories to which they bolong. The model proposed here does include such a mechanism. If a model like ours were to be used in a case retrieval system, it is likely that fewer of the irrelevant cases, and more of the relevant cases would be retrieved. Of course, the use of this kind of model would require that large numbers of case decisions be translated--either manually or automatically--into representational data structures. Whether or not this effort would be worth the possible inprovement in performance is not clear, and would have to be the subject of further research.

Chapter 3 Representations of Human Activity

3.1 An Architecture Based on Things and Relations

The basic pieces with which we will construct machine representations of factual situations will be called things (all kinds of things, as we will see) and relations. Each thing and each relation is represented in the computer by a unique data item with which we will associate a name. The difference between a thing and a relation in our model is simply that a relation always "runs" from one thing to a second thing. This concept can be represented in the computer by linking together the addresses of the memory cells representing the relation and the two things. There is no need for us to examine the details of this technique here. In list-processing languages, such as the one we will use, this method of machine representation is straightforward.

In this chapter, we will describe our computer model by means of graphical representations, which are better suited for human comprehension. We will represent a thing by drawing a point (or sometimes a rectangle). We will represent a relation that runs from one thing to another by drawing an arrow between two points. Beside the points and arrows we will write the names of the things and relations they represent. Figure 3-1 represents two things, called thing-A and thing-B, and the relation between them, called relation-R. (The rules of list-processing languages require that we use hyphens instead of spaces within the name of a single thing or relation.)

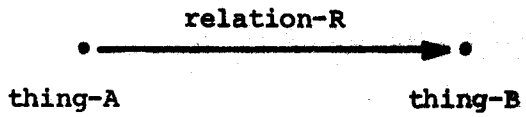


Figure 3-1.

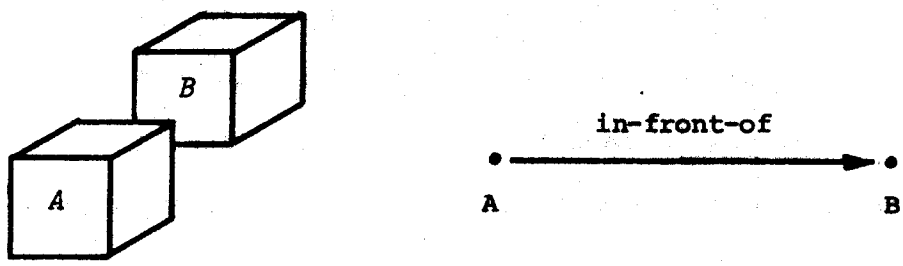


Figure 3-2.

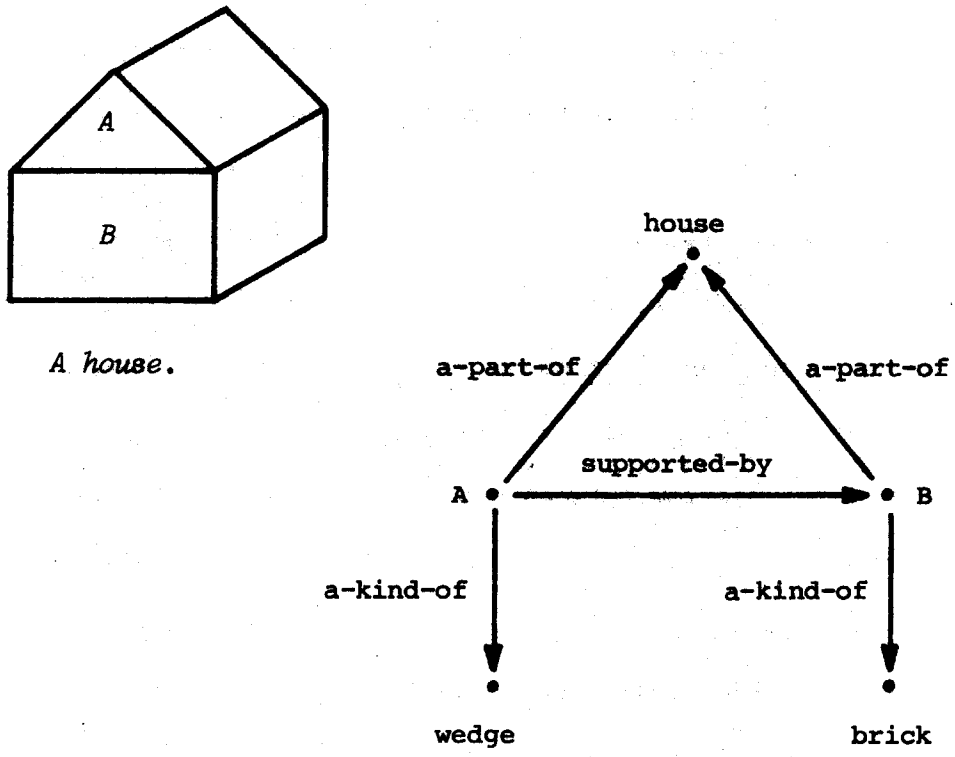


Figure 3-3.

The use of this system or representation can be illustrated with some examples described by Patrick H. Winston for modeling visual scenes comprising building blocks [28]. Figure 3-2 contains a visual scene with two simple cubes, one positioned in front of the other. The graph to the right of the scene contains two points representing the two cubes, and an arrow running between the points, representing the spacial relation in-front-of. Notice that the direction of the arrow is used to indicate which way the relation runs, i.e., which cube is in front of which.

Figure 3-3 illustrates a structure that Winston calls a house. The representation for the structure as a whole tells us that the house has two parts, one, which is a kind of wedge, that is supported by the other, a kind of brick. This representation contains not only a physical relation between two objects, but conceptual relations: membership in a composite thing and membership in a category of things. These two relations play a critical role in the representations we will develop.

The world that Winston can represent, like the world of Winograd mentioned earlier, is a world of cubes, bricks, wedges, and their combinations as houses and arches. We want to use the same fundamental architecture to represent a world of people, events, and legal doctrine--or, at least, a simplified version of this world. We might begin by

noting some of the things and some of the relations we will want to include.

Some <u>things</u> in our world:

- people
  - John Doe
  - Richard Roe
  - Marsha Moe
- objects
  - bricks
    - brick A
  - briefcases
  - firearms
- judgments
  - judgment for the defendant
- events
  - running
  - selling
  - telling
- .
- .
- .

Some <u>relations</u> in our world:

kind	(a-kind-of)
part	(a-part-of)
parent	(a-parent-of)
monetary-value	(the-value-of)
attorney	(the-attorney-of)
owner	(the-owner-of)
expectation	(the-expectation-of)
belief	(the-belief-of)
.	
.	
.	

Next, we might note how some of these things and relations could be assembled to represent fact-like states of affair:

1. firearms are a kind of object.

2. John Doe is the owner of brick A.

3. Richard Roe is the attorney of Marsha Moe.

4. Richard Roe expects a judgment for the defendant.

These representations are illustrated in Figure 3-4. Notice that by using only one point to represent Richard Roe of fact 3 and Richard Roe of fact 4, we are indicating that they are the same person.

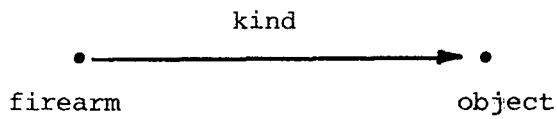
The world of human activity is far more complex than the world of building blocks. Before we can hope to build meaningful representations, there are some important considerations we must explore.

3.11 Things and Relations

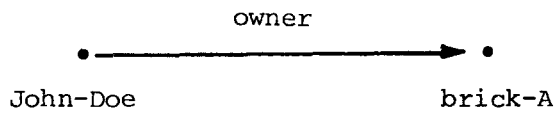
In our model, the basic division into things and relations must not be taken as a description of the world itself. We are not suggesting that the world of human affairs is, in some objective sense, actually composed of Things and Relations. Nor do we propose that things and relations necessarily are basic components of natural epistomology or linguistics. We are following this scheme because it is the simplest one that allows us to explore machine tasks we wish to study. The important roles played by things and relations in human thought and language certainly contribute to usefulness of this model.

We must be careful not to regard the machine's concept of a thing or a relation as anything so semantically rich as our own. To the machine, these are two different kinds of data item, with different structural and

Firearms are a kind of object.



John Doe is the owner of brick A.



Richard Roe is the attorney of Marsha Moe.
Richard Roe expects a judgment for the defendant.

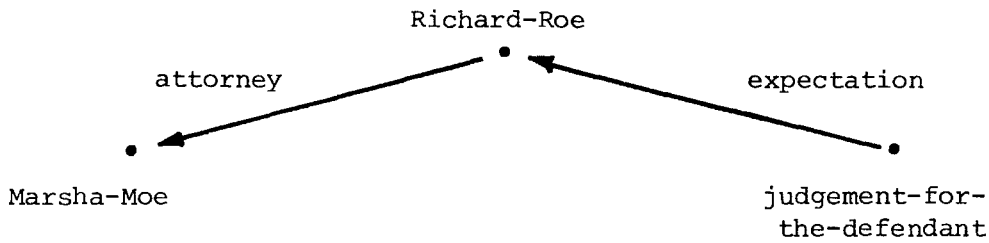


Figure 3-4.

procedural rules. The fact that we choose to label one "thing" and one "relation" is, by itself, irrelevant. The machine knows some differences between things and relations in terms of what it knows about various kinds of things and kinds of relations, but we will see that this knowlege is relatively sparse.

In many cases, it is obvious whether we want to represent something in our world as a thing or as a relation. A brick is easily thought of as a thing; the idea of "taller" clearly is a relation between two things. But often, the choice is not obvious. How should we represent an event, such as John's hitting Richard? Is "hitting" to be considered a thing related to John and Richard, or is it to be considered a kind of relation between them, or is it both? We cannot appeal to the reality of the world to resolve questions like these. We will take an approach that makes our scheme as useful as possible to our task, and one that fits easily into the framework of the machine language PSL that we be using to implement the scheme. (PSL will be described in Chapter 4.)

We will include, as kinds of things, <u>objects</u> (like bricks, people, and judgments), <u>events</u> (like hitting, telling, and civil actions), and <u>values</u> (like red, tall, healthy, and expensive.) A major kind of relation will be the <u>feature-relation</u>, which we use to relate objects, events, and values to each other. For example, the relation <u>near</u> is a spacial relation between two objects; the legal relation <u>owner</u> relates a person to a physical object; the semantic relation <u>agent</u> relates a person or some other thing to an event (John is the agent of the event: John hits

Richard); and the relation health relates a person to a value like healthy. Thus, Richard Roe's being tall and healthy would be represented as shown in figure 3-5.

Our preliminary breakdown of things and relations is illustrated in figure 3-6.

3.12 The Kind Hierarchy

From figure 3-6, we can see the important role that is played by the kind relation. We use it to construct the categorization scheme for the things and the relations with which the system is familiar. We will call this taxonomy the kind hierarchy.

For purposes of this preliminary study, we will depend entirely upon the kind hierarchy for representing the system's knowledge about things and relations in the world. All that the prototype system "knows" about any thing or relation is its position in the kind hierarchy.

We could represent a much richer model of world knowledge by including feature-relations running between the things and relations in the kind hierarchy. For example, all that the prototype system knows about bricks is that they are a kind of building-block. By using feature relations, we could let the system know also that bricks are portable-size in size and heavy in weight. We might also use a relation (perhaps called characteristic) by which we could let the system know which feature-relations are characteristic of which things. For example, we could include the fact that color is a characteristic of persons as well as of

Richard Roe is tall and healthy.

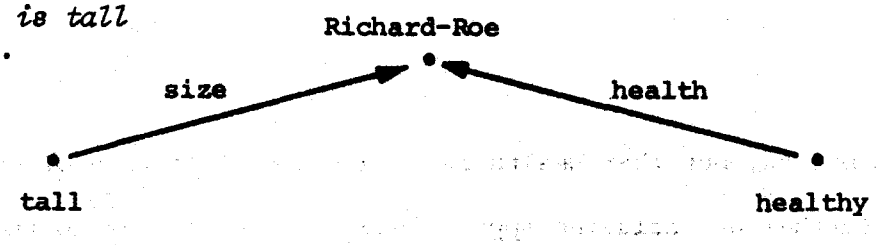
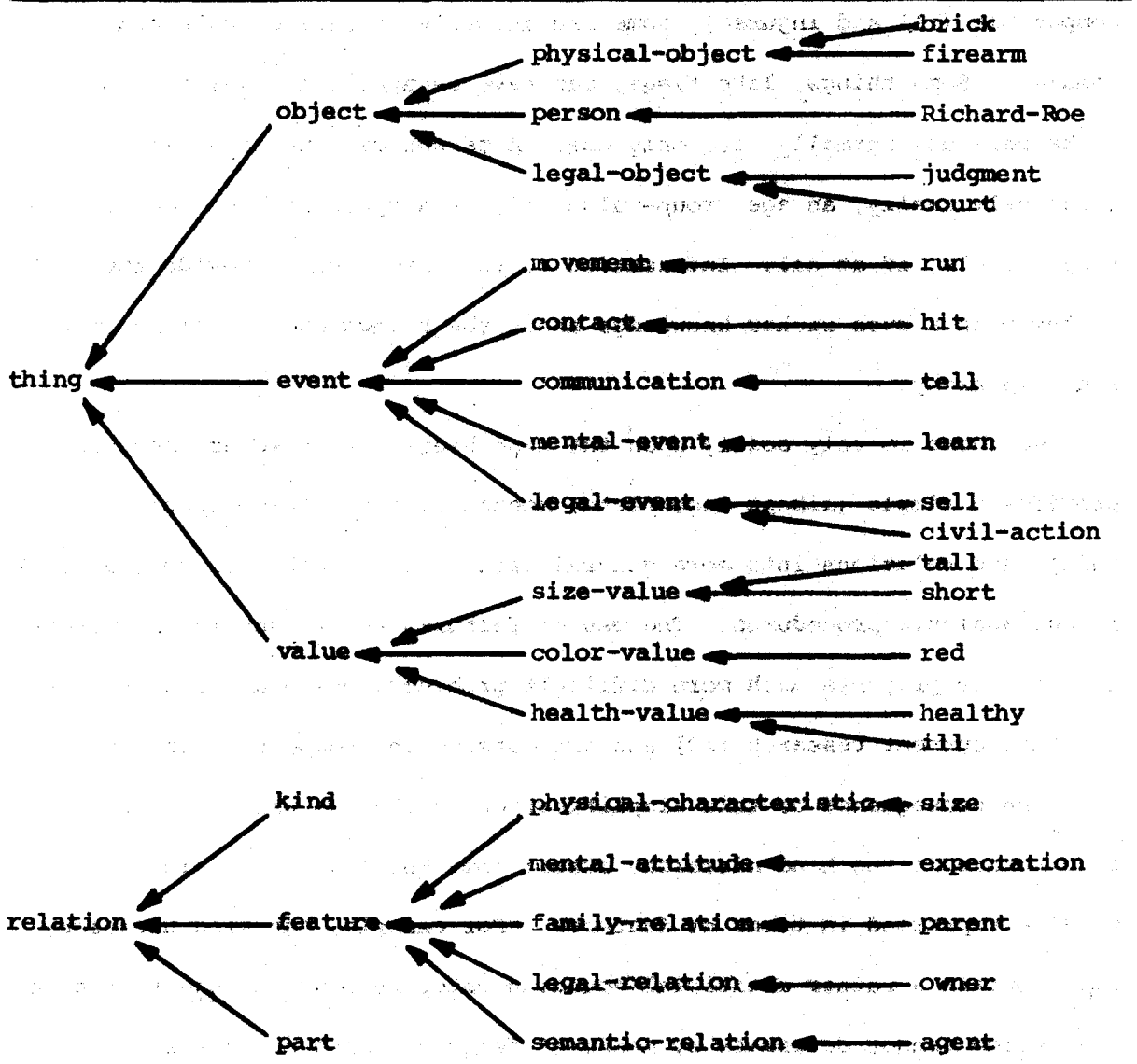


Figure 3-5.



(All arrows represent the kind relation.)

Figure 3-6.

physical objects, but that health is a characteristic only of persons. We could go further by including appropriate restrictions as to the permissible values for feature-relations. Some values are mutually compatible (ill and injured); some are mutually exclusive (male and female). Some things, like flags, can have several color values; some (like persons) normally have only one. A person can change a health-value relatively easily, an age-group-value only at a specific time, and a sex-value rarely, if at all. Information of this kind would provide the system with a much richer knowledge about the things and relations in the kind hierarchy.

We chose to rely solely upon the kind hierarchy, however, because it provides a simple (albeit incomplete) mechanism for fitting specific things and relations into more general categories, which is a central task in our analysis procedures. The use of features and values in the matching process presents much more difficult problems, which are the subject of other current research [29] and are outside the scope of this study.

One consequence of this simplification is that we will have to incorporate in the kind hierarchy itself some knowledge that might otherwise be expressed in terms of features. For example, our kind hierarchy will contain a rather artificial-sounding category <u>movable-object</u> because the system cannot determine by means of features which objects are movable.

Another consequence is that the system cannot make "common-sense" inferences that a human being or a more knowledgeable system might be able to make. For example, when a person fires a rifle, injuring another

person, it normally can be inferred that the first person caused the injury. Our system is not able to make inferences of this kind. (Although, as we will see in Chapter 5, the need to make such inferences sometimes can be bypassed by means of alternative descriptions for the same situation.)

We can make our kind hierarchy far less restrictive by allowing things and relations to belong to more than a single kind category. For example, instead of making an arbitrary decision as to whether a knife ought to be classified as a kind of weapon or as a kind of tool, we can place it in both categories. The use of multiple classifications increases the amount of information that we can include about an item. The examples examined in this study will not make use of multiple classifications, but we anticipate that for larger systems involving many different areas of law, a multiply classified kind hierarchy could eliminate some of the problems caused by the absence of feature information. By classifying a knife both as a kind of weapon and as a kind of tool, we are representing that it has the features of both.

Finally, for the sake of completeness, we will represent the very top of the kind hierarchy by the category <u>something</u>. Thus, things and relations are kinds of something.

3.13 Kinds and Instances

Looking once again at figure 3-6, we see that Richard Roe is represented as a kind of person in the same way that a brick is represented as a kind of physical object. To some extent this is sensible. In one case, we are saying that within the set of physical objects, bricks constitute a subset presumably with certain distinguishing characteristics (features). Similarly, Richard Roe, within the set of persons, can be said to belong to a one-member subset distinguishable by those characteristics that form Richard Roe's identity. Treating both cases in the same way, however, overlooks an important difference. It may be useful to have a category of brick-like objects, but of what use is a category of persons with Richard Roe's identity?

Furthermore, it is necessary to have an explicit method for differentiating between information about whole categories, like bricks and persons, and information about certain bricks and persons.

For this purpose, we introduce the instance relation. We represent Richard Roe as an instance of a person, rather than as a kind of person. The instance relation is identical to the kind relation except it does not imply a distinguishable category. On the other hand, while an instance indicates an individual rather than a category, it can be an abstract individual. We might use Richard Roe as a token, the name of a role rather than of the actor who plays it.

Consider again the house structure described by Winston (figure 3-3). The bottom part of the house, which is given the name B, is represented as a kind of brick. In our model, we would represent it as an instance of a brick. There is nothing about its inclusion in the house structure that makes it different from any other brick; therefore, it does not represent a distinguishable category of bricks. At the same time, it does not represent some particular brick in the real world, but a brick in the abstract. Its purpose is to serve as a token of a brick--which token is to be associated with the things and relations appropriate to defining a house.

Sometimes we will encounter distinguishable kind categories with single members. In these cases, the kind relation merges with the instance relation.

Care must be taken regarding the scope of the name we give to an instance. That is, we must be able to define the structural boundaries within which the same instance-name is understood to refer to the same instance. This problem will be examined in section 4.2.

3.14 Semantic Relations

The user of this prototype system does not communicate with the system in any natural language. The user expresses things in the PSL language, which we will see shortly. One objective of those who are currently developing the OWL system is the automatic translation between

languages like PSL and simple English. For this reason, among others, PSL is oriented around a linguistic model known as case grammar [30].

Simply described, the case-grammar model treats a sentence as an action (corresponding loosely to the verb) with a set of associated cases, like the agent and the object of the action. (The agent corresponds loosely to the subject of an active-voice sentence, the object corresponds loosely to the direct object.) Some other cases are: co-agent, instrument, source, method, destination, purpose, cause, result.

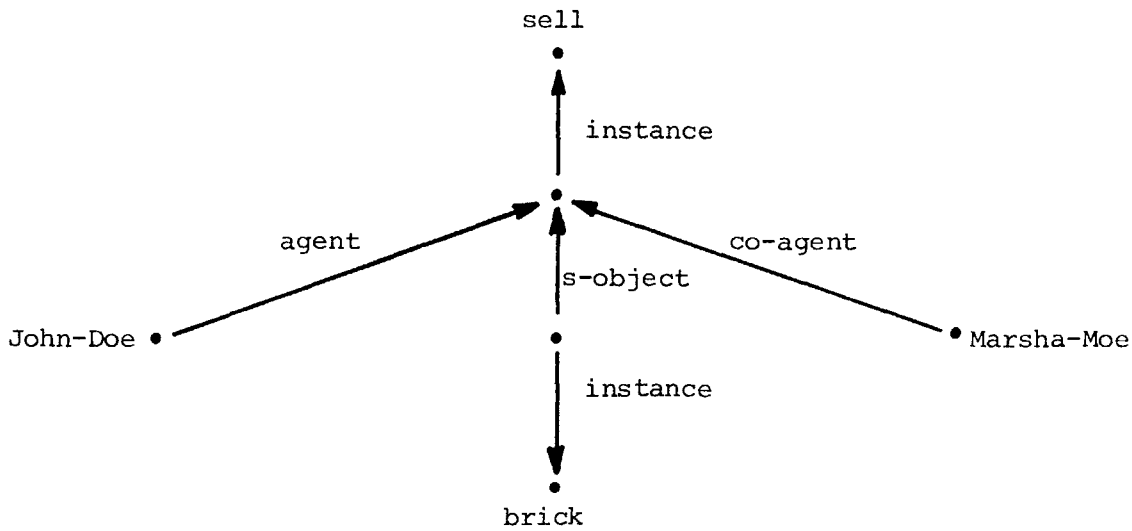
We are not interested directly in the case grammar as a linguistic model, and we will not explore the details of the different cases. The basic idea of actions and semantic cases, however, is useful to our model, especially because these can be expressed easily in terms of things and relations.

The use of semantic cases will be illustrated with the following examples:

1. John Doe sells a brick to Marsha Moe.
2. Joe hits Fred in the shoulder with his fist.

In the first example, we can represent the action as an instance of the event sell. John Doe is the agent of the action, the brick is the object of the sale, and Marsha Moe is the co-agent. In the second example, the action is an instance of the action hit, the agent is Joe, and the object is Fred, the instrument is Joe's fist, and the destination of the hit is Fred's shoulder. Representations for these examples are shown in figure 3-7. (We use the semantic relation s-object to avoid confusion with object, which we have defined as a kind of thing.)

John Doe sells a brick to Marsha Moe.



Joe hits Fred in the shoulder with his fist.

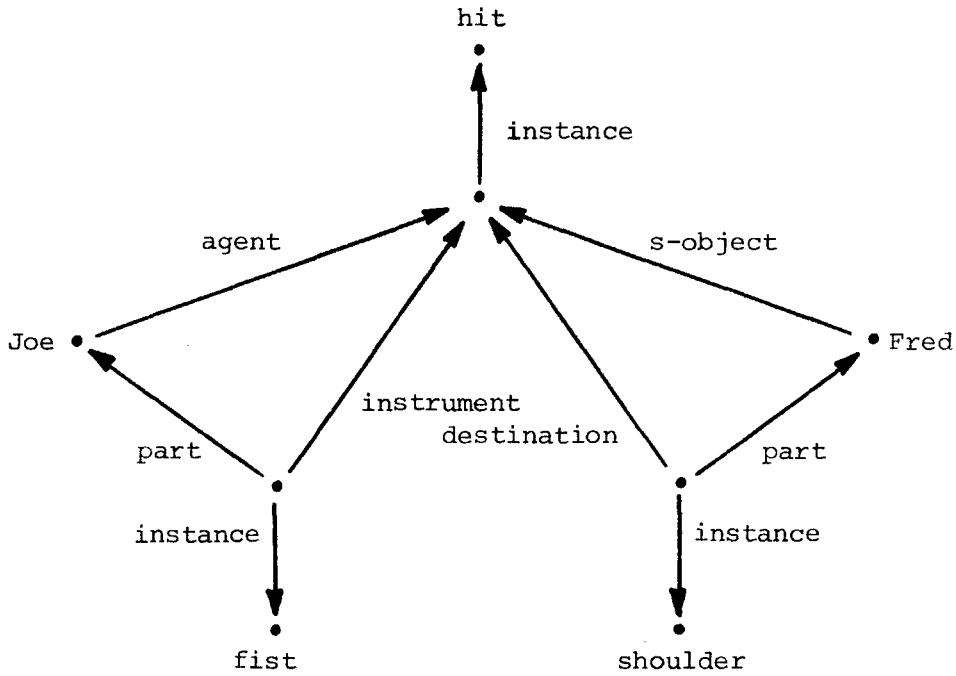


Figure 3-7.

3.15 The Absence of Temporal Relations

One important aspect of human activity is the temporal sequence and concurrence of events, and the relationship between events and the changes in states of affairs. We have seen, for example, how von Wright (page 41 uses representations for states and transitions to model activities and norms. In order to keep the examples in this study relatively simple, we have left out the temporal dimension. We will refer to all events as if they are taking place in a fictitious "present tense." For the legal doctrines that we will examine, this simplification does not result in much distortion. There are very few issues in battery or assault that turn on timing or sequence.

Of course, there are many areas of legal doctrine where time considerations are critical. For example, in the creation of a contract by mail, the sequence of the posting and receipt of offers, acceptances, and withdrawals is most important.

Timing and sequence could be included in our model in a manner precisely analogous to that used for spacial location. By using temporal relations (like <u>before</u> and <u>after</u>), we can relate events according to their proper sequence. A relation like <u>at-time</u> can be used to relate events to specific times, which would be included as time-values.

Some of the more difficult problems of sequence, concurrence, and state changes, in relation to the modeling of legal doctrine, are the subject of other current research [31].

3.2 Facts and Situations

3.21 Facts

So far, we have been discussing representations of simple things and relations. These correspond to data items in our system, and we give these items names that suggest to us the things and relations they represent. The relationship formed by connecting two such things and a relation also represents something in the real world, and it will be very useful to recognize and to represent these relationships explicitly. We will call a thing-relation-thing a fact. Just like our definitions of thing and relation, this definition is stipulative. It is not meant to describe "correctly" the way we use the word fact in common parlance, in legalese, or in any other technical context. A fact behaves just like a a thing, in the sense that it can bear a relation to another thing. Therefore we will enter a fact in the kind hierarchy as a kind of thing.

Representing a fact as a kind of thing allows us to construct composite facts within which one (or both) of the two things is itself a fact. Consider the fact expressed by the sentence, "Marsha Moe believes that Richard Roe is her attorney." We represent this fact by relating Marsha-Moe, an instance of a person, by the relation belief to a fact comprising Marsha-Moe (again), the relation attorney, and the person Richard-Roe. This representation is illustrated in figure 3-8. The box in the figure represents the internal fact, and it contains the end-points of the arrow representing the internal relation, attorney. The belief relation is drawn as an arrow from this box to Marsha-Moe. Notice that in

It is the belief of Marsha Moe that Richard Roe is her attorney.

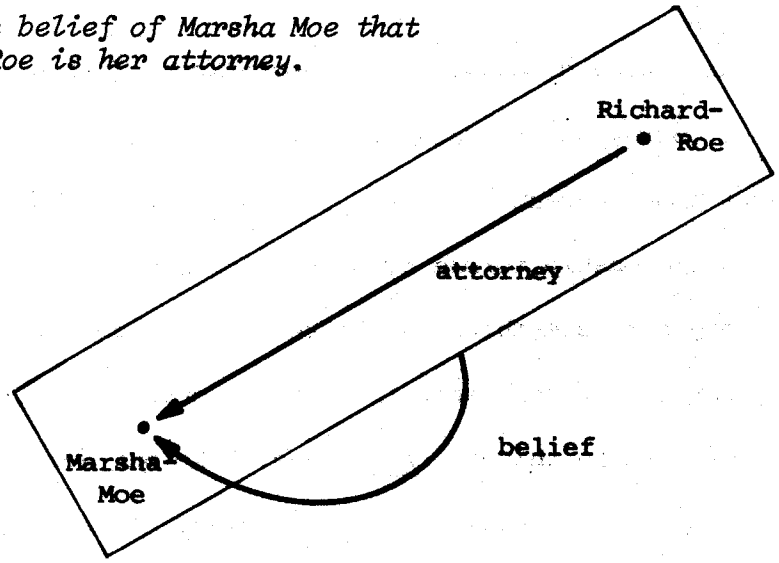


Figure 3-8.

In this situation, Marsha Moe believes that Richard Roe is her attorney.

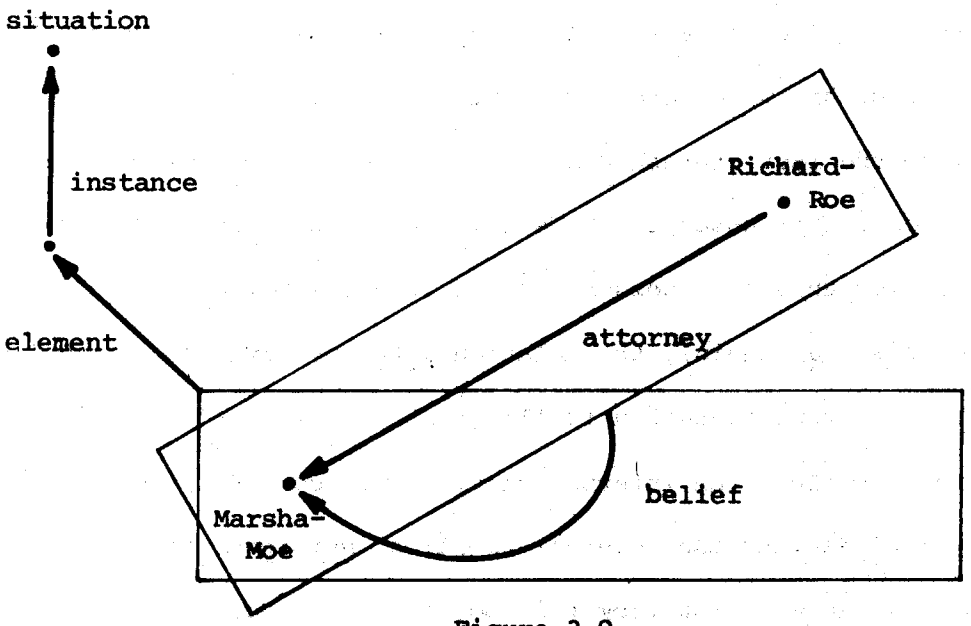


Figure 3-9.

this example, the internal fact corresponds to a subordinate clause in the natural-language expression of the fact. This is often, but not always, the case.

3.22 Situations

Next, we wish to recognize and to represent explicitly sets of facts, which, taken together, constitute a collective state of affairs. We might think of this as a story or as a set of circumstances. We will call such a set a situation. As we did with facts, we will represent a situation as a kind of thing that may be related to other things. We will call the relation between a situation and each of the facts that belong to that situation the element relation (an-element-of). The element relation behaves something like the part relation in Winston's structural representations (see figure 3-3).

Now consider again the fact: Marsha Moe believes that Richard Roe is her attorney (figure 3-8). We would represent the situation in which this was the only fact as is shown in figure 3-9. Note that Richard Roe's actually being (or not being) her attorney is not an element of this situation. If we wished to represent this circumstance as well, we would have to include a second element relationship explicitly, as is shown in figure 3-10. Of course, we might also wish to represent a situation in which Marsha is "wrong" in her belief; perhaps her attorney "really"--that is, according to this situation--is John Doe. This set of circumstances would be represented as is shown in figure 3-11.

In this situation, Marsha Moe believes that Richard Roe is her attorney, and Richard Roe is indeed her attorney.

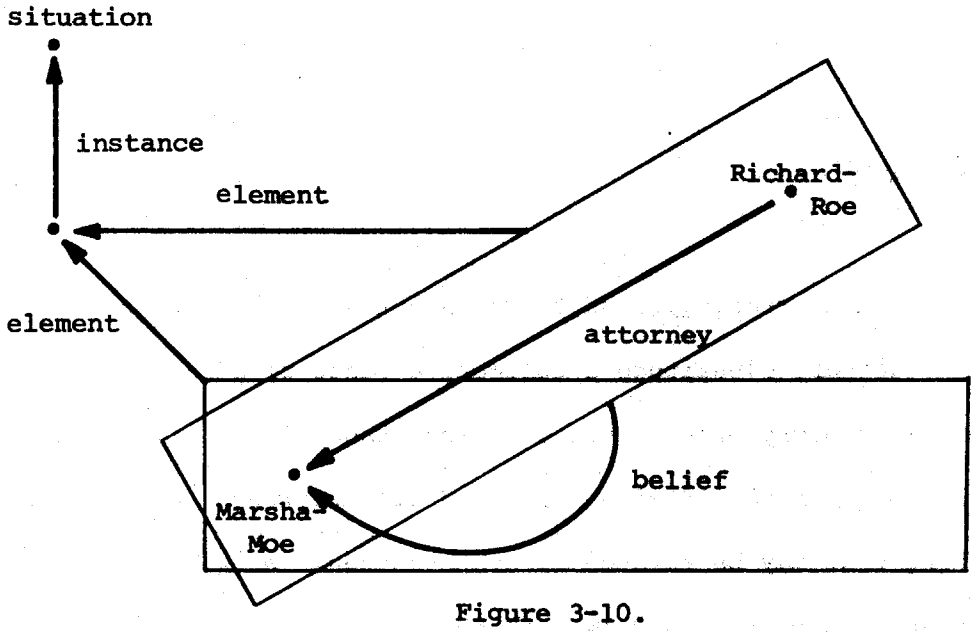


Figure 3-10.

In this situation, Marsha Moe believes that Richard Roe is her attorney, but her attorney is actually John Doe.

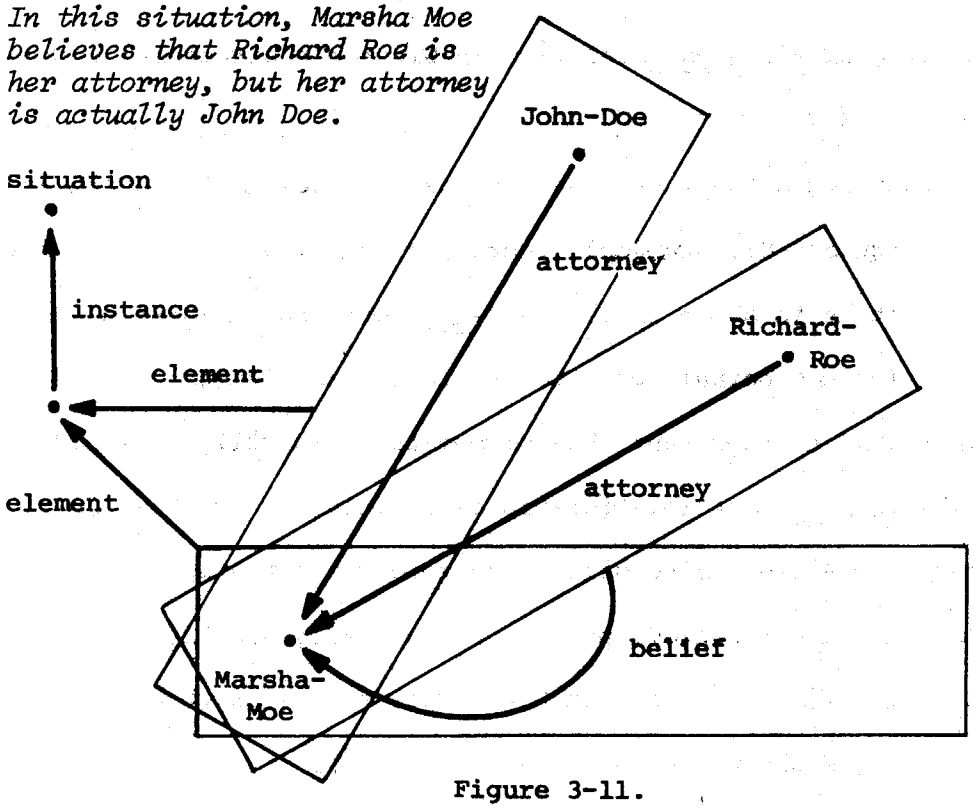


Figure 3-11.

There is one particularly important kind of situation used by the system. This is a situation representing the facts that are presented to the system as the hypothetical case to be analyzed. We call this kind of situation the facts-at-hand. In this prototype, there is only one such situation under consideration at any one time, so the facts-at-hand can be thought of as an instance of a situation as well as a kind of situation. (More sophisticated versions of this system might include several hypothetical cases at once, in which case there could be serveral instances of this kind of situation.) Each of the facts presented by the user to the system is understood to be an element of facts-at-hand.

Instances of situations can be used, like facts, as things within facts, allowing us to assemble more complex fact structures. Let us consider an example: The user tells the system of a set of circumstances in which Marsha Moe tells her attorney, Richard Roe, that John Doe sold her a brick.

This can be represented as is shown in figure 3-12. In the representation, the situation facts-at-hand includes (1) the fact that Richard Roe is Marsha's attorney, and (2) the facts constituting her telling something to Roe. What she tells Roe is the semantic object of this event, and is represented by a subordinate situation whose elements are the facts constituting the selling event. (As indicated in section 3.15, we are ignoring the temporal sequence of events.)

In the facts at hand, Marsha Moe tells her attorney, Richard Roe, that John Doe sold her a brick.

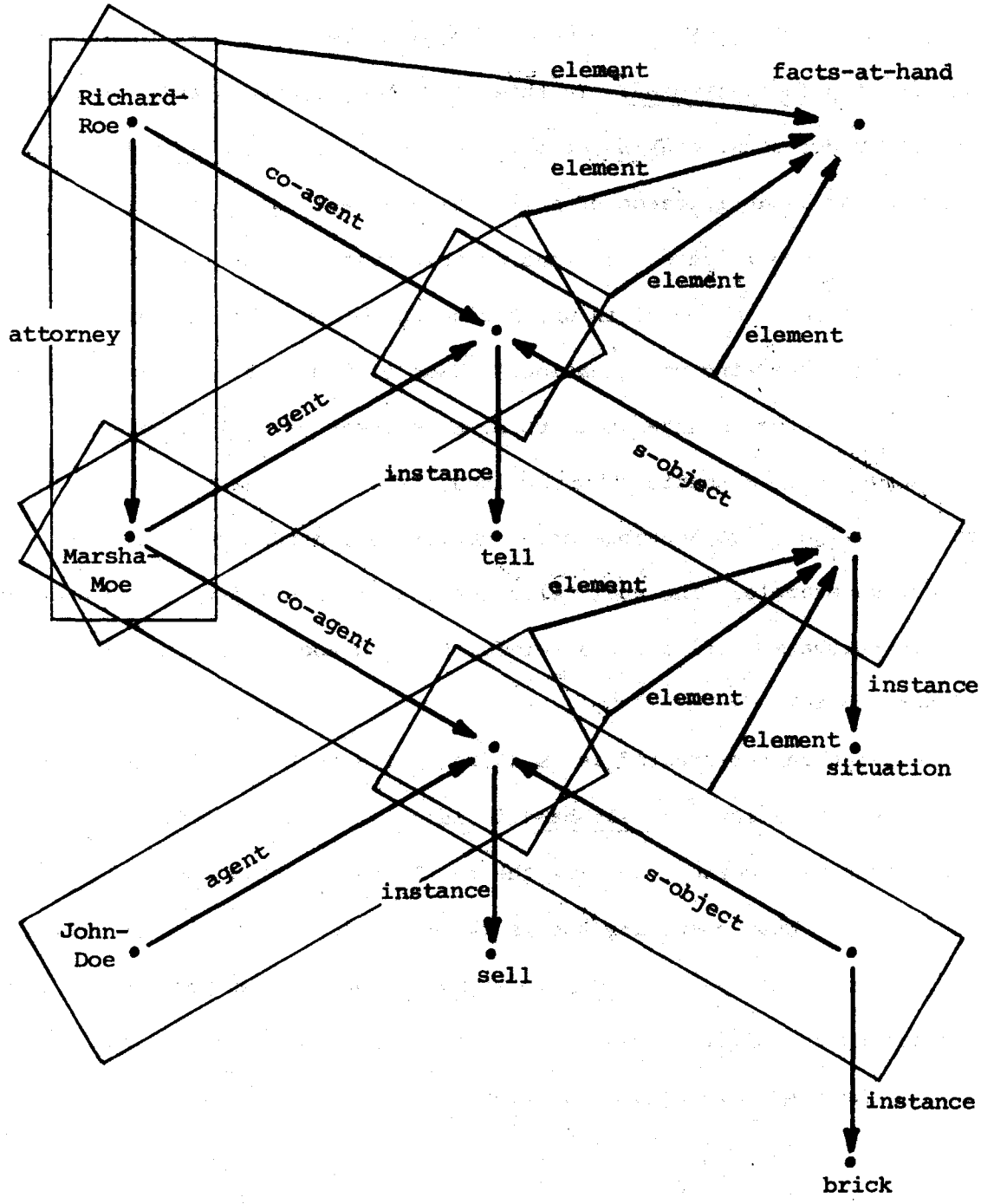


Figure 3-12.

3.23 The Situational Structure of the System's Knowledge

As the examples in the last section demonstrate, facts and situations generally do not stand alone as direct pieces of the system's knowlege. Instead, they often appear as components or elements of other facts and situations. It is this aspect of our architecture that allows us to represent what people believe, tell, expect, et cetera, in contexts removed from what we wish the system itself to know or to believe. In the example of figure 3-11, we saw a situation in which a person believed a fact which, in that situational context, was not correct. Similarly, in the example of figure 3-12, the fact that a brick was sold is far removed from the system's direct knowledge. What the system knows in that example is that the user hypothesizes that someone says that a brick was sold. Thus we see that there is a hierarchy of situational contexts in the system. The organization of the system's knowledge about the world and about the law is based on this hierarchy.

At the top of the situational hierarchy are the facts that the system knows directly. The main component of the system's direct knowledge is the kind hierarchy. If we were to include in the prototype system the other kinds of knowledge discussed in section 3.12, this, too, would be part of the system's direct knowledge, Also known directly are the machine procedures, to be described in Chapter 6.

All of the facts and procedures that are known directly could be represented as <u>elements</u> of a top-level situation representing the system's

overall knowledge. We will not need to represent this top-level situation explicitly. Facts that are not subordinate to other facts and situations will be understood to be elements of this implicit, top-level situation.

The situation facts-at-hand is one level lower in this hierarchy. The elements of facts-at-hand are known indirectly by the system as facts and situations that the user is presenting hypothetically.

The system's knowledge of legal doctrine is also at the second level in the context hierarchy. This permits us to attribute legal doctrine to sources of authority. From the kind hierarchy, the system knows directly about the existence of various legal authorities such as judicial decisions, statutes, treatises, encyclopedias, et cetera. In the prototype system, primary legal authority will be represented only for case law, and will take the form of small factual examples and holdings. More general doctrine will be represented in the form of a fictitious secondary source called Corpus Juris Mechanicum. The system contains legal knowledge in the sense that it knows what these authorities assert about the law.

The situational structure of the system's knowledge is illustrated in figure 3-13.

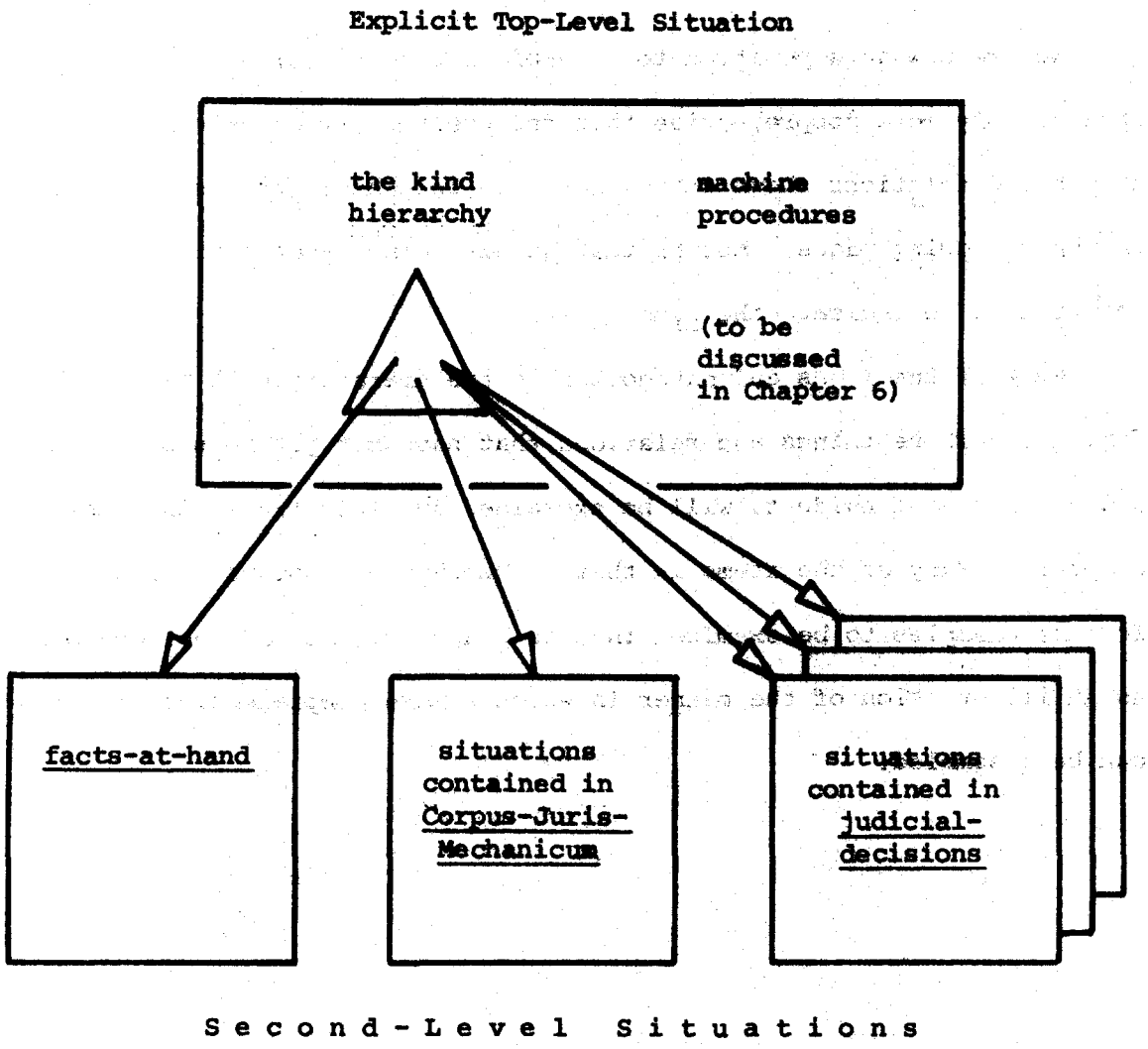

Explicit Top-Level Situation
the kind hierarchy
machine procedures
(to be discussed in Chapter 6)
facts-at-hand
situations contained in Corpus-Juris-Mechanicum
situations contained in judicial-decisions
Second-Level Situations

Figure 3-13.

3.3 A More Comprehensive Kind Hierarchy

We are now in a position to assemble a kind hierarchy that is considerably more comprehensive than the preliminary breakdown of things and relations shown in figure 3-6. This hierarchy is presented on the following pages. Notice that we are using typographical indentation to indicate the kind relation.

Some of the items and categories in the hierarchy will not yet be familiar. Those things and relations that have not yet been discussed, and are not self-evident, will be explained and illustrated in later chapters. Many of the items in this hierarchy will not be needed or used for the examples to be examined in this study. These are included only as an illustration of the manner in which a more comprehensive hierarchy can be assembled.

The Kind Hierarchy

List A (top level)

Something

- thing
  - object (see list B)
  - event (see list E)
  - value (see list D)
  - fact
  - situation
    - facts-at-hand
- relation
  - classification-relation
    - kind
    - instance
    - type
    - counter-type
    - example
    - counter-example
  - composition-relation
    - part
    - element
    - counter-element
    - facts
  - feature-relation (see list C)
  - instantiation-relation (see list H)
  - logical relation (see list F)
  - arithemetic-relation (see list G)
  - execute

List B

object

- physical-object
  - stationary-object
    - place
    - building
    - tree
    - mountain
    - boulder
    - room
      - jailcell
    - room-part
      - wall
      - floor
    - hallway
    - street
    - ground
    - sky
    - city
    - state
  - animate-object
    - animal
      - house-pet
        - dog
        - cat
      - wild-animal
    - tornado
    - meteorite
  - movable-object
    - tool
      - hammer
      - wrench
      - knife
    - furniture
      - chair
      - table
      - desk
      - sofa
      - bed

movable-object (continued)

- access-objects
  - door
  - window
  - lock
- article-of-clothing
  - hat
  - coat
  - dress
  - jacket
  - troussers
  - shoe
- personal-accessory
  - cane
  - handbag
  - briefcase
  - umbrella
- weapon
  - firearm
    - pistol
    - rifle
    - shotgun
  - knife
  - axe-handle
  - stone
- sports-implement
  - golf-club
  - baseball-bat
  - hockey-stick
- building-block
  - cube
  - brick
  - wedge
  - pyramid
- vehicle
  - automobile
  - bus
  - glider

physical-object (continued)

- physiological-object (human)
  - anatomical-object
    - head
    - eye
      - left-eye
      - right-eye
    - nose
    - mouth
    - lips
    - neck
    - shoulder
    - chest
    - abdomen
    - back
    - limb
      - arm
      - leg
    - extremity
      - hand
      - fist
      - foot
  - skeletal-object
    - skull
    - spinal-column
    - rib-cage
    - rib
    - clavicle
    - humerus
    - radius
    - ulna
    - (et cetera)
  - organ
    - brain
    - heart
    - liver
    - lung
    - stomach
    - (et cetera)

Object (continued)

- legal-object
  - legal-institution
    - trial-court
    - appellate-court
    - legislature
    - administrative-agency
  - legal-judgment
    - judgment-for-defendant
    - judgment-against-defendant
  - legal-doctrine
    - criminal-law
      - law-of-criminal-battery
      - law-of-criminal-assault
    - law-of-civil-actions
      - in-contract
      - in-tort
        - negligence
        - intentional-tort
          - interference-with-property
            - tresspass-to-land
          - interference-with-person
            - assualt
            - battery
            - false-imprisonment
            - invasion-of-privacy
  - legal-authority
    - primary-authority
      - judicial-decision
      - statute
      - constitution
    - secondary-authority
      - Prosser-on-Torts
      - American-Jurisprudence
      - Corpus-Juris-Secundum
      - Corpus-Juris-Mechanicum

Object (continued)

- person
  - user
  - doctor
    - surgeon
  - lawyer
  - garage-mechanic

- legal-person
  - natural-person
    - reasonable-man
  - corporation
  - partnership

- judicial-person
  - plaintiff
  - prosecution
  - defendant
  - witness
  - adjudicator
    - trial-judge
    - appellate-judge
  - fact-finder
    - jury
    - trial-judge-without-jury
    - appointed-master

List C

Feature-Relation

- physical-characteristic
  - size
  - weight
  - color
  - spatial-relation
    - contact-relation
      - on
      - attached-to
      - held-by
    - distance-relation
      - near
      - in-front-of
      - to-the-left-of
      - far
  - orientation
  - loadedness
  - count
  - health
  - monetary-worth
  - sex
  - age-group
- mental-attitude
  - knowledge
  - belief
  - perception
  - expectation
  - apprehension
  - willingness
  - consent
  - assumption
- family-relation
  - parent
    - mother
    - father
  - sibling
    - brother
    - sister
  - spouse
    - husband
    - wife

feature-relation (continued)

- semantic-relation
  - agent
  - s-object
  - co-agent
  - purpose
  - precondition
  - source
  - method
  - destination
  - result
  - instrument
  - beneficiary
  - cause
- legal-relation
  - owner
  - possessor
  - attorney
  - right
  - duty
  - liability
  - immunity
  - legal-consequence
  - assertion
  - holding

List D

Value

size-value
- short
- small
- portable-size
- large
- tall
- a-meter

weight-value
- light
- portable-weight
- heavy
- a-gram

color-value
- red
- orange
- yellow
- green
- blue
- purple
- black
- white

orientation-value
- lying-down
- sitting
- standing
- open
- closed

loadedness-value
- loaded
- unloaded

value (continued)

count-value
- one
- two
- three
- .
- .
- .
- few
- many

health-value
- healthy
- fair-health
- ill
- injured
- disabled

sex-value
- male
- female

age-group-value
- minor
- adult

monetary-worth-value
- wealthy
- moderate-income
- poor
- cheap
- reasonable
- expensive
- a-dollar

List E

Event

- movement-event
  - move
    - move-self
      - walk
      - run
      - leave
      - enter
      - jump
      - drive-in-car
    - move-object
      - push
      - pull
      - slide
      - raise
      - lower
      - carry
      - drop
      - throw
      - send
      - point
  - not-move
    - stay-there
    - confine
- orientation-event
  - re-orient
    - sit-down
    - stand-up
    - fall-down
    - get-up
    - open-up
    - close-up
  - keep-oriented
    - keep-lying
    - keep-sitting
    - keep-standing
    - keep-open
    - keep-closed

event (continued)

- contact-event
  - touch
  - grab
  - strike
    - hit
    - kick
    - slap
    - punch
    - knock-off
  - kiss
- health-event
  - get-worse
    - injure
    - disable
    - become-ill
  - get-better
    - become-well
    - recover
    - operation
      - appendectomy
  - no-change
    - stay-well
    - stay-ill
    - stay-injured
- communication-event
  - communicate
    - tell
    - manifest
  - perceive
    - see
    - hear
- athletic-competition
  - boxing-match
  - tennis-match
  - baseball-game
  - hockey-game
  - round-of-golf

Event (continued)

- mental-event
  - start-knowing (learn)
  - stop-knowing (forget)
  - start-believing
  - stop-believing
  - (et cetera)

- legal-event
  - legal-action
    - criminal-proceeding
    - civil-action
      - action-in-contract
      - action-in-tort
    - appellate-action
  - offer
  - accept
  - purchase
  - sell

- machine-event
  - learn
  - insert
  - display
  - receive
  - listen-to-facts
  - instantiate
  - instantiate-by-elements
  - instantiate-by-types
  - instantiate-by-examples
  - discuss-analysis

List F

Logical (Boolean)

not
and
or
xor
implies
equivalent
is

List G

Arithmetic

plus
minus
times
equal
less-than
greater-than

List H

Instantiation-relation

inst-element
inst-type
inst-example
inst-syllogism
inst-analogy
inst-assume
c-inst-element
c-inst-type
c-inst-example
c-inst-syllogism
c-inst-analogy
c-inst-assume

Instantiation-relation (continued)

n-inst-element
n-inst-type
n-inst-example
n-inst-fact

Chapter 4 Implementation in PSL

In Chapter 3, we presented the model that we will use to represent specific situations of fact as well as the more generalized situations contained in legal doctrine. For purposes of introducing the model, and for demonstrating some simple examples, we used a graphical notation that corresponds to machine representations that we would want to create in the computer. For more complicated situations, however, the use of the graphical notation becomes rather cumbersome. Therefore, from this point on, we will express our representations directly in a machine comprehendible language called PSL (Preliminary Study Language). A computer system called OWL is able to translate such expressions into machine representation. OWL is a modeling system currently being developed by William A. Martin and others at the Automatic Programming Division of M.I.T.'s Project MAC [32]. OWL comprises a data structure, a set of machine procedures, and a programming language that are ideally suited to embody the fundamental features of our model.

4.1 Representing Facts and Situations in PSL

The basic PSL statement comprises a relation and two things, corresponding to what we have been calling a fact. Each such statement is written enclosed in parentheses:

(rel th1 th2),

where rel stands for the name of the relation, and where th1 and th2 stand for the names of the things. Recall that in our graphical notation, we used an arrow to denote the relation between two things. In the PSL notation, th1 corresponds to the thing at the head of the arrow, and th2 corresponds to the thing at the tail of the arrow. Here are two examples of facts expressed in PSL:

(kind building-block brick)	*A brick is a kind of building block.*
(attorney Marsha-Moe Richard-Roe)	*The attorney of Marsha Moe is Richard Roe.*

To make our examples easier to follow, we will include brief English translations to the right of the PSL statements. Based on current research in natural language processing, it is reasonable to expect that a system like OWL will be able to translaate statements like those on the right into machine statements like those on the left.

The OWL system is written in the LISP programming language, which is based on data structures that take the form of lists. The data structure corresponding to a statement of fact in PSL is a list of data cells representing rel, th1, and th2. The data cells are connected by linkages called pointers. In addition, the OWL system creates linkages called back-pointers running from each rel, th1, and th2 to every fact in which that thing or relation appears. As we shall see in Chapter 6, these back-pointers provide a handy index for finding quickly those facts that

contain a given thing or relation.

The kind hierarchy can be expressed in PSL as a series of statements of the following form:

(kind something thing)	*A thing is a kind of something.*
(kind something relation)	*A relation is a kind of something.*
(kind thing object)	*An object is a kind of thing.*
(kind thing event)	*An event is a kind of thing.*
(kind thing value)	*A value is a kind of thing.*
.	.
.	.
.	.
(kind furniture chair)	*A chair is a kind of furniture.*
(kind furniture table)	*A table is a kind of furniture.*
.	.
.	.
.	.

To make repetitive lists like this more convenient for our discussion, we will adopt an abbreviated notation. We will allow either of the <u>th</u>'s in a fact to comprise a series of things, separated by commas:

(kind thing object,event,value,fact,situation).

This expression will be understood to be equivalent to several separate facts, one for each thing in the series:

(kind thing object)	*An object is a kind of thing.*
(kind thing event)	.
(kind thing value)	.
(kind thing fact)	.
(kind thing situation)	.

Compound facts, in which one (or both) of the things is itself a fact, are represented simply as compound lists in which one (or both) of the th's is itself a list. The fact that Marsha Moe believes that Richard Roe is her attorney (figure 3-8) is expressed:

(belief Marsha-Moe	*It is the belief of Marsha Moe that*
(attorney Marsha-Moe	*the attorney of Marsha Moe is*
Richard-Roe))	*Richard Roe.*

Situations and their elements also are expressed quite simply (especially with the comma notation). Consider the example from figure 3-11, in which Marsha Moe believes that Richard is her attorney, when her attorney is actually John Doe:

(instance situation s1)	*s1 is a situation*
(element s1	*in which*
(attorney Marsha-Moe	*the attorney of Marsha Moe is*
John-Doe),	*John Doe, and*
(belief Marsha-Moe	*it is the belief of Marsha Moe that*
(attorney Marsha-Moe	*the attorney of Marsha Moe is*
Richard-Roe)))	*Richard Roe.*

Because we will make frequent use of instances of things, it will be convenient to introduce another abbreviation. To represent the fact that Joe, an instance of a person, is the agent of an instance (call it h1) of a hitting event, we could express three separate facts:

(instance person Joe)	*Joe is a person.*
(instance hit h1)	*h1 is a hit.*
(agent h1 Joe)	*Joe is the agent of h1.*

Instead, we will combine these three facts into a single PSL statement by using colons. The notation X:x is used to indicate that x is an instance of X. Thus, we can write:

(agent hit:h1 person:Joe),

which might be expressed in English as, "Joe, who is a person, is the agent of h1, which is a hit."

An individual fact is always understood to be an instance of the category fact. Sometimes it is convenient to give a name to an individual fact, just as other instances are given names. We will use the colon notation for this purpose as well. For example, let us add to the situation s1 in the example on page 94 the fact that John Doe believes (correctly) that he is Marsha Moe's attorney. The elements of this situation can be expressed thus:

(element s1	*In s1,*
(attorney Marsha-Moe	*the attorney of Marsha Moe is*
John-Doe):f1,	*John Doe, and*
(belief Marsha-Moe	*it is the belief of Marsha Moe that*
(attorney Marsha Moe	*the attorney of Marsha Moe is*
Richard-Roe)),	*Richard Roe, and*
(belief John-Doe f1)).	*it is the belief of John Doe that the attorney of Marsha Moe is John Doe.*

In this way, the PSL expression (attorney Marsha-Moe Richard-Roe) needs to be written out only once.

4.2 The Scope of the Instance Names

In the graphical representations used in Chapter 3, when we wanted to indicate the involvement of the same instance of something in several facts, we represented that instance with a single point (or rectangle). We need a similar means to indicate when the same instance name, appearing in several PSL statements, is meant to indicate the same instance (i.e., the same data-item).

We could adopt the convention that instance names are global in scope. This would mean that the same instance name always refers to the same instance. It will be more convenient, however, if the names of instances can be given smaller, more localized scopes. This permits us to use the same names (like John Doe) to represent different instances in different contexts. One advantage of using names with localized scopes is that it eliminates the need to check the entire knowledge structure every time we name a new instance, in order to make sure that no other instance has been given the same name. In order to achieve this flexibility, we must establish a clear rule for determining the <u>scope</u> of an instance name. The scope is the contextual boundary within which the same instance name is meant to represent the same instance.

An obvious candidate for a contextual boundary of this kind is the situation. It seems sensible to adopt a rule like: Within any one situation, and only within that situation, the same instance name refers to the same instance. However, this rule does not make clear whether names appearing in a subordinate situation are to be considered as appearing "within" the superior situation. This is often desirable because many of the instances in a subordinate situation are the same as those in the superior situation. This is not universally desirable, however, because it implies that all scopes are global. (All situations in the system are subordinate to the implicit, top-level situation).

To escape this dilemma, we introduce the notion of declaring an instance name. The name of an instance is said to be declared in the situation in which its instance relationship appears, whether the appearance is explicit, like:

(instance person Joe),

or implicit, as in:

(agent hit:h1 person:Joe).

The appearance of an instance relationship (or a colon) declares an instance name in a situation, whether or not that relationship (fact) is actually an element of the situation. However, when an instance relationship appears in a subordinate situation, it is decleared only in that situation, not in the superior situation. An instance declared in a superior situation is recognized by the same name in all subordinate situations, except where that name has been redeclared in a subordinate

situation. Finally, when there is need to refer to an instance not otherwise recognized in a particular situation, this is done by using the name of a situation wherein the instance is recognized, followed by a slash, followed by the instance name. Thus, if the name Joe were recognized in situation s1 but not in situation s2, we could refer to the instance named Joe from within s2 by using the symbol s1/Joe.

Readers who are not familiar with the structure of programming languages need not worry if they do not follow this discussion of scope and declarations. Programmers, on the other hand, will recognize these rules as conventions for declaring and binding local variables.

4.3 Summary of PSL Notation

The following is a summary of the conventions that we will use for representing facts and situations in the PSL language.

1. A fact is represented by a PSL statement of the form:

(rel th1 th2),

where rel is a relation that runs from a thing th2 to another thing th1.

2. Within the statement of a fact, th1 or th2 (or both) may itself be the representation of a fact, for example:

(relA th1 (relB th3 th4)).

3. Separate statements of fact that differ only in regard to th1 or th2 (but not both) can be represented by a single fact statement by using the following comma notation:

(rel th1 th2,th3,. . . ,thN),

which is understood to be equivalent to:

(rel th1 th2)

(rel th1 th3)

.

.

.

(rel th1 thN).

This pluralism applies at the highest level of list structure, so that

(relA th1 (relB th3 th4,th5))

is equivalent to:

(relA th1 (relB th3 th4))

(relA th1 (relB th3 th5)).

4. A fact involving the <u>instance</u> relation can be expressed implicitly within another fact by using the following colon notation:

(rel th1 th2:instance-name),

which is understood to be equivalent to:

(instance th2 instance-name)

(rel th1 instance-name).

5. A fact (which is always understood to be an instance of the category <u>fact</u>) can be given an instance-name by using the same colon notation:

(rel th1 rel2):instance-name.

6. An instance name is said to be <u>declared</u> in the situation in which the name appears in conjunction either with the <u>instance</u> relation or with the colon notation. An instance name is recognized as a reference to the same instance within all of the facts and situations that are subordinate to the situation in which the name is declared, except where the the same name has been redeclared in a subordinate situation.

7. An instance whose name has been declared in one situation can be referred to from within a situation where that name is not recognized by using the following slash notation:

```
(element  situation:s1
     (instance  th1  instance-name))
(element  situation:s2
     (rel  th2  s1/instance-name)).
```

I.e., the symbol s1/instance-name is understood to mean, "the instance that is called instance-name within situation s1."

4.4 A More Complicated Example

As an illustration of the conventions presented in the last three sections, let us consider again the example represented graphically in figure 3-12. Recall that the user has hypothesized to the system a set of circumstnaces in which Marsha Moe tells her attorney, Richard Roe, that John Doe sold her a brick. In PSL notation, this would be represented as follows:

(element facts-at-hand	*In the facts at hand,*
(attorney person:Marsha-Moe	*the attorney of Marsha Moe is*
person:Richard-Roe),	*Richard Roe, and*
(agent tell:t1 Marsha-Moe),	*Marsha Moe tells*
(co-agent t1 Richard-Roe),	*Richard Roe*
(s-object t1 situation:s1))	*that*
(element facts-at-hand/s1	
(agent sell:s1	*John Doe sells*
person:John-Doe),	
(co-agent s1 Marsha-Moe),	*to Marsha Moe*
(s-object s1 brick:brick-A))	*a brick.*

This example demonstrates several points about our conventions. Notice first that, within the situation facts-at-hand, the instances of person and of tell are declared with the colon notation. These instance names are recognized throughout facts-at-hand as well as in the subordinate situation s1. Notice also that the name s1 is used twice as the name of two different instances, one an instance of a situation, and the other an instance of a selling event. As the name of a situation, s1 is limited in scope to the situation facts-at-hand, in which it is declared. The top-level situation of the system does not recognize this name directly, so it must refer to situation s1 by the name facts-at-hand/s1. Then, within the situation s1, the name s1 is redeclared as the name of a selling event. It is recognized as such throughout this subordinate situation.

Chapter 5 Representations of Legal Doctrine

Using the modeling language presented in Chapter 4, we will now construct machine representations for legal doctrine. We will build these models at two levels of generality. First we will consider more general statements of doctrine, like those found in secondary authority and in statutes. Then we will consider the more particular doctrine found primarily in the facts and holdings of individual cases. The area of legal doctrine that we will explore comprises the torts of battery and assualt. Before building our models, it will be helpful to discuss briefly the contextual setting of these torts.

5.1 The Torts of Battery and Assualt

An action in tort is a legal action in which a private individual (or individuals) called the plantiff complains to the court of a wrong committed by another, called the defendant, which wrong has caused the plaintiff to suffer financial loss, physical injury, or some other legally recognized form of harm. The plaintiff asks the court for a remedy, usually in the form of financial compensation from the defendant.

An action in tort is a civil action. This differs from a criminal proceeding, in which a state or federal government accuses a defendant of engaging in proscribed behavior and asks the court to punish the defendant with confinement in prison, or a fine paid to the government, or both.

An action in tort is one of several kinds of civil action. It differs, for example, from a civil action in contract, because the plaintiff's claim is based on legally recognized rights and duties among individuals, independant of any contractual promises.

We will consider the torts called battery and assault. These torts belong to a category sometimes called intentional interference with the person. This category is distinct from intentional interference with property (e.g. tresspass to land), and from unintentional torts, which include torts involving negligence. The terms "assault" and "battery" are also used to describe certain criminal behavior, but we will consider only the tort doctrines.

The contextual hierarchy of battery and assault is illustrated in figure 5-1. Notice that this breakdown corresponds to the portion of the kind hierarchy listed on page 88 under the category legal-action and to the portion listed on page 83 under the category legal-doctrine.

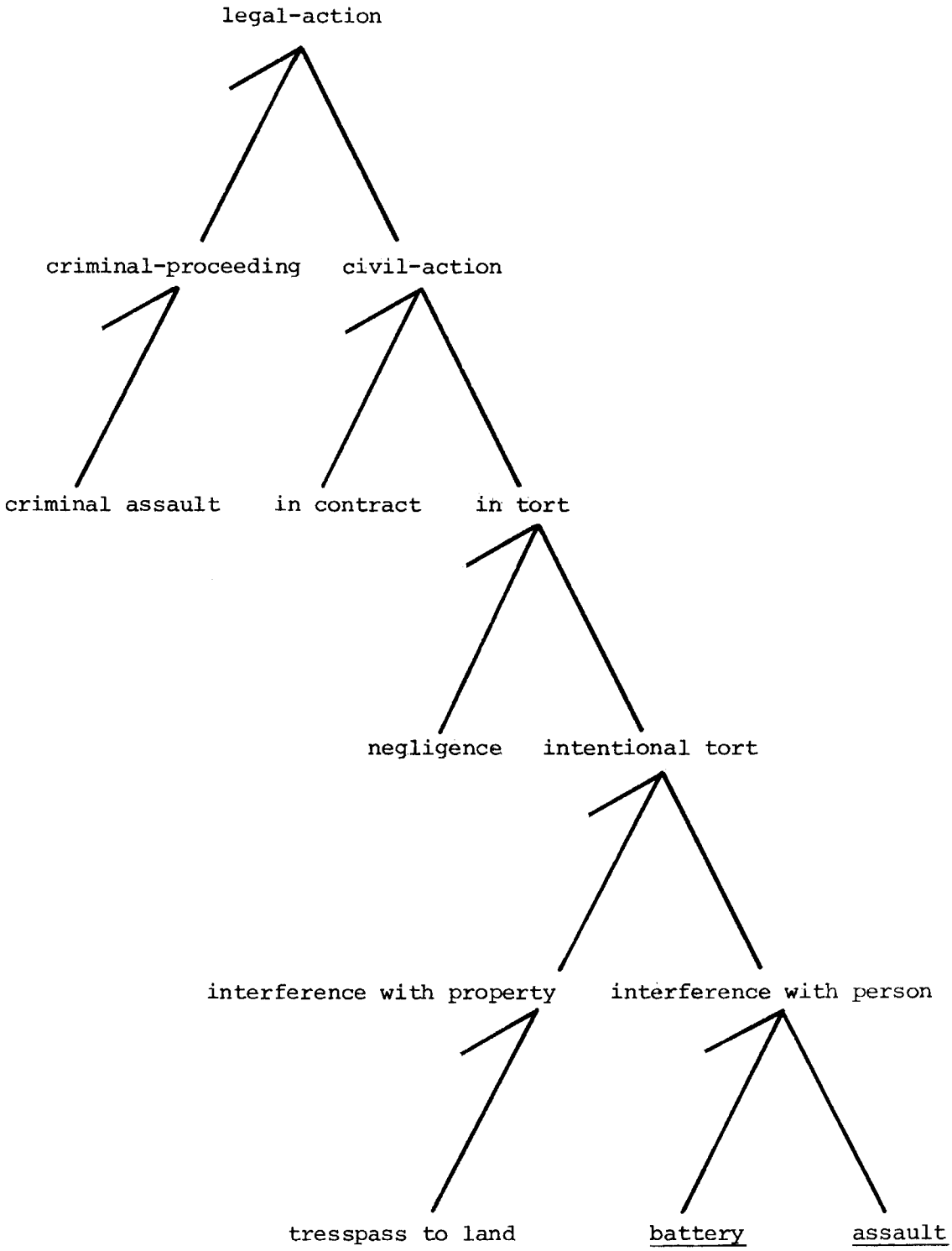

legal-action
criminal-proceeding
civil-action
criminal assault
in contract
in tort
negligence
intentional tort
interference with property
interference with person
tresspass to land
battery
assault

Figure 5-1.

5.2 Generalized Representations of Battery and Assault

We will base our generalized representations of battery and assault on the following summary statements fround in the well known treatise on torts by William L. Prosser:

BATTERY

> One is liable to another for unpermitted, unprivileged contacts with his person, caused by acts intended to result in such contacts, or the apprehension of them, directed at the other or a third person [33].

ASSAULT

> The defendant is liable for the apprehension of immediate harmful or offensive contact with the plaintiff's person, caused by acts intended to result in such contacts, or the apprehension of them, directed at the plaintiff or a thrid person [34].

We should observe, as Prosser points out, that "the difference between assault and battery is that between physical contact and the mere apprehension of it. One may exist without the other [35]."

We will attribute our generalized representations to a fictitious legal encyclopedia, Corpus Juris Mechanicum, which, like Posser's treatise, is understood to be a source of secondary legal authority.

5.21 Battery

Our representation of battery begins with the following statement:

(assertion	*It is an assertion*
	of
Corpus-Juris-Mechanicum	*Corpus Juris Mechanicum*
	that
(legal-consequence	*the legal consequence*
	of
situation:cjm-battery	*a situation called cjm-battery*
	is
(liability	*a liability*
cjm-battery/p	*to p (the plaintiff)*
cjm-battery/d)))	*from d (the defendant).*

In other words, according to Corpus Juris Mechanicum, d becomes liable to p as a consequence of a factual situation, which will be called cjm-battery, and within which d and p are intentified (declared). Notice that this framework is compatible with Layman Allen's system of propositional logic, wherein propositions of legal consequences are implied by propositions of conditions (discussed on page 43). It is also compatible with Wesley Hohfeld's system of rights, duties, liabilities, and immunities (discussed on page 40).

Next, we represent the main components of the situation cjm-battery:

(element cjm-battery	*Cjm-battery comprises*
(instance person p),	*a person, p (the plaintiff),*
	and
(instance person d),	*a person, d (the defendant),*
	and
situation:contact,	*a situation called contact,*
	and
situation:intent)	*a situation called intent.*

I.e., this situation, in which p (the plaintiff) and d (the defendant) are declared, comprises two subordinate situations called contact and intent.

Next, we state:

(counter-element cjm-battery	*Cjm-battery is avoided*
situation:consent,	*by a situation called consent,*
	or
situation:privilege)	*by a situation called privilege.*

Here we are using a new relation counter-element to indicate that the subordinate situations called consent and priviledge must be absent from cjm-battery. Alternatively, we could express this fact by using the Boolean relation not:

(element cjm-battery	*Cjm-battery further comprises*
(not situation:consent),	*the lack of consent,*
	and
(not situation:privilege)	*the lack of privilege.*

Our next step is to represent the subordinate situations contact, intent, consent, and priviledge. Contact is represented like this:

(element cjm-battery/contact	*Cjm-battery/contact comprises*
(s-object contact-event:c	*contact to the*
p):f,	*plaintiff*
(cause c event:e),	*as a result of an act*
(agent e d))	*of the defendant.*

This represents a situation in which the plaintiff is the semantic object of a contact-event (e.g., hit) caused by an act of the plaintiff. Recall (page 67) that s-object, cause, and agent are kinds of semantic relations. We have given a name to the fact f so that we can refer to it in other parts of our representation. This situation is sufficiently simple to be illustrated, in figure 5-2, with the graphical method used in Chapter 3. (For purposes of completeness, the illustration contains a few facts that are not elements of cjm-battery/contact.)

Next, we need a representation for intent. Because the issue of intent is involved in both battery and assault, we will construct a model that can be used in the representations of both of these torts. Indeed, intent is the characteristic component in all intentional torts. Prosser, for example, states the doctrine of intent in a section of his treatise that discusses intentional torts in general. He summarizes:

The situation called cjm-battery/contact comprises a contact to the plaintiff as a result of an act of the defendant.

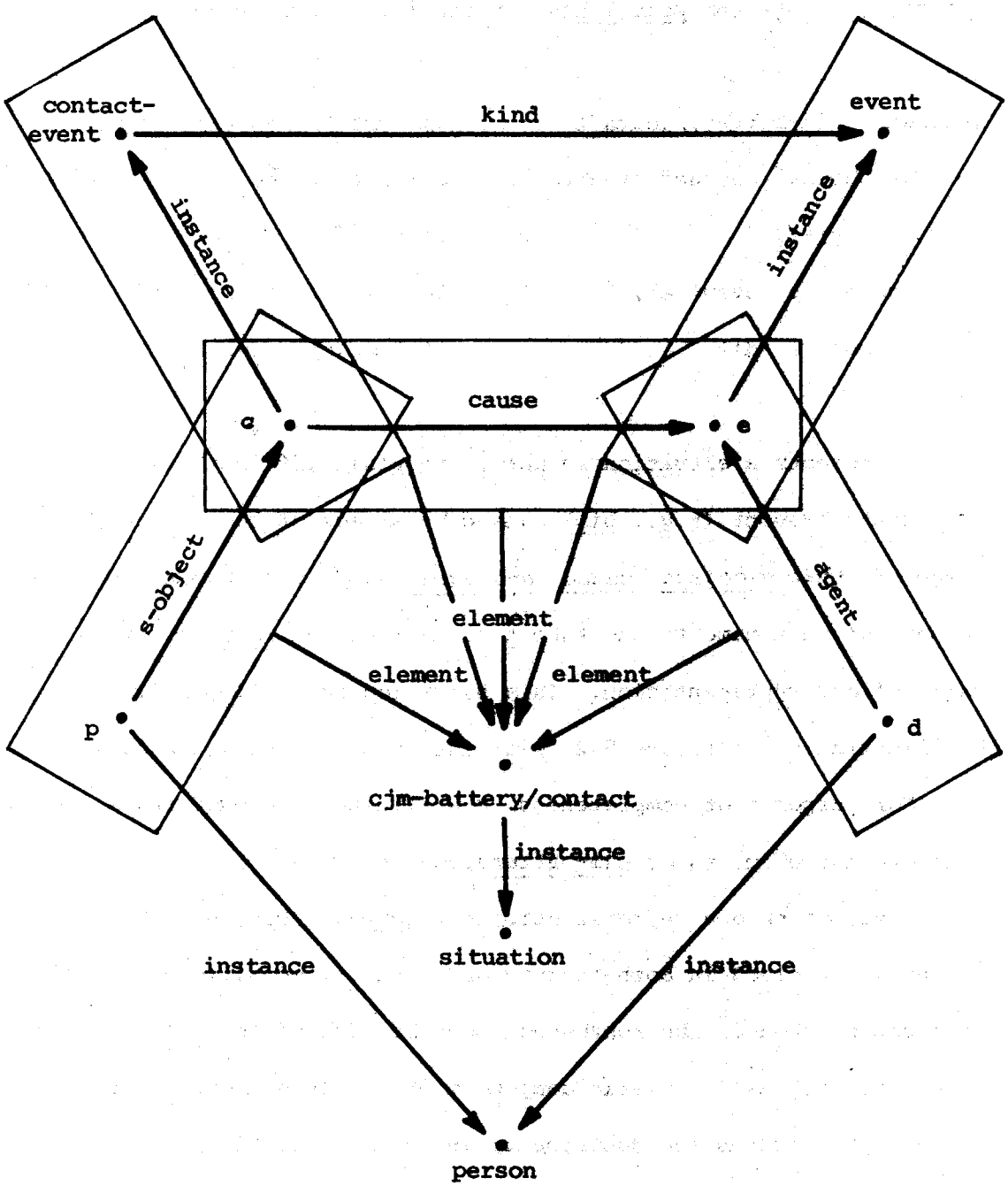


Figure 5-2.

MEANING OF INTENT

> 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 [36].

We will base our representation on this statement (although we will ignore the issue of substantial certainty). We represent the general doctrine of intent as follows:

(element cjm-intent	*Cjm-intent comprises*
(agent event:e person:p)	*a person's acting*
(or (purpose e	*either* *with the purpose*
situation:result	*of* *achieving a certain result,*
(belief p	*or* *with the belief by that person*
(cause result e))))	*that* *the act will cause that result.*

I.e., cjm-intent is a situation in which a person acts either for the purpose of causing a certain result, or which the belief that his action will cause that result. (The semantic relation purpose relates an action to the purpose of the agent of that action.) Notice that this representation makes use of the Boolean relation or.

Now we want to invoke this general doctrine of intent within the doctrine of battery. The person p mentioned in cjm-intent must be identified with the defendant d in cjm-battery. The situation result mentioned in cjm-intent corresponds to what Prosser describes in his statement of battery as "such contacts, or the apprehension of them, directed

at the other or a third person." We represent all of this as follows:

(element cjm-battery/intent	*Cjm-battery/intent comprises*
cjm-intent,	*cjm-intent*
	where
(is cjm-intent/p d),	*the person who intends is the defendant, and*
(is cjm-intent/result	*the intended result is*
	either
(or (s-object	
(contact-event:c	*a contact*
	to
person:x):fl	*someone,*
	or
(apprehension	*the apprehension*
x fl))),	*by someone of such a contact,*
	where
(not (is d x)))	*that someone is somebody other than the defendant.*

Here we use a new relation <u>is</u> to represent the necessary identifications between the instances in <u>cjm-intent</u> and those in <u>cjm-battery/intent</u>. We use this relation also in the final fact, (not (is d x)), to indicate that person <u>x</u> is someone other than the defendant. Thus, <u>x</u> represents the plaintiff "or a third person."

Next we must represent the situation <u>consent</u>. Again we turn to further discussion by Prosser:

> Consent to an act is simply willingness that it should occur. Actual consent to the defendant's conduct . . . will prevent liability. But a manifestation of consent . . . will be equally effective. . . . [In addition,] the defendant is sometimes at liberty to infer consent as a matter of usage or custom [37].

This discussion demonstrates a point that will become important for our modeling technique. Sometimes it is useful to represent a situation not in terms of component elements, but in terms of alternative types. Here for example, we see that there are three basic types of consent: actual, manifested, and inferred. Accordingly, we will represent these three types separately, rather than attempting a representation of consent as a whole in terms of component elements.

For this purpose, we introduce the relation type. We will allow situations (and sometimes facts) to be represented as types of other situations (and sometimes of other facts). Notice that the relation type is similar to the more primitive relation kind, just as the relation element is similar to the more primitive relation part. Type and kind behave disjunctively, like the Boolean relation or. Element and part behave conjunctively, like the Boolean relation and.

Using the type relation, we can represent the situation consent as follows:

(type cjm-battery/consent	*Cjm-battery/consent can be*
situation:actual,	*actual consent,*
	or
situation:manifested,	*manifested consent,*
	or
situation:inferred)	*inferred consent.*

We can represent actual and manifested consent in terms of component elements:

(element	
cjm-battery/consent/actual	*Actual consent comprises*
(willingness p	*the willingness of the plaintiff*
cjm-battery/contact/f))	*to being contacted.*
(element	
cjm-battery/consent/manifested	*Manifested consent comprises*
(agent manifest:m p),	*the plaintiff's manifesting*
(co-agent m d),	*to the defendant*
(s-object m actual))	*the willingness of the plaintiff to being contacted.*

I.e., in the situation representing manifested consent, the plaintiff manifests to the defendant the situation called actual which we use to represent actual consent. Recall from page 87, that manifest is a kind of communication-event. We will not construct a more detailed representation of inferred consent here. In section 5.3, we will see how this type of consent can be represented by an example.

Finally, we need a representation for privilege. Like consent, this situation is best represented in terms of alternative types, of which there are many, such as: self-defense, authority of a police officer, disciplinary action of a parent toward a child, public necessity, et cetera:

```
(type  cjm-battery/privilege
            situation:self-defense,
            situation:authority-of-law,
            situation:parental-discipline,
            situation:public-necessity,
            .
            .
            . ).
```

Rather than explore this large area of tort law, we will omit privilege as a counter-element in our representation of battery.

5.22 The Use of Co-Descriptive Models

In the previous section, we represented some situations in terms of their component elements, and we represented some in terms of their alternative types. The analysis procedures to be described in Chapter 6 rely on these representations in their attempts to fit particular facts into situational categories. It is not always possible to predict whether the method of elements or the method of types will be most helpful for this task. It is therefore important to realize that these two methods for representing situations are not mutally exclusive. It often will be very useful to include both kinds of representation for a particular situation. We will call such representations <u>co-descriptive</u>.

Let us consider again the representation we presented for the situation cjm-battery/contact:

(element cjm-battery/contact	*Cjm-battery/contact comprises*
(s-object contact-event:c	*contact to the*
p):f,	*plaintiff*
(cause c event:e),	*as a result of an act*
(agent e d))	*of the defendant.*

We can supplement this representation with others based on some different types of contact that are easily brought to mind.

The simplest type of contact between a defendant and a plaintiff occurs when the defendant contacts the plaintiff directly with a part of his or her own body, such as a hand or foot. In this situation, the defendant is the semantic agent of the same contact event of which the plaintiff is the semantic object. We can represent this as a type of contact as follows:

(type cjm-battery/contact	*Cjm-battery/contact can be*
situation:direct-contact)	*direct contact.*
(element	
cjm-battery/contact/direct-contact	*Direct contact comprises*
(s-object contact-event:c	*contact to the*
p),	*plaintiff*
(agent c d))	*by the defendant.*

A similar type of contact takes place when the defendant directly <u>moves</u> the plaintiff (e.g., pushes or pulls), although it happens that our kind hierarchy does not include these events under the category <u>contact-event</u>. We can remedy this omission as follows:

(type cjm-battery/contact	*Cjm-battery/contact can be*
situation:movement)	*movement.*
(element	
cjm-battery/contact/movement	*Movement comprises*
(s-object move-object:m	*the moving of the*
p),	*plaintiff*
(agent m d))	*by the defendant.*

Another type of contact occurs when the defendant throws an object at the plaintiff:

(type cjm-battery/contact	*Cjm-battery/contact can be*
situation:projectile)	*with a projectile.*
(element	
cjm-battery/contact/projectile	*Contact with a projectile comprises*
(s-object contact-event:c	*contact to the*
p),	*plaintiff*
(agent c movable-object:o),	*by a moveable object*
(s-object throw:t o)	*that is thrown*
(agent t d))	*by the defendant.*

There are other types of contact that could be added to this list, but these will suffice for our current representation. We will include these types of contact, as well as the previously expressed elements of contact, within our representation of battery.

In the previous section, we represented cjm-battery/consent solely in terms of three alternative types, actual, manifested, and inferred. Here, too, it will be more helpful to the analysis procedures if this situation were co-descriptively represented, in terms of elements as well as types. Accordingly, we will add the following fact to our representation of cjm-battery/consent:

(element cjm-battery/consent	*Cjm-battery/consent comprises*
(consent p	*consent of the plaintiff to*
cjm-battery/consent/f))	*being contacted.*

The relation consent is found in the kind hierarchy as a kind of mental attitude. This co-description does not provide a breakdown of consent into smaller pieces, but it does allow the user to include in facts-at-hand facts relating to consent as a whole, as well as facts relating to the three types of consent.

5.23 Assault

We now turn our attention from battery to assault. In the interest of simplification, we will remove some of the elements included in Prosser's restatement quoted on page 107. Ignoring the issues of immediacy, harmfulness, and offensiveness, we have:

> One is liable for the apprehension of contact with the plaintiff's person, caused by acts intended to result in such contacts or the apprehension of them, directed at the plaintiff or a third person.

This leaves us with two major components: apprehension and intent. We begin our representation in the same manner used for battery:

(assertion	*It is the assertion*
	of
Corpus-Juris-Mechanicum	*Corpus Juris Mechanicum*
	that
(legal-consequence	*the legal consequence*
	of
situation:cjm-assault	*a situation called cjm-assault*
	is
(liability	*a liability*
cjm-assault/p	*to p (the plaintiff)*
cjm-assault/d)))	*from d (the defendant).*
(element cjm-assault	*Cjm-assault comprises*
(instance person p),	*a person, p (the plaintiff)*
	and
(instance person d),	*a person, d (the defendant),*
	and
situation:apprehension,	*a situation called apprehension,*
	and
situation:intent)	*a situation called intent.*

Notice that the instance names p, d, and intent are declared separately in cjm-battery and in cjm-assault.

We represent the situation apprehension as follows:

(element	
cjm-assault/apprehension	*Cjm-assault/apprehension comprises*
(apprehension p	*apprehension by the plaintiff*
(s-object	*of*
contact-event:c	*contact to the*
p)):f,	*plaintiff,*
(cause f event:e),	*which apprehension is the result of an act*
(agent e d))	*of the defendant.*

I.e., an act of the defendant causes the plaintiff to be apprehensive that he or she will be the (semantic) object of a contact event.

We can include co-descriptive representations for the situation cjm-assault/apprehension, just as we did for the situation cjm-battery/contact. One typical form of this situation occurs when the defendant visibly moves his or her fist close to the plaintiff:

(type cjm-assault/apprehension	*Cjm-assault/apprehension can be*
situation:threaten-with-fist)	*a threat made with a fist.*

(element cjm-assault/apprehension/ threaten-with-fist	*A threat made with a fist comprises*
situation:threat,	*a situation called threat, and*
(perception p threat))	*the perception by the plaintiff of the threat,*
(element cjm-assault/apprehension/ threaten-with-fist/threat	*where the threat comprises*
(agent move-object:m d),	*the defendant's moving*
(s-object m fist:f),	*a fist*
(part d f),	*of the defendant*
(destination m place:pl),	*to a place*
(near p pl))	*near the plaintiff.*

This situation may seem more complicated than <u>apprehension</u> itself, but it is made up of much smaller, less generalized concepts. Notice, for example, that it incorporates--and therefore avoids--the issue of causation, which is an explicit element of <u>apprehension</u>. (All of the types of <u>contact</u> presented in the previous section also avoid the issue of causation.)

Another frequently occurring type of apprehension involves a threat with a firearm:

(type cjm-assault/apprehension	*Cjm-assault/apprehension can be*
situation:threaten-with-gun)	*a threat made with a gun.*

(element cjm-assault/apprehension/ threaten-with-gun	*A threat made with a gun comprises*
situation:threat,	*a situation called threat, and*
(perception p threat))	*the perception by the plaintiff of the threat,*
(element cjm-assault/apprehension/ threaten-with-gun/threat	*where the threat comprises*
(agent point:pt d),	*the defendant's pointing*
(s-object pt firearm:f),	*a firearm*
(destination pt p))	*at the plaintiff.*

We terminate our representation of apprehension here, realizing of course, that there are many other types that could be included.

Our representation of cjm-assault/intent parallels that used for cjm-battery-intent. Once again, we invoke the more general doctrine of cjm-intent, and then we specialize it in accordance with the rest of the representation of cjm-assault:

(element cjm-assault/intent	*Cjm-assault/intent comprises*
cjm-intent	*cjm-intent, where*
(is cjm-intent/p d),	*the person who intends is the defendant, and*
(is cjm-intent/result	*the intended result is either*
(or (s-object	
contact-event:c	*a contact to*
person:x):fl	*someone, or*
(apprehension	*the apprehension*
x fl))),	*by someone of such a contact,*

(not (is d x)))	*where that someone is somebody other than the defendant.*

Finally, we can represent the situations cjm-battery and cjm-assault as types of the situation cjm-intentional-tort (just as battery and assault are kinds of intentional-tort in the kind hierarchy):

(type situation:cjm-intentional-tort	*Cjm-intentional-tort can be*
situation:cjm-battery,	*cjm-battery, or*
situation:cjm-assault)	*cjm-assault.*

5.24 Summary of Generalized Representations

The following is a summary of the representations presented in sections 5.21 through 5.23:

(type situation:cjm-intentional-tort	*Cjm-intentional-tort*
	can be
situation:cjm-battery,	*cjm-battery,*
	or
situation:cjm-assault)	*cjm-assault.*
(element cjm-intent	*Cjm-intent comprises*
(agent event:e person:p),	*a person's acting*
	either
(or (purpose e	*with the purpose*
	of
situation:result	*achieving a certain result,*
	or
(belief p	*with the belief by that person*
	that
(cause result e))))	*the act will cause that result.*
(assertion	*It is an assertion*
	of
Corpus-Juris-Mechanicum	*Corpus Juris Mechanicum*
	that
(legal-consequence	*the legal consequence*
	of
situation:cjm-battery	*a situation called cjm-battery*
	is
(liability	*a liability*
cjm-battery/p	*to p (the plaintiff)*
cjm-battery/d)))	*from d (the defendant).*

(element cjm-battery	*Cjm-battery comprises*
(instance person p),	*a person, p (the plaintiff), and*
(instance person d),	*a person, d (the defedant), and*
situation:contact,	*a situation called contact, and*
situation:intent)	*a situation called intent.*
(counter-element cjm-battery	*Cjm-battery is avoided by*
situation:consent)	*a situation called consent.*

(element cjm-battery/contact	*Cjm-battery/contact comprises*
(s-object contact-event:c	*contact to the*
p):f,	*plaintiff*
(cause c event:e),	*as a result of an act*
(agent e d))	*of the defendant.*

(type cjm-battery/contact	*Cjm-battery/contact can be*
situation:direct-contact,	*direct contact, or*
situation:movement,	*movement, or*
situation:projectile)	*with a projectile.*
(element	
cjm-battery/contact/direct-contact	*Direct contact comprises*
(s-object contact-event:c	*contact to the*
p),	*plaintiff*
(agent c d))	*by the defendant.*

(element	
cjm-battery/contact/movement	*Movement comprises*
(s-object move-object:m	*the moving of the*
p),	*plaintiff*
(agent m d))	*by the defendant.*
(element	
cjm-battery/contact/projectile	*Contact with a projectile comprises*
(s-object contact-event:c	*contact to the*
p),	*plaintiff*
(agent c movable-object:o),	*by a moveable object*
(s-object throw:t o),	*that is thrown*
(agent t d))	*by the defendant.*
(element cjm-battery/intent	*Cjm-battery/intent comprises*
cjm-intent,	*cjm-intent where*
(is cjm-intent/p d),	*the person who intends is the defendant, and*
(is cjm-intent/result	*the intended result is either*
(or (s-object	
(contact-event:c	*a contact to*
person:x):fl	*someone, or*
(apprehension	*the apprehension*
x fl))),	*by someone of such a contact, where*
(not (is d x)))	*that someone is somebody other than the defendant.*

(element cjm-battery/consent	*Cjm-battery/consent*
	comprises
(consent p	*consent of the plaintiff*
	to
cjm-battery/consent/f))	*being contacted.*
(type cjm-battery/consent	*Cjm-battery/consent can be*
situation:actual,	*actual consent,*
	or
situation:manifested,	*manifested consent,*
	or
situation:inferred)	*inferred consent.*
(element	
cjm-battery/consent/actual	*Actual consent comprises*
(willingness p	*the willingness of the plaintiff*
cjm-battery/contact/f))	*to being contacted.*
(element	
cjm-battery/consent/manifested	*Manifested consent comprises*
(agent manifest:m p),	*the plaintiff's manifesting*
(co-agent m d),	*to the defendant*
(s-object m actual))	*the willingness of the plaintiff to being contacted.*

(assertion	*It is the assertion*
	of
Corpus-Juris-Mechanicum	*Corpus Juris Mechanicum*
	that
(legal-consequence	*the legal consequence*
	of
situation:cjm-assault	*a situation called cjm-assault*
	is
(liability	*a liability*
cjm-assault/p	*to p (the plaintiff)*
cjm-assault/d)))	*from d (the defendant).*
(element cjm-assault	*Cjm-assault comprises*
(instance person p),	*a person, p (the plaintiff)*
	and
(instance person d),	*a person, d (the defendant),*
	and
situation:apprehension,	*a situation called apprehension,*
	and
situation:intent)	*a situation called intent.*

(element	
cjm-assault/apprehension	*Cjm-assault/apprehension comprises*
(apprehension p	*apprehension by the plaintiff*
(s-object	*of*
contact-event:c	*contact to the*
p)):f,	*plaintiff,*
(cause f event:e),	*which apprehension is the result*
	of an act
(agent e d))	*of the defendant.*

(type cjm-assault/apprehension	*Cjm-assault/apprehension can be*
situation:threaten-with-fist,	*a treat made with a fist, or*
situation:threaten-with-gun)	*a threat made with a gun.*
(element cjm-assault/apprehension/ threaten-with-fist	*A threat made with a fist comprises*
situation:threat,	*a situation called threat, and*
(perception p threat))	*the perception by the plaintiff of the threat,*
(element cjm-assault/apprehension/ threaten-with-fist/threat	*where the threat comprises*
(agent move-object:m d),	*the defendant's moving*
(s-object m fist:f),	*a fist*
(part d f),	*of the defendant*
(destination m place:pl),	*to a place*
(near p pl))	*near the plaintiff.*
(element cjm-assault/apprehension/ threaten-with-gun	*A threat made with a gun comprises*
situation:threat,	*a situation called threat, and*
(perception p threat))	*the perception by the plaintiff of the threat,*
(element cjm-assault/apprehension/ threaten-with-gun/threat	*where the threat comprises*
(agent point:pt d),	*the defendant's pointing*
(s-object pt firearm:f),	*a firearm*
(destination pt p))	*at the plaintiff.*

(element cjm-assault/intent	*Cjm-assault/intent comprises*
cjm-intent	*cjm-intent,*
	where
(is cjm-intent/p d),	*the person who intends is the defendant, and*
(is cjm-intent/result	*the intended result is either*
(or (s-object	
contact-event:c	*a contact*
	to
person:x):fl	*someone,*
	or
(apprehension	*the apprehension*
x fl))),	*by someone of such a contact*
	where
(not (is d x)))	*that someone is somebody other than the defendant.*

5.3 Representations of Specific Cases

In this section, we will describe representations for the more specific doctrine expressed by the facts and holdings of individual case decisions. We have already seen how alternative types can be used in the representation of general doctrine. Each alternative type of a situation provides an example that is more specific than any representation for that situation with which the type might be co-descriptive. The examples provided by individual cases behave in a similar manner, except that they are often more specific, and they are attributed to separate sources of legal authority--the decisions themselves.

The cases that we will represent in the prototype system are fictitious simplifications. While they are based on the actual case law of battery and assault, they involve only small sets of facts.

5.31 Specific Facts and Categorized Holdings

In Chapter 2, we discussed the difference between the specific facts in a case and the more general categories in terms of which the holding of a case often is written. We looked at one holding, for example, in which an instance of a particular appendectomy was represented by the category "internal operation" (page 22). We will want to include representations for both the specific facts and the categories used in the holdings of cases. Our analysis procedures will use the categorized representations when attempting to fit facts being analyzed into, or near

to, the holding of a case. When this succeeds, the system will also provide the user with the particular facts in that case.

We will describe our method for representing facts and holdings with and example. Consider a case in which the facts are simply: Joe Moe punches Fred Foe in the nose. The holding in such a case might be: "When one person strikes the anatomy of another, there is contact as required for a battery." We represent the basic framework for this holding in a manner similar to that used for the doctrines contained in Corpus Jurus Mechanicum:

(holding	*It is the holding*
	in
judicial-decision:foe-v-moe	*Foe v. Moe*
	that
(legal-consequence	*the legal consequence*
	of
situation:s-foe-v-moe	*a situation called s-foe-v-moe is*
cjm-battery/contact))	*contact as required for a battery.*

On other words, the case Foe v. Moe holds that the situation called s-foe-v-moe establishes the contact component of battery. Notice that s-foe-v-moe is, in effect, a type of cjm-battery/contact. When a type of situation is part of a case holding, we will call it an example of that situation. We will use the following representation as an equivalent to the one immediately preceding:

(example cjm-battery/contact situation:s-foe-v-moe)	*An example of cjm-battery/contact is the situation s-foe-v-moe.*

The situation <u>s-foe-v-moe</u> can be represented as follows:

(element s-foe-v-moe	*In s-foe-v-moe,*
(agent strike:s person:p1),	*one person strikes*
(s-object s person:p2),	*another person*
(destination s anatomical-object:a),	*in the anatomy*
(part p2 a),	*of the other person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and*
(is cjm-battery/d p1))	*the first person is the defendant, in an action in battery.*

We represent the specific facts in this case by using a new relation called <u>facts</u>:

(facts s-foe-v-moe	*Specifically, in <u>Foe</u> v. <u>Moe</u>,*
(is p1 person:Joe-Moe),	*the first person is Joe Moe, and*
(is p2 person:Fred-Foe),	*the second person is Fred Foe, and*
(is a nose:n),	*the anatomical object is a nose, and*
(is s punch:p))	*the strike is a punch.*

In this way, the <u>facts</u> of <u>s-foe-v-moe</u> behave as supplementary elements of that situation.

Notice that the representation for the holding of this case makes reference to cjm-battery, a piece of doctrine contained in Corpus Juris Mechanicum. It is unusual for a case deision to incorporate doctrine from a secondary authority. It is necessary that we do so in our prototype, however, because Corpus Juris Mechanicum is its only embodiment of generalized doctrine. This impropriety can be lessened by including two cases that hold (by weight of primary authority) that the doctrines of cjm-battery and cjm-assault are indeed the law of the land:

(holding	*It is the holding*
judicial-decision:smith-v-jones	*in* *Smith v. Jones*
(legal-consequence	*that* *the legal consequence of the*
cjm-battery	*situation cjm-battery, defined in Corpus Juris Mechanicum,*
(liability cjm-battery/p	*is* *a liability to the plaintiff*
cjm-battery/d)))	*from the defendant.*

(holding	*It is the holding*
judicial-decision:jones-v-smith	*in* *Jones v. Smith*
(legal-consequence	*that* *the legal consequence of the*
cjm-assault	*situation cjm-assault, defined in Corpus Juris Mechanicum,*
(liability cjm-assault/p	*is* *a liability to the plaintiff*
cjm-assault/d)))	*from the defendant.*

These cases can now be used as more authoritative embodiments of our general doctrines of battery and assault.

Let us consider a second specific case. Here are the facts: John Doe knocks off the hat that Richard Roe is wearing. Contacts to things closely attached to a person are generally held to be contacts to the person for purposes of establishing battery. Therefore, we might state a holding for this case: "When a person strikes an article of clothing on the person of another, there is contact as required for battery." Our representation is:

(example cjm-battery/contact	*An example of cjm-battery/contact*
situation:s-roe-v-doe)	*is a situation*
(element s-roe-v-doe	*in which*
(agent strike:s person:p1),	*one person strikes*
(s-object s	
article-of-clothing:a),	*an article of clothing*
(on person:p2 a),	*on another person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and the first person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*
(facts s-roe-v-doe	*Specifically, in Roe v. Doe,*
(is p1 person:John-Doe),	*the first person is John Doe, and*
(is p2 person:Richard-Roe),	*the second person is Richard Roe, and*
(is a hat:h),	*the article of clothing is a hat, and*
(is s knocks-off))	*the strike is a knocking off.*

Sometimes the facts in a particular case provide a counter-example, rather than an example, of a piece of more general doctrine. Let us consider a case similar to the previous case, but where the doctrine of contact is not established. The facts: Bill Boe slaps a hat belonging to Carl Coe when the hat is lying on a table. The holding: "If a person contacts an article of clothing when it is not on another's person, there is no contact as required for battery." The representation:

(counter-example	*A counter-example*
cjm-battery/contact	*of cjm-battery/contact*
situation:s-coe-v-boe)	*is a situation*
(element s-coe-v-boe	*in which*
(agent contact-event:c	
person:p1),	*one person contacts*
(s-object c	
article-of-clothing:a),	*an article of clothing, and*
(not (on person:p2 a)),	*the article of clothing is not on a second person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and the first person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*

(facts s-coe-v-boe	*Specifically, in Coe v. Boe,*
(is p1 person:Bill-Boe),	*the first person is Bill Boe,*
	and
(is p2 person:Carl-Coe),	*the second person is Carl Coe,*
	and
(is c slap:s),	*the contact is a slap,*
	and
(is a hat:h),	*the article of clothing is a hat,*
	and
(owner j Carl-Coe),	*the owner of the hat is Carl Coe,*
	and
(on table:t h))	*the hat is on the table.*

Notice that we are using the counter-example relationship as an equivalent for:

(holding	*It is the holding*
	in
judicial-decision:coe-v-boe	*Coe v. Boe*
	that
(legal-consequence	*the legal consequence*
	of
situation:s-coe-v-boe	*a situation called s-coe-v-boe*
	is
(not cjm-battery/contact)))	*not contact as required for*
	a battery.

Next, let us examine a case that provides an example of inferred consent. These are the facts: Perry Poe and Quentin Quoe are fighting in a boxing match. Perry Poe punches Quentin Quoe in the jaw. This is the holding: "Where two persons participate in an athletic competition, consent as will avoid a battery may be inferred for any physical contact that is part of the competition." The representation looks like this:

(example	*An example of*
cjm-battery/consent/inferred	*cjm-battery/consent/inferred*
situation:s-quoe-v-poe)	*is a situation*
(element s-quoe-v-poe	*in which*
(agent	
athletic-competition:a	*an athletic competition is*
person:p1,person:p2),	*played by two persons, and*
(s-object	
contact-event:c p2),	*one of the persons is contacted, and*
(part a c),	*the contact is part of the competition, where*
(is cjm-battery/p p2),	*the contacted person is the plaintiff, and the other person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*
(facts s-quoe-v-poe	*Specifically, in* <u>*Quoe*</u> *v.* <u>*Poe*</u>*,*
(is p2 person:Quentin-Quoe),	*the person who is contacted is Quentin Quoe, and*
(is p1 person:Perry-Poe),	*the other person is Perry Poe, and*
(is a boxing-match:b),	*the competition is a boxing match, and*
(is c punch:p)	*the contact is a punch*
(destination p jaw:j)	*to the jaw*
(part Quentin-Quoe j),	*of Quentin Quoe*
(agent p Perry-Poe)	*by Perry Poe.*

Finally, we consider a case example within the doctrine of assault. The facts are these: Winnie Woe visibly points a rifle at Zeke Zoe. The rifle is not loaded, but Zeke does not know that. The holding: "Where a

person visibly points an unloaded firearm at another person, apprehension as is required for assault is established if the other person does not know that the firearm is unloaded."

This is an example of the type of apprehension that we called threaten-with-gun. The simplest way to represent the situation of this holding is to invoke the more general situation cjm-assault/apprehension/threaten-with-gun (see page 123), and then to specialize that situation with additional facts about the rifle's being unloaded:

(example	*An example of*
cjm-assault/apprehension/ threaten-with-gun	*cjm-assault/apprehension/ treaten-with-gun*
situation:s-zoe-v-woe)	*is a situation*
(element s-zoe-v-woe	*comprising*
cjm-assault/apprehension/ threaten-with-gun,	*a threat made with a gun,*
(loadedness f unloaded):fl,	*where the firearm is unloaded,*
(not (knowledge p fl)))	*and where the plaintiff does not know that the firearm is unloaded.*
(facts s-zoe-v-woe	*Specifically, in Zoe v. Woe,*
(is d person:Winnie-Woe),	*the defendant is Winnie Woe*
(is p person:Zeke-Zoe),	*and the plaintiff is Zeke Zoe,*
(is f rifle:r))	*and the firearm is a rifle.*

Notice that we are incorporating into this representation the instance names used in threaten-with-gun.

It should be clear that a large assortment of simple case examples like these can be represented in a similar manner. We will not do so here. The above cases, taken together with the more generalized doctrine described in section 5.2, are sufficient to illustrate our method of representation, and to support the examples of analysis that are described in Chapters 1 and 7.

5.32 Summary of Case Representations

The following is a summary of the representations presented in the previous section:

(holding	*It is the holding*
	in
judicial-decision:smith-v-jones	*Smith v. Jones*
	that
(legal-consequence	*the legal consequence of the*
	situation cjm-battery, defined
cjm-battery	*in Corpus Juris Mechanicum,*
	is
(liability cjm-battery/p	*a liability to the plaintiff*
cjm-battery/d))	*from the defendant.*

(holding	*It is the holding*
	in
judicial-decision:jones-v-smith	*Jones v. Smith*
	that
(legal consequence	*the legal consequence of the*
	situation cjm-assault, defined
cjm-assault	*in Corpus Juris Mechanicum,*
	is
(liability cjm-assault/p	*a liability to the plaintiff*
cjm-assault/d)))	*from the defendant.*

(example cjm-battery/contact	*An example of cjm-battery/contact*
situation:s-foe-v-moe)	*is a situation*
(element s-foe-v-moe	*in which*
(agent strike:s person:p1),	*one person strikes*
(s-object s person:p2),	*another person*

(destination s	*in the anatomy*
anatomical-object:a),	
(part p2 a),	*of the other person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and*
(is cjm-battery/d p1))	*the first person is the defendant, in an action in battery.*
(facts s-foe-v-moe	*Specifically, in Foe v. Moe,*
(is p1 person:Joe-Moe),	*the first person is Joe Moe, and*
(is p2 person:Fred-Foe),	*the second person is Fred Foe, and*
(is a nose:n),	*the anatomical object is a nose, and*
(is s punch:p))	*the strike is a punch.*

(example cjm-battery/contact	*An example of cjm-battery/contact*
situation:s-roe-v-doe)	*is a situation*
(element s-roe-v-doe	*in which*
(agent strike:s person:p1),	*one person strikes*
(s-object s	
article-of-clothing:a),	*an article of clothing*
(on person:p2 a),	*on another person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and the first person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*
(facts s-roe-v-doe	*Specifically, in Roe v. Doe*
(is p1 person:John-Doe),	*the first person is John Doe, and*
(is p2 person:Richard-Roe),	*the second person is Richard Roe, and*
(is a hat:h),	*the article of clothing is a hat, and*
(is s knocks-off))	*the strike is a knocking off.*

(counter-example	*A counter-example*
cjm-battery/contact	*of cjm-battery/contact*
situation:s-coe-v-boe)	*is a situation*
(element s-coe-v-boe	*in which*
(agent contact-event:c	
person:p1),	*one person contacts*
(s-object c	
article-of-clothing:a),	*an article of clothing, and*
(not (on person:p2 a)),	*the article of clothing is not on a second person,*
(is cjm-battery/p p2),	*where the second person is the plaintiff, and the first person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*
(facts s-coe-v-boe	*Specifically, in Coe v. Boe,*
(is p1 person:Bill-Boe),	*the first person is Bill Boe, and*
(is p2 person:Carl-Coe),	*the second person is Carl Coe, and*
(is c slap:s),	*the contact is a slap, and*
(is a hat:h),	*the article of clothing is a hat, and*
(owner h Carl-Coe),	*the owner of the hat is Carl Coe, and*
(on table:t h))	*the hat is on the table.*

(example	*An example of*
cjm-battery/consent/inferred	*cjm-battery/consent/inferred*
situation:s-quoe-v-poe)	*is a situation*

(element s-quoe-v-poe	*in which*
(agent	
athletic-competition:a	*an athletic competition is*
person:p1,person:p2),	*played by two persons, and*
(s-object	
contact-event:c p2),	*one of the persons is contacted, and*
(part a c),	*the contact is part of the competition, where*
(is cjm-battery/p p2),	*the contacted person is the plaintiff, and the other person*
(is cjm-battery/d p1))	*is the defendant, in an action in battery.*
(facts s-quoe-v-poe	
(is p2 person:Quentin-Quoe),	*Specifically, in Quoe v. Poe the person who is contacted is Quentin Quoe, and*
(is p1 person:Perry-Poe),	*the other person is Perry Poe, and*
(is a boxing-match:b),	*the competition is a boxing match, and*
(is c punch:p),	*the contact is a punch*
(destination p jaw:j),	*to the jaw*
(part Quentin-Quoe j),	*of Quentin Quoe*
(agent p Perry-Poe)	*by Perry Poe.*

(example	*An example of*
cjm-assault/apprehension/ threaten-with-gun	*cjm-assault/apprehension/ threaten-with-gun*
situation:s-zoe-v-woe)	*is a situation*
(element s-zoe-v-woe	*comprising*
cjm-assault/apprehension/ threaten-with-gun,	*a threat made with a gun,*

	where
(loadedness f unloaded):fl,	*the firearm is unloaded,*
	and where
(not (knowledge p fl)))	*the plaintiff does not know that the firearm is unloaded.*
(facts s-zoe-v-woe	
	Specifically, in <u>*Zoe*</u> *v.* <u>*Woe*</u>*,*
(is d person:Winnie-Woe),	*the defendant is Winnie Woe*
	and
(is p person:Zeke-Zoe),	*the plaintiff is Zeke Zoe,*
	and
(is f rifle:r))	*the firearm is a rifle.*

Chapter 6 Machine Procedures for Legal Analysis

Having examined the machine models used to represent factual situations and legal doctrine, we are ready to consider the machine procedures that are used for performing legal analysis. We will describe three sets of procedures. The loading procedures permit the user of the system to put representations of fact and doctrine into the computer's memory. The instantiation procedures guide the analysis per se. The system uses these in its attempts to fit a particular situation of fact into, or near to, the generalizations that are contained in legal doctrines. Finally, the discussion procedure is used by the system to describe to the user the results of an analysis.

6.1 Machine Procedures in PSL

When it is fully developed, the OWL system will be partially self-aware. It will "know" its own machine procedures in the same manner that it knows everything else about the world--in terms of the same OWL data structure. Its model of the world includes a special category of events for which the system itself is the semantic agent. We might call these events machine-events. The data upon which such a procedure operates, i.e., the argument of the procedure, is represented as the semantic object of that machine-event. Other semantic cases, such as precondition, method, result, are used to describe explicitly how the system is to

perform the operation. In addition to the procedures written in its own OWL language, the system contains a small set of primitive procedures written in LISP. These are used to perform simple operations on the data structure.

For the purposes of this study, we will not attempt to describe the analysis procedures by means of OWL structure or any other programming language. The simple procedures will be described in terms of their effects upon the machine representations. The more complicated procedures will be described in terms of their rules and algorithms.

We will, however, incorporate into PSL the basic concept that a machine procedure is a kind of event for which the system is the agent. When we wish the system to perform such an event, we will present it with a fact in which the relation is the word <u>execute</u>:

(execute machine-event-m th2).

When the PSL interpreter sees a command like this, it causes the system to perform (an instance of) the named procedure, taking the data structure named <u>th2</u> as its argument.

6.2 The Loading Procedures

Before our system can use the machine representations described in the previous chapters, these representations must be put into the computer's memory. In one part of the memory, called long-term memory, we will put representations of world knowledge (i.e., the kind hierarchy) and of legal doctrine. These representations are relatively permanent; they remain in the computer's memory from one analysis session to the next. In another part of the memory, called intermediate memory, the system stores the facts of a particular situation that is under analysis. The user of the system presents these facts at the beginning of each analysis sesssion, and they are deleted at the conclusion of the session. (A third part of memory, called short-term memory is used as work space by the system itself.)

6.21 Building Permanent Knowledge

A simple PSL procedure called learn sets up a fact in the system's long-term memory. Given the command:

(execute learn (rel th1 th2)),

the system establishes the appropriate memory cells, pointers, and back-pointers so that the fact:

(rel th1 th2)

becomes known permanently and directly by the system. Thus, the entire

kind hierarchy can be loaded into the system with a command like:

```
(execute  learn  (kind  something  thing,relation),
                 (kind  thing  object,event,value . . .),
                 .
                 .
                 .
                 (kind  furniture  chair,table,desk . . .),
                 .
                 .
                 . ).
```

Similarly, a piece of legal doctrine can be loaded with a command like:

```
(execute  learn  (assertion  Corpus-Juris-Mechanicum
                    (legal-consequence  situation:cjm-battery
                         (liability  cjm-battery/p
                                     cjm-battery/d))),
                 (element  cjm-battery  (instance  person  p),
                                        (instance  person  d),
                                        .
                                        .
                                        . ),
```

.

.

.).

6.22 Inserting a Factual Situation for Analysis

At the beginning of each analysis session, the user of the system must describe to the system the factual situation upon which legal analysis is to be performed. Recall that these facts are to be represented as elements of the situation called facts-at-hand. These facts are loaded into intermediate memory, instead of long-term memory, by use of the machine procedure insert, in place of the procedure learn. Thus, the user might type a command of the form:

(execute insert (element facts-at-hand fact-1,fact-2, . . .)).

During an analysis session, the system user is likely to be an attorney rather than a computer technician. We can make the insertion of the hypothetical situation less awkward by putting the system in a "state of mind" wherein it realizes that the facts presented are meant to be inserted as elements of facts-at-hand. (Recall also that in its ultimate embodiment, the system would receive these facts in English, not in PSL statements.) This could be accomplished by instructing the system to begin each analysis session with a procedure, called listen-to-facts,

that automatically inserts the facts typed by the user. The same procedure could screen the factual statements to detect certain kinds of errors. We will not explore errors here, except for one that is simple to detect and to correct. Whenever the user refers to a thing or a relation that is not known to the system (i.e., not in the kind hierarchy), the procedure listen-to-facts will ask the user to define it (as a kind of something that in the kind hierarchy.) Furthermore, whenever the user supplies a fact expressing a kind relation, the procedure listen-to-facts can invoke learn, rather than insert, on the assumption that such facts will be useful to keep in the system's permanent knowledge. The algorithm for listen-to-facts is shown in figure 6-1. Notice that the procedure assumes some primitive machine operations like display (which displays a message to the user) and receive (which reads a fact, called user-fact, typed by the user).

Algorithm for listen-to-facts

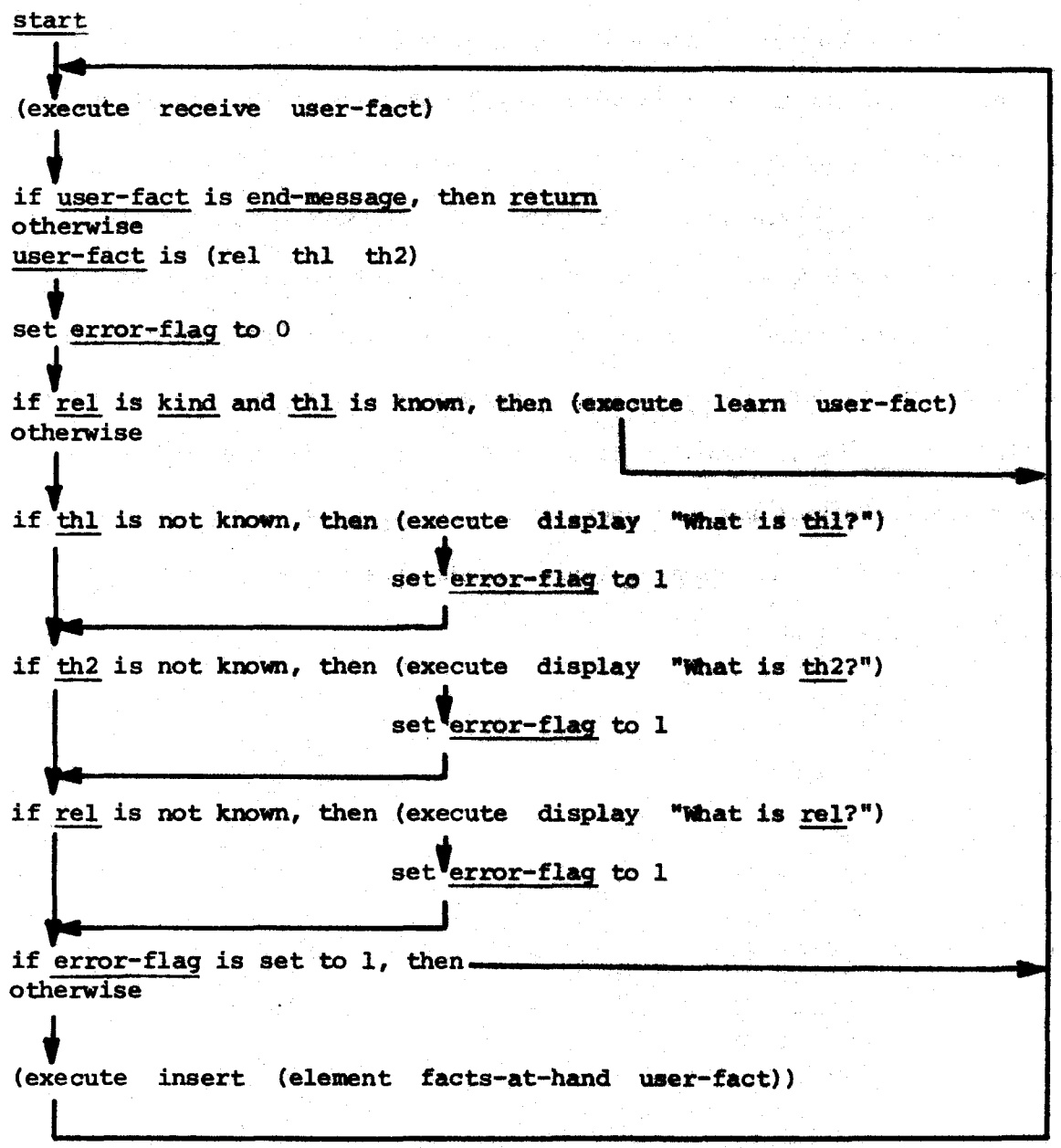


Figure 6-1.

6.3 The Instantiation Procedures

The instantiation procedures lie at the heart of our analysis system. We use the term instantiating to mean: finding, within the hypothetical situation facts-at-hand, specific facts that fit into, or near to, the generalized facts that are contained in the situational representation of a legal doctrine. This is the fundamental process on which our model of legal analysis is based.

Before exploring the different methods by which instantiation can be achieved, let us look at a simple example to illustrate the concept of instantiation. Recall one of our representations for the doctrine cjm-battery/contact:

```
(type  cjm-battery/contact
       situation:direct-contact)
(element
       cjm-battery/contact/direct-contact
            (s-object  contact-event:c
                       p),
            (agent  c  d))
```

If we were to find among the elements of facts-at-hand facts such as

```
(s-object  hit:h  person:Sam-Soe)
```

and

```
(agent  h  person:Tom-Toe),
```

then we would say that cjm-battery/contact is instantiated by these two

facts. Notice the similarity between this process and the instance relation. In a loose sense, instantiation is a matter of finding "instances" of more general situations.

6.31 Instantiating a Fact

Suppose we wish to instantiate a single fact stated as:

(relation-R thing-A thing-B),

which we will refer to as fact-F. For the present, we will assume that thing-A and thing-B are atomic, that is, they are not themselves facts or situations. It should be clear that any fact of the form:

(relation-R thing-A:a thing-B:b)

is an instantiation of fact-F. Now assume that we know, from the kind hierarchy, that relation-RR is a kind of relation-R, that thing-AA is a kind of thing-A, and that thing-BB is a kind of thing-B. Then each of the following facts also instantiates fact-F:

(relation-R thing-A:a thing-BB:bb)

(relation-R thing-AA:aa thing-B:b)

(relation-RR thing-A:a thing-B:b)

(relation-R thing-AA:aa thing-BB:bb)

et cetera.

For example, the fact:

(s-object contact-event person):f

in instantiated by the fact:

(s-object strike:s person:p)

because the event strike is a kind of contact-event. Fact f is also instantiated by the fact:

(s-object hit:h person:p)

because the event hit is a kind of strike, which is a kind of contact-event. Let us abbreviate the idea "kind of a kind of a kind of a . . ." with the symbol kind*. When we say that thing-TT is a kind* of thing-T, we mean that thing-TT is the same as thing-T, or is a kind of thing-T, or is a kind of a kind of thing-T, et cetera. Then we can express all of the above forms of instantiation by the following rule.

The fact:

(relation-R thing-A thing-B):fact-F

is instantiated by any fact of the form:

(relation-RR thing-AA:aa thing-BB:bb):fact-FF,

where relation-RR is a kind* of relation-R, thing-AA is a kind* of thing-A, and thing-BB is a kind* of thing-B.

We will call the above form of instantiation syllogistic, because the things and relation within fact-FF fit within the scope of the categorized things and relation expressed in fact-F. Thus we might say that "Socrates, who is a man, is mortal: is a syllogistic instantiation of "All men are mortal."

When the fact to be instantiated is part of the doctrine expressed in a case holding, we will also consider instantiation by analogy. When we say that thing-TT is an analog of thing-T we mean that each is a kind of the same thing. For example, touch is an analog of strike because each is a kind of contact-event. An instantiation by analogy is the same as an instantiation by syllogism, as defined above, except that one or more of the kind* relations in that definition is replaced by the double relation: kind*-of-an-analog-of. An example:

(instrument strike weapon)

is instantiated analogously by

(instrument hit:h wrench:w),

bacause a wrench is a kind of a tool, and a tool is an analog of a weapon (these are both kinds of movable-objects).

It should be emphasized that this method is a simplification of the general process of analogy. As we mentioned in Chapter 2, the soundness of an analogy depends on aspects of reasoning that lie outside the process of logical analysis as we have defined it for this study. Whenever instantiation by analogy is employed, the user will be told, so that he or she may supply this reasoning. In order to reduce the number of unsound analogies, the process will be restricted to the lower part of the kind hierarchy. For example, mental-attitude is not likely to be a very useful analog to family-relation, although they are both kinds of feature-relations. Exactly where these restrictions should be placed can best be determined from experience with the system.

Now let us consider the case in which the fact to be instantiated is compound, that is, one (or both) of the things in the fact is itself a fact. We take the following as a prototype:

```
(relation-R  thing-A:a
             (relation-S  thing-B:b
                          thing-C:c)):fact-G.
```

the same rule described above for fact-F is applied to all of the things and relations in fact-G. In addition, however, we must now begin to pay attention to the commonality of instance names within the fact to be instantiated. Common instance names add restrictions that can eliminate a possible instantiation. Consider the following fact, which is part of the doctrine of <u>cjm-assault/apprehension</u>:

```
(apprehension  person:p
               (s-object  contact-event:c  person:p)):f,
```

i.e., a person is apprehensive that he or she is going to be the semantic object of a contact event. The instantiation rules presented thus far imply that fact <u>f</u> is instantiated by:

```
(apprehension  person:Harry-Hoe
               (s-object  hit:h  person:Jerry-Joe)),
```

i.e., Harry Hoe is apprehensive that someone else, Jerry Joe, will be hit. Here we have ignored the requirement, expressed by the common instance name p within fact f, that both instances of person be the same. This is the precise reason for using instance names in our generalized representations.

Therefore, we must add one further rule for instantiating a fact: Fact-F is instantiated by Fact-FF only if all commonalities of instances appearing in Fact-F appear also in Fact-FF.

Next, consider the case in which one (or both) of the things in a fact to be instantiated is a situation. In this case, we first instantiate the situation (using one of the methods described in the next section), and then we instantiate the rest of the fact, following the rules described above.

How does the system look for facts that may instantiate a given fact? It would be inefficient to examine every fact in the system's memory and to determine whether or not each obeys the instantiation rules. Instead, the system uses OWL back-pointers to restrict searches of this kind. We mentioned in Chapter 4 that every OWL data-item (i.e., every thing and every relation) is provided with back-pointers that indicate all of the facts of which that data item is a part. This serves as an index--a listing of all of the locations of that data item within the entire data structure. In its search for instantiations, the system first consults the back-pointers for the situation facts-at-hand, since only facts

appearing in the context:

(element facts-at-hand fact-FF)

are valid candidates for instantiation. There would be not more than perhaps 100 such facts. Within the limited domain of these facts, the back-pointers of relation-R can be consulted to locate only those facts in which relation-R, or a kind* of relation-R, or an analog of relation-R, appears. Even with kinds and analogs, there are not likely to more than a handful of such facts. Only this small number of facts is examined in detail to determine whether or not each fact is an instantiation of fact-F.

An attempt to instantiate fact-F succeeds when one or more of the facts thus examined obeys the rules of instantiation with respect of fact-F. We then say that the fact is instantiated. If no instantiating facts are found, the system tries to instantiate the logical negative of fact-F, i.e.: (not fact-F). If this attempt succeeds, we say that fact-F is counter-instantiated. If both of these attempts fail, we say that fact-F is non-instantiated.

6.32 Instantiating a Situation

There are three basic methods by means of which a situation can be instantiated: by instantiating all of its elements, by instantiating one (or more) of its types, or by instantiating one (or more) of its examples.

The rule for instantiation by elements can be stated more fully: Situation-S is instantiated when all of its elements are instantiated and all of its counter-elements are counter-instantiated. Situation-S is counter-instantiated when one (or more) of its elements is counter-instantiated or one (or more) of its counter-elements is instantiated. Facts containing the relation instance or the relation is are not considered elements for purposes of instantiation by elements. However, the commonalities of instances appearing in situation-S must be preserved. In other words, wherever two or more instances in situation-S are identical, the corresponding instances in the instantiation (or counter-instantiating) elements must also be identical.

The rule for instantiation by types can be stated thus: Situation-S is instantiated when at least one of its types is instantiated. Situation-S is counter-instantiated when at least one of its counter-types is instantiated. The rule for instantiation by examples is logically equivalent to the rule for instantiation by types. Recall that we are distinguishing types form examples only because examples are attributable to separate, primary sources of legal authority.

Let us return to the example of instantiation presented on page 154. It illustrates both instantiation by elements and instantiation by types. In the example, we are able to instantiate the situation cjm-battery/contact/direct-contact by instantiating its two elements. The

first element:

(s-object contact-event:c p)

is instantiated by the fact:

(s-object hit:h person:Sam-Soe).

This is a syllogistic instantiation, since hit is a kind of contact-event. The second element:

(agent c d)

is instantiated by the fact:

(agent h person:Tom-Toe).

Because the instance c is used within both elements of the situation to be instantiated, a similar commonality of instances is required in the set of facts instantiating this situation. This requirement is satisfied by the common use of the instance h.

Once the situation cjm-battery/contact/direct-contact is seen to be instantiated by elements, the situation cjm-battery/contact is seen to be instanticated by types, namely, by the instantiation of one type of cjm-battery/contact.

The logical rules for instantiating and counter-instantiating a situation are summarized in the following table. When a situation cannot be instantiated nor counter-instantiated, we say it is non-instantiated. Notice that it is possible for the same situation to be both instantiated and counter-instantiated. This can occur, for example, when two cases assert conflicting doctrine. Under these conditions, the same situation can be instantiated by one example and counter-instantiated by a counter-

example. (We will not explore such conflicts any further in this study.)

	it is necessary		
In order to instantiate:	to instantiate:		to counter-instantiate:
by element	all elements	and	all counter-elements
by type	≥1 type		--
by example	≥1 example		--
In order to counter-instantiate:			
by element	≥1 counter-element	or	≥1 element
by type	≥1 counter-type		--
by example	≥1 counter-example		--

6.33 Instantiation by Query

It can be seen from the previous two sections that the successful instantiation (or counter-instantiation) of a situation like cjm-battery depends ultimately on the successful instantiation (or counter-instantiation) of facts. When a fact cannot be instantiated (or counter-instantiated), it is because there is no fact in facts-at-hand that instantiates it (or its logical negative). Of course, we cannot expect that the user will include in facts-at-hand every fact that might possibly

be needed for instantiation. Instead, we will provide the means for the system to ask the user about additional facts whenever it reaches such an impasse.

In particular, immediately after an attempted instantiation by elements of situation-S, if the situation is non-instantiated, then each of the facts that, as elements of situation-S, were themselves non-instantiated will be displayed to the user with the query: "Is it the case that fact-F?" Fact-F is the non-instantiated fact, except that the instance names displayed are those names that have the proper commonalities with the instance names appearing in the successful instantiations of other elements of situation-S.

The user responds to each query in one of four ways: "yes," "no," "assume so," or "assume not." The fact becomes instantiated by a positive response. It becomes counter-instantiated by a negative response. When the response involves an assumption, the system makes an internal "note" that the instantiation (or counter-instantiation) is based on an assumption. When the system describes its analysis to the user, it repeats these assumptions.

This method of instantiation is perhaps more easily understood from an example. Suppose we were trying to instantiate, by elements, the situation cjm-assault/apprehension:

```
(element  cjm-assault/apprehension
     (apprehension  person:p  (s-object  contact-event:c  p)):f1,
     (cause  f1  event:e):f2,
     (agent  e  person:d):f3).
```

Assume that <u>facts-at-hand</u> contains only the following relevant facts:

```
     (apprehension  person:Kate-Koe
                    (s-object  hit:h  Kate-Koe)):f4,
     (agent  raise:r  person:Larry-Loe):f5,
     (s-object  r  golf-club:g).
```

We see that <u>f1</u> is instantiated by <u>f4</u>, and that <u>f3</u> is instantiated by <u>f5</u>. However, fact <u>f2</u> is non-instantiated, and therefore the situation as a whole is non-instantiated. In such circumstances, the system asks the user about fact <u>f2</u>: "Is it the case that (cause (apprehension Kate-Koe (s-object h Kate-Koe)) r)?" An English version of this question would be: "Was Kate Koe's apprehension of being hit caused by Larry Loe's raising the golf club?" Notice that the instance names from the other instantiating facts (e.g., <u>Kate-Koe</u>, <u>h</u>, <u>r</u>) are used in the query.

If the answer to this query is "yes," then the fact:

```
(cause  f4  r)
```

is inserted as an element of facts-at-hand. If the answer is "assume so," then the fact:

(assumption user (cause f4 r))

is inserted instead. In either case, the fact f2 is thereby instantiated, and, since the commonality of instances is correct, the situation cjm-assault/apprehension is instantiated by elements. If the answer to the query is "no" or "assume not," then the fact:

(not (cause f4 r))

or the fact:

(assumption user (not (cause f4 r)))

is inserted as an element of facts-at-hand. In either of these cases, the fact f2, and therefore the situation as a whole, is counter-instantiated.

6.34 The General Process of Instantiation

We have established several rules to govern the instantiation of facts and situations. We now will examine the machine procedure that implements these rules. This procedure is called instantiate. It takes as its argument a fact or a situation that represents a piece of legal doctrine. It attempts to instantiate its argument with respect to the specific facts contained in the situation facts-at-hand.

Each time the procedure instantiate is invoked, there is one of three possible results: its argument is instantiated, or counter-instantiated, or non-instantiated. The result is determined by the rules described in

the previous three sections. Under these rules, it is possible to attempt several different instantiations of a given situation or fact. It is also possible that several different instantiations (or counter-instantiations) will result for the same situation or fact. For example, a single situation might be instantiated by elements, as well as by one or more types or examples. A single fact might be instantiated separately by several facts in facts-at-hand, some syllogistically, perhaps, and others analogously.

Should we allow the procedure instantiate exhaustively to explore every instantiation that is possible under these rules? If we were to follow this approach, the computer might spend a great amount of its time exploring instantiations that are unnecessary. Should we therefore terminate the instantiation of a fact or situation as soon as a single instantiation is found? This approach entails the risk of bypassing instantiations that turn out to be necessary.

The problem of bypassing necessary instantiations can best by understood from an example. Suppose we were attempting to instantiate, by its elements, the situation cjm-battery/contact/direct-contact:

```
(element  cjm-battery/contact/direct-contact
            (s-object  contact-event:c  person:p):f1,
            (agent  contact-event:c  person:d):f2).
```

Now assume that facts-at-hand includes the following elements:

(s-object hit:h1 person:Sam-Soe):f3,

(agent hit:h2 person:Vaughn-Voe):f4,

(agent hit:h1 person:Tom-Toe):f5.

We see that f1 is instantiated by f3, and that f2 is instantiated twice by f4 and by f5. However, for purposes of instantiating cjm-battery/contact/direct-contact, facts f3 and f5 taken together have the proper commonality of the instance of hit (namely, h1), whereas the facts f3 and f4 taken together do not. The fact that f4 instantiates f2 becomes irrelevant for the instantiation of this situation as a whole. If we had terminated the instantiation of f2 as soon as we had found f4, we would have bypassed the instantiation of f5, which instantiation is necessary for the instantiation of cjm-battery/contact/direct-contact.

In the prototype system, the total number of examples, types, and elements subject to instantiation is relatively small. This means we can follow a low-risk, more time-consuming approach. With two exceptions, the procedure instantiate will try to find as many instantiations of a given fact or situation as possible. One exception relates to instantiation by query. It will be our policy to bother the user with instantiation queries only as a last resort, i.e., when no other means of instantiation is successful. We will implement this policy with the following procedural rule:

> If all possible attempts to instantiate situation-S by examples and by types have resulted in non-instantiations, and if Situation-S is not an example or a type of a situation for which instantiation possibilities, without query, still exist, and if either there is only one element (or counter-element) in Situation-S or else at least one element of situation-S has been instantiated (or one counter-element has been counter-instantiated), and if the commonalities of instances appearing thus far among the instantiating elements (and counter-elements) are in accord with the commonalities in situation-S, then--and only then--one of the non-instantiated facts will be presented for query. If the above conditions continue to hold true after one query and response, another non-instantiated fact from situation-S will be presented for query, and so on.

The second exception relates to multiple instantiations based on the same facts from facts-at-hand. The rule can be stated: When an instantiation of a situation by type or by elements is based upon (i.e., ultimately is instantiated by) the same facts from facts-at-hand, or on a subset of such facts, that instantiate an example of that situation, the instantiation by type or by elements is ignored (deleted). We include this rule because an instantiation by example carries the weight of primary legal authority. Where we have such an instantiation, any further instantiation by the same facts, but based on secondary authority, serves no purpose. It should be clear that there is no risk bypassing a necessary instantiation, because the redundant instantiation(s) involve the same facts as the retained instantiation. In a more comprehensive analysis system, other procedural rules would have to be developed to limit multiple instantiations, while keeping the risk of bypassing necessary instantiations as low as possible. One possible technique is to look

initially for only a single instantiation, and then to return to look for another only if the first instantiation becomes eliminated due to improper instance commonality. Another time-saving technique is the labeling of some of the elements of a type as indicator (or counter-indicator) elements. These would be the elements that characterize that particular type. Whenever the indicator elements of a type are not instantiated, the instantiation of that could be abandoned. The desirability of using the various techniques like these can be better determined after gaining some experience with the prototype system.

With the above observations in mind, we can set forth the full scenario of the instantiation process. When instantiate is called upon to instantiate a situation, it first tries to instantiate each example of that situation that is known to the system. It then tries to instantiate each type known to the system. Finally, it tries to instantiate each element known to the system. The order in which it tries these three methods is chosen to facilitate our rules regarding multiple instantiations. For convenience, we will separate these three methods into three sub-procedures called instantiate-by-examples, instantiate-by-types, and instantiate-by-elements.

The first two sub-procedures operate exhaustively, that is, they pursue every example and type, regardless of the results of any prior instantiations. The order in which they examine the individual examples and the individual types is arbitrary. Instantiate-by-elements attempts the instantiation of every element of the situation, again in an arbitrary order. It queries the user about a fact only when the conditions

stated on page 169 are met. The operations of these three sub-procedures result in instantiation, counter-instantiation, or non-instantiation, according to the rules of logic and of instance commonality, as discussed in section 6.32.

When instantiate is called upon to instantiate a fact, it first instantiates any situations that may be contained in the fact. It then follows the rules of syllogism and of analogy, and the rule of instance commonality, as discussed in section 6.31.

When instantiate is called upon to instantiate a situation, or a fact containing a situation, the procedure or one of its sub-procedures must perform subordinate instantiations. To do this, other instances of the procedure instantiate are invoked. This process, in which one application of a procedure uses other applications of the same procedure is called recursion. The recursion terminates when the argument of instantiate is a fact that contains no further situations.

The process of instantiation begins with the most general legal doctrine known to the system. In the prototype, this doctrine is cjm-intentional-tort. Thus, immediately after invoking the procedure listen-to-facts (by which means the user's hypothethical facts are inserted as elements of facts-at-hand), the system invokes the command:

(execute instantiate cjm-intentional-tort).

There are no examples or elements of cjm-intentional-tort known to the system. Therefore, only instantiate-by-types can be used for this instantiation. Accordingly the system then attempts to instantiate cjm-battery and cjm-assault. Each of these is instantiated or

counter-instantiated, by examples, types, and elements, as is appropriate.

6.35 The Record of Instantiation

There is one other function that the procedure instantiate must perform in addition to instantiation itself. It must keep a record of the instantiations and counter-instantiations that have been found. This record serves two purposes. It informs instantiate as to what it has already done, so that the procedure knows what to do next. Then, after the analysis is finished, it provides a history of the analysis that allows the procedure called discuss-analysis (to be described in section 6.4) to explain to the user how the analysis was performed.

When the OWL system has been fully developed, it will include an automatic mechanism for recording the history of the OWL procedures that were performed in a given machine session. For purposes of this study, however, we will keep track of instantiation by inserting facts called instantiation relationships into the system's memory. Every time instantiate attempts an instantiation, it will insert a fact of the following form:

(instantiation-relation th1 th2).

There are 16 kinds of instantiation-relation, corresponding to instantiation, counter-instantiation, and non-instantiation, by type, by example, by syllogism, by analogy, and by assumption:

```
(kind  instantiation-relation

     inst-element,    c-inst-element,    n-inst-element,
     inst-type,       c-inst-type,       n-inst-type,
     inst-example,    c-inst-example,    n-inst-example,
     inst-syllogism,  c-inst-syllogism,  n-inst-fact,
     inst-analogy,    c-inst-analogy,
     inst-assume,     c-inst-assume).
```

In such a relationship, th1, is the fact or situation that is instantiated. If th1 is a fact, then th2 is the fact from facts-at-hand that instantaites it. If th1 is a situation, and if the instantiation is by example or by type, then th2 is an instantiation relationship for an example or a type of th1. If the instantiation-relation indicates non-instantiation, then th2 is absent from the instantiation relationship.

To illustrate the creation of the instantiation record, let us re-examine a recent example. We wish to instantiate the situation cjm-battery/contact, given the following facts:

```
(type  cjm-battery/contact  situation:direct-contact)
(element  cjm-battery/contact/direct-contact
               (s-object  contact-event:c  person:p):f1,
               (agent  c  person:d):f2)
(element  facts-at-hand
               (s-object  hit;h1  person:Sam-Soe):f3,
               (agent  hit:h2  person:Vaughn-Voe):f4,
               (agent  hit:h1  person:Tom-Toe):f5)
```

(kind contact-event strike)

(kind strike hit).

After determining that f1 is instantiated by f3, instantiate would insert the following fact:

(inst-syllogism f1 f3):f6

Similarly, for the two instantiations of f2 there would be inserted:

(inst-syllogism f2 f4):f7

(inst-syllogism f2 f5):f8.

Recall that cjm-battery/contact/direct-contact is then instantiated (by elements) by the combination of facts f3 and f5. This is recorded by the insertion of two facts:

(inst-element cjm-battery/contact/direct-contact situation:s-inst-1):f9

and

(element s-inst-1 f6,f8).

Finally, cjm-battery/contact is instantiated by type. We therefore insert:

(inst-type cjm-battery/contact f9).

6.36 The Algorithms for Instantiation

The machine procedures for instantiation can be summarized by the algorithms shown in figures 6-2 through 6-6 on the following pages.

<u>start</u>

<u>Algorithm for instantiate</u>

If the argument of <u>instantiate</u> is a situation, s,
then
(execute instantiate-by-examples s)
(execute instantiate-by-types s)
(execute instantiate-by-elements s)
<u>return</u>

otherwise
the argument is a fact, f.

if f contains a situation, s', then (execute instantiate s')

find all syllogistic and analogous instantiations and counter-instantiations of f (see section 6.31)

retain only those instantiations and counter-instantiations that properly reflect the commonalities of instances (if any) appearing in f

for each such instantiating fact, i,

(execute insert (inst-syllogism f i)), or

(execute insert (inst-analogy f i)), or

(execute insert (c-inst-syllogism f i)), or

(execute insert (c-inst-analogy f i)),

whichever is appropriate

if there are no such instantiating facts,
then
(execute insert (n-inst-fact f))

<u>return</u>

Figure 6-2.

Algorithm for instantiate-by-examples

```
start
  |
  v
if there is an example, ex, of s for which instantiation has not been
                attempted from within this procedure,
otherwise       then
                (execute  instantiate  ex)
  |                 |
  |                 v
  |             if ex is instantiated, with instantiation relationship r,
  |             then
  |             (execute  insert  (inst-example  s  r))
  |
  v
if there is a counter-example, cex, of s for which instantiation has not
                been attempted from within this procedure,
otherwise       then
                (execute  instantiate  cex)
  |                 |
  |                 v
  |             if cex is instantiated, with instantiation relationship r,
  |             then
  |             (execute  insert  (c-inst-example  s  r))
  |
  v
if no instantiation of any example or any counter-example of s has been
                found,
                then
                (execute  insert  (n-inst-example  s))
  |
  v
return
```

Figure 6-3.

Algorithm for instantiate-by-types

```
start
  ↓
if there is a type, t, of s for which instantiation has not been
                attempted from within this procedure,
otherwise       then
                (execute  instantiate  t)
                   ↓
                if t is instantiated, with instantiation relationship r,
                then
                (execute  insert  (inst-type  s  r))
  ↓
if there is a counter-type, ct, of s for which instantiation has not
                been attempted from within this procedure,
otherwise       then
                (execute  instantiate  ct)
                   ↓
                if ct is instantiated, with instantiation relationship r,
                then
                (execute  insert  (c-inst-type  s  r))
  ↓
if no instantiation of any type or any counter-type of s has been
                found,
                then
                (execute  insert  (n-inst-type  s))
  ↓
return
```

Figure 6-4.

Algorithm for instantiate-by-elements

```
start
  |
  v
if there is an element, el, of s for which instantiation has not been
                attempted from within this procedure,
otherwise       then
                (execute  instantiate  el)
  |                 |
  |                 v
  |               if el is counter-instantiated, with inst. relationship r,
  |               then
  |               (execute  insert  (c-inst-element  s  r))
  |
  v
if there is a counter-element, cel, for which instantiation has not
                been attempted from within this procedure,
otherwise       then
                (execute  instantiate  cel)
  |                 |
  |                 v
  |               if cel is instantiated, with instantiation relationship r,
  |               then
  |               (execute  insert  (c-inst-element  s  r))
  |
  v
if all elements of of s have been instantiated and all counterelements of
                s have been counter-instantiated,
                then
                for each complete set, x, of instantiation relationships
                r1, . . . ,rn, for the elements and counter-elements of s,
                which set properly reflects the commonalities of instances
                (if any) appearing in s,
                    |
                    v
                (execute  insert  (inst-element  s  situation:s-inst-x),

                                  (element  s-inst-x  r1, . . . ,rn))
```

(Continued in figure 6-6.)

Figure 6-5.

(Continued from figure 6-5.)

```
if any non-instantiated element or counter-element of s is
a fact, f, and if the query conditions (page 169) are met,
     then
     (execute  display  "Is it the case that f'?")*
     (execute  receive  response)

if response is "Yes,"
     then
     (execute  insert  (element  facts-at-hand  f'),
                       (inst-syllogism  f  f'):f1)
     if f is a counter-element of s,
     then
     (execute  insert  (c-inst-element  s  f1))

if response is "No,"
     then
     (execute  insert  (element  facts-at-hand  (not  f')),
                       (c-inst-syllogism  f  (not  f')):f2)
     if  f is an element of s,
     then
     (execute  insert  (c-inst-element  s  f2))

if response is "Assume so,"
     then
     (execute  insert  (element  facts-at-hand
                                 (assumption  user  f')),
                       (inst-assume  f  f'):f3)
     if f is a counter-element of s,
     then
     (execute  insert  (c-inst-element  s  f3))

if response is "Assume not,"
     then
     (execute  insert  (element  facts-at-hand
                                 (assumption  user  (not  f'))),
                       (c-inst-assume  f  (not  f')):f4)
     if f is an element of s,
     then
     (execute  insert  (c-inst-element  s  f4))
```

* Fact f' is fact f with appropriate instance names.

Figure 6-6.

6.4 The Discussion Procedure

After the system has finished its instantiation process, it informs the user of the results of its analysis. The user primarily wants to know whether or not an intentional tort has been established (i.e., instantiated) or ruled out (i.e., counter-instantiated) by the hypothetical facts that were presented to the system. In particular, the user wants to know whether or not these facts establish or rule out the two torts of battery and assault, and the user often would like to know how the system reached it conclusions.

All of this information is available from the instantiation record that was created in the system's memory during the instantiation process. For example, by looking at the following instantiation relationships:

```
(inst-type  cjm-intentional-tort
                    (inst-element  cjm-battery  . . .))
(c-inst-element  cjm-assault
                    (c-inst-element  cjm-assault/intent . . .)),
```

We can see that battery had been established (instantiated by elements) and that, therefore, and intentional tort had been established (instantiated by type). Assault, however had been ruled out (counter-instantiated by elements) because of a lack of intent. By tracing through the rest of these instantiation relationships, we could determine exactly how each instantiation was obtained.

Clearly, this is an unmanageably awkward manner in which to learn the results of the machine's analysis. In its place, we will introduce a machine procedure called discuss-analysis that translates the instantiation record into an English-like discourse.

We mentioned at the outset that our study is not concerned with the actual process of translating between statements in the PSL language and statements in English. Accordingly, we will not examine the procedure discuss-analysis in detail. Instead, we will discuss briefly the general approach of the procedure, and we will illustrate this approach with a simple example.

The instantiation record is structured much like a tree. At the root of the tree is the instantiation of the general doctrine cjm-intentional-tort. The tree branches out by elements, by types, and by examples, ultimately reaching instantiations by the facts contained in facts-at-hand. The procedure discuss-analysis begins its discussion at the root of the tree, and then proceeds towards the outer branches, in response to request from the user for further explanation.

Let us re-examine the instantiation relationship that for cjm-battery/contact in the example from section 6.35. We can restate the relationship in its entirety as follows:

```
(inst-type  cjm-battery/contact
     (inst-element  cjm-battery/direct-contact
                    situation:s-inst-1))
```

```
(element  s-inst-1
     (inst-syllogism  (s-object  contact-event:c  person:p)
                      (s-object  hit:h1  person:Sam-Soe)),
     (inst-syllogism  (agent  contact-event:c  person:d)
                      (agent  hit:h1  person:Tom-Toe))).
```

A summary statement for this relationship might be effected by the command:

```
(execute  display  "Contact appears to be staisfied by the facts at
                   hand.")
```

If the user asks how contact is satisfied, the system can go into the details of the instantiation relationship:

```
(execute  display  "One form of contact involves direct contact.  When
                   there is contact to the plaintiff by the defendant,
                   there is contact as required for a battery.  This
                   appears to cover the facts at hand, in which Sam Soe
                   was hit by Tom Toe.")
```

In addition to the form of discussion illustrated here, the system can add appropriate phrases to remind the user when an instantiation is based on assumption ("according to your assumption that . . ."), and to inform the user when an instantiation is based on analogy ("The decision

in . . . provides an analogy"), since this information is recorded in the instantiation relationships. When instantiation is by example (case decision), the system also provides the user with the facts of the particular case whose doctrine is being instantiated. These aspects of the discussion procedure are illustrated in the examples described in Chapter 7.

When the discussion of analysis has been displayed to the user, the analysis session is concluded. If the user wishes to begin another analysis, all of the facts inserted during the previous session (the facts in facts-at-hand, and the instantiation relationships) first will be deleted, so that new facts can be presented for analysis.

Our machine procedure for legal analysis thus can be summarized by the following algorithm:

```
begin analysis
(execute  listen-to-facts)
(execute  instantiate  cjm-intentional-tort)
(execute  discuss-analysis)
end analysis.
```

Chapter 7 Examples of Analysis, Explained

We can now return to the examples presented in Chapter 1, examining more closely the details of the analysis. Remember that we are not concerned with the process of translation between statements in English and statements in PSL. We include the English representations to make the examples more comprehensible.

Analysis Session 1

USER: Aaron Aardvark purposely kicked Zachary Zetz in the leg.

The PSL representation for these simple facts looks like this:

```
(element  facts-at-hand                    In the facts at hand,
    (agent  kick:k
        person:Aaron-Aardvark),            Aaron Aardvark kicks
    (s-object  k
        person:Zachary-Zetz):f1,           Zachary Zetz
    (destination  k  leg:l),               in the leg
    (part  Zachary-Zetz  l),               of Zachary Zetz
    (purpose  k  f1))                      purposely.
```

As always, the instantiation process begins with an attempt to instantiate cjm-intentional-tort (by type), which leads to attempts to instantiate cjm-battery and cjm-assault (by elements). The contact component of battery is instantiated by the example provided in the case Foe v. Moe, as recorded by the following instantiation relationship:

```
(inst-example  cjm-battery/contact
     (inst-element  s-foe-v-moe  situation:s-inst-1)):r1
(element  s-inst-1
     (inst-syllogism  (agent  strike:s  person:d)
                      (agent  kick:k  person:Aaron  Aardvark)),
     (inst-syllogism  (s-object  s  person:p)
                      (s-object  k  person:Zachary Zetz)),
     (inst-syllogism  (destination  s  anatomical-part:a)
                      (destination  k  leg:l)),
     (inst-syllogism  (part  p  a)
                      (part  Zachary-Zetz  l))).
```

The intent component of battery is instantiated by elements:

```
(inst-element  cjm-battery/intent  situation:s-inst-2):r2
(element  s-inst-2
     (inst-syllogism  (agent  event:e  person:d)
                      (agent  kick:k  person:Aaron-Aardvark)),
```

```
(inst-syllogism  (or  (purpose  e  situation:result)
                      (belief  p  (cause  result  e)))
                 (purpose  kick:k
                      (s-object  k  person:Zachary-Zetz)))),
```

where

```
(element  result  (or  (s-object  contact-event:c  person:x):fl
                       (apprehension  x  fl))).
```

Notice that a disjunctive fact, (or fact-1 fact-2), is instantiated if either fact-1 or fact-2 is instantiated.

Cjm-battery/consent can be neither instantiated nor counter-instantiated from the facts currently contained in facts-at-hand. When an attempt is made to instantiate cjm-battery/consent by elements, the conditions for instantiation by query are met. (There is no other successful instantiation, and the situation contains only one element.) The user is therefore asked:

"Is it the case that

```
(consent  Zachary-Zetz
          (s-object  kick:k  Zachary-Zetz))?"
```

which might be stated in English as "Did Zachary Zetz consent to being kicked?" In this example, the user responds, "No." Cjm-battery/consent is thereby counter-instantiated:

```
(c-inst-element  cjm-battery/consent
     (c-inst-syllogism  (consent  p  (s-object  c  p))
                  (not  (consent  Zachary-Zetz
                                (s-object  k  Zachary-Zetz))))):r3.
```

Thus, cjm-battery becomes instantiated by elements:

```
(inst-element  cjm-battery  situation:s-inst-3):r4
(element  s-inst-3  r1,r2,r3).
```

Next, the eystem attempts to instantiate the other known type of cjm-intentional tort, namely, cjm-assault. Proceeding again by elements, the system attempts to instantiate cjm-assault/apprehension and cjm-assault/intent. Apprehension cannot be instantiated (or counter-instantiated) by examples or by types. The only element of apprehension that can be instantiated is (agent event:e person:d), as follows:

```
(inst-syllogism  (agent  event:e  person:p)
                 (agent  k  Aaron-Aardvark)).
```

The condition for instantiation by query is thereby met with regard to other elements of apprehension. (There is no other successful instantiation of apprehension, and at least one element of apprehension has been instantiated). The system therefore asks the user:

"Is is the case that

(apprehension Zachary-Zetz

(s-object k Zachary-Zetz)) ?"

which might be stated in English as, "Was Zachary Zetz apprehensive that he would be kicked?" In this example, the user responds, "Assume not." Cjm-assault/apprehension is thereby counterinstantiated:

```
(c-inst-element  cjm-assault/apprehension
     (c-inst-assume  (apprehension  p  (s-object  c  p))
                (not  (apprehension  Zachary Zetz
                          (s-object  k  Zachary-Zetz)))))):r5.
```

Because the situation is now counter-instantiated, the condition for instantiation by query is no longer met. The remaining element of apprehension (the cause of the apprehension) is therefore not explored. Furthermore, cjm-assault has itself become counter-instantiated:

(c-inst-element cjm-assault r5):r6.

Finally, cjm-intentional-tort is instantiated by type:

(inst-type cjm-intentional-tort r4).

The procedure discuss-analysis now provides the user with summary statements about the instantiation of battery and assault:

SYSTEM: Aaron Aardvark appears to be liable to Zachary Zetz for a battery. Aaron Aardvark appears not to be liable to Zachary Zetz for an assault.

If the user asks why there is liability for a battery, the system summarizes the instantiation relationship for cjm-battery (r4, above):

USER: Why is there liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and the lack of consent. These components appear to be satisfied by the facts at hand.

Then, if the user asks how each of these components, in turn, is satisfied, the system goes into the details of the instantiation relationship for each component (r1, r2, and r3, above):

USER: How is contact satisfied?

SYSTEM: The decision in Foe v. Moe is applicable. In that case, Joe Moe punched Fred Foe in the nose. It was held that when one person strikes the anatomy of another, there is contact as required for a battery. This appears to cover the facts at hand, in which Aaron Aardvark kicked Zachary Zetz in the leg.

USER: How is intent satisfied?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or the apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for a battery. This appears to cover the facts at hand, in which Aaron Aardvark kicked Zachary Zetz purposely.

USER: How is lack of consent satisfied?

SYSTEM: When a person consents to being contacted, there is consent as will avoid a battery. This appears to be contradicted by the facts at hand, in which Zachary Zetz did not consent to being kicked.

The counter-instantiation of cjm-assault is explained in a similar manner. First, relationship r6 is summarized, and then, if the user so requests, the details of relationship r5 are discussed:

USER: Why is there no liability for an assault?

SYSTEM: According to the doctrine adopted in Jones v. Smith, an action in assault is established by two components: apprehension and intent. Apprehension appears to be contradicted by the facts at hand.

USER: How is apprehension contradicted?

SYSTEM: When a person is apprehensive of being contacted, there is apprehension as required for an assault. This appears to be contradicted by the facts at hand, in which, according to your assumption, Zachary Zetz was not apprehensive that he would be kicked.

Notice that the user is reminded that the absence of Zachary's apprehension was an assumption. Notice also that when Foe v. Moe was cited, the specific facts as well as the holding were recited.

Analysis Session 2

USER: Fred Dobbs and Benjie Hooray are playing in a hockey game. Fred raises his hockey stick. Fred believes that this will cause Benjie to be apprehensive that he will be hit. Benjie is not apprehensive. Fred hits Benjie in the leg with his hockey stick, but not purposely.

The PSL representation for these facts looks like this:

(element facts-at-hand)	*In the facts at hand,*
(agent hockey-game:h	*a hockey game is played*
person:Fred-Dobbs,	*by Fred Dobbs*
	and
person:Benjie-Hooray),	*by Benjie Hooray,*
	and
(agent raise:r Fred Dobbs),	*Fred Dobbs raises*
(s-object r hockey-stick:hs),	*the hockey stick*
(possessor hs Fred-Dobbs),	*of Fred Dobbs,*

	and
(belief Fred-Dobbs	*it is the belief of Fred Dobbs*
	that
(cause (apprehension	*apprehension*
Benjie-Hooray	*by Benjie Hooray*
(s-object hit:h	*of a hit*
Benjie-Hooray)):f1	*to Benjie Hooray*
r)),	*will be caused by the raising,*
	and
(not f1),	*there is no such apprehension,*
	and
(agent hit:h2 Fred-Dobbs),	*Fred Dobbs hits*
(s-object h2 Benjie-Hooray):f2,	*Benjie Hooray*
(instrument h2 hs),	*with his hockey stick*
(destination h2 leg:1),	*in the leg*
(part Benjie-Hooray 1),	*of Benjie Hooray*
(not (purpose h2 f2)))	*not purposely.*

In the attempt to instantiate battery (again, by elements), the contact component is instantiated by the example provided in Foe v. Moe in the same manner as that described in the previous analysis.

The intent component can be instantiated despite the fact that Fred Dobbs did not purposely hit Benjie Hooray. This is because Dobbs did believe that the raising of this hockey stick would cause Hooray to become apprehensive that he would be hit:

```
(inst-element  cjm-battery/intent  situation:s-inst-1)
(element  s-inst-1
     (inst-syllogism  (agent  event:e  person:d)
                      (agent  raise:r  person:Fred-Dobbs),
     (inst-syllogism  (or  (purpose  e  situation:result)
                           (belief  p  (cause  result  e)))
                      (belief  Fred-Dobbs
                         (cause  (apprehension  Benjie-Hooray
                           (s-object  hit:h1  Benjie-Hooray))
                                r)))),
```

where

```
(element  result  (or  (s-object  contact-event:c  person:x):f
                       (apprehension  x  f))).
```

The consent component of battery is instantiated by type, namely, by the situation representing inferred consent. This type is instantiated in turn by the example provided by the case Quoe v. Poe. Recall that the elements of the holding of that case are:

```
(agent  athletic-competition:a  person:p1):f1,
(agent  a  person:p2):f2,
(s-object  contact-event:c  p2):f3,
(part  a  c):f4,
(is  cjm-battery/p  p2), and
(is cjm-battery/d  p1).
```

Elements f1 and f2 each can be instantiated twice by facts in facts-at-hand:

```
(inst-syllogism  (agent  athletic-competition:a  d)
                 (agent  hockey-game:h  Fred-Dobbs)):r1
(inst-syllogism  (agent  a  d)  (agent  h  Benjie-Hooray)):r2
(inst-syllogism  (agent  a  p)  (agent  h  Fred-Dobbs)):r3
(inst-syllogism  (agent  a  p)  (agent  h  Benjie-Hooray)):r4.
```

Element f3 is instantiated once:

```
(inst-syllogism  (s-object  contact-event:c  p)
                 (s-object  hit:h2  Benjie-Hooray)):r5.
```

Element f4 can be neither instantiated nor counter-instantiated from the facts currently in facts-at-hand, however, the condition for instantiation by query is met. (There is no other way to instantiate this example or inferred consent itself, and at least one of the elements of this example has already been instantiated.) The system therefore asks the user: "Is it the case that (part h h2) ?" which might be stated in English as, "Was the hit that Benjie Hooray received a part of the hockey game?" In this example, the user responds, "Assume so." Element f4 is thereby instantiated:

(inst-assume (part a c) (part h h2)):r6.

The example has now been instantiated by elements:

(inst-element s-quoe-v-poe situation:s-inst-2)

(element s-inst-2 r1,r4,r5,r6).

Notice that instantiation realtionships r2 and r3 contain improper instance commonalities vis-a-vis the other instantiating facts for this example. These instantiations are therefore irrelevant.

Battery has now been successfully instantiated by elements, and it is given an instantiation relationship similar to that illustrated for the previous analysis. Assault is again counter-instantiated by the absence of apprehension, but this time without the need for a query to, or an assumption by, the user.

The English discussion of this instantiation record will therefore look something like this:

SYSTEM: Fred Dobbs appears not to be liable to Benjie Hooray for a battery. Fred Dobbs appears not be liable to Benjie Hooray for an assault.

USER: Why is there no liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and the lack of consent. Lack of consent appears to be contradicted by the facts at hand.

USER: How is lack of consent contradicted?

SYSTEM: One form of consent involves inferred consent. The decision in Quoe v. Poe is applicable. In that case, the parties were fighting in a boxing match. Perry Poe punched Quentin Quoe in the jaw. It was held that when two persons participate in an athletic competition, consent as will avoid a battery may be inferred for any physical contact that is part of the competition. This appears to cover the facts at hand, in which Fred Dobbs and Benjie Hooray were playing in a hockey game, and in which, according to your assumption, the hit that Benjie Hooray received was a part of the hockey game.

USER: Why is there not liability for an assault?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in assault is established by two components: apprehension and intent. Apprehension appears to be contradicted by the facts at hand.

USER: How is apprehension contradicted?

SYSTEM: When a person is apprehensive of being contacted, there is apprehension as required for an assault. This appears to be contradicted by the facts at hand, in which Benjie Hooray was not apprehensive that he would be hit.

Analysis Session 3

USER: With the purpose of frightening Gordon Good, Howard Hood visibly points a saturday-night special at him and grabs the umbrella that he is holding. The saturday-night special is not loaded.

The term "saturday-night special" is not recognized by the procedure listen-to-facts. (We purposely left it out of the kind hierarchy.) The system therefore asks the user:

SYSTEM: What is a saturday-night special?

USER: A saturday-night special is a kind of pistol.

The PSL representation for the facts at hand can then be inserted:

(element facts-at-hand	*In the facts at hand,*
(agent point:pt	
person:Howard-Hood):f1,	*Howard Hood points*
(s-object pt	
saturday-night-special:n):f2,	*a saturday-night special*
(destination pt	
person:Gordon-Good):f3,	*at Gordon Good*
(purpose pt (apprehension	*with the purpose of achieving apprehension*
Gordon-Good	*of Gordon Good*

(s-object contact-event:c	*of a contact*
Gordon-Good)):f4):f5,	*to Gordon Good, and*
(agent grab:g Howard-Hood),	*Howard Hood grabs*
(s-object g umbrella:u),	*an umbrella*
(held-by Gordon-Good u),	*held by Gordon Good*
(purpose g f4):f6,	*with the same purpose, and*
(loadedness n unloaded),	*the saturday-night special is unloaded, and*
(perception Gordon-Good	*it is perceived by Gordon Good*
situation:scene))	*that*
(element facts-at-hand/scene	*Howard Hood points a saturday-night special*
f1,f2,f3)	*at Gordon Good.*

Battery will be instantiated again by its elements. First, contact must be instantiated. Neither party is the semantic object of any contact event, although Good's umbrella was grabbed by Hood. In attempting to instantiate contact by example, the system will encounter the case of Roe v. Doe, which holds the striking of an article of clothing on a person is sufficient to establish contact. Now, according to our kind hierarchy, an umbrella is not an article of clothing; it is a personal accessory. However, an article of clothing is an analog of a personal accessory; they are both kinds of movable objects. Similarly, grab is an analog of strike; they are both kinds of contact events. Finally, the relations on and held-by are analogs because they are both kinds of contact relations. Thus, the holding of Roe v. Doe can be instantiated by three instantia-

tions by analogy:

```
(inst-example  cjm-battery/contact
     (inst-element  s-roe-v-doe  situation:s-inst-1))
(element  s-inst-1
     (inst-analogy  (agent  strike:s  person:d)
                    (agent  grab:g  Howard-Hood)),
     inst-analogy   (s-object  s  article-of-clothing:a)
                    (s-object  g  umbrella:u)),
     (inst-analogy  (on  p  a)  (held-by  Gordon-Good  u))).
```

Intent for battery is instantiated directly by fact f6 in the manner explained in the first two analyses. Consent must once again be counter-instantiated by query:

SYSTEM: "Did Gordon Good consent to the grabbing of his umbrella?"

USER: "No."

Next we examine the instantiation (by elements) of assault. Apprehension can be instantiated by type, namely a gun threat. In particular, an attempt will be made to instantiate the example provided by the Zoe v. Woe, in which a gun threat is made with an unloaded gun. Every element of s-zoe-v-woe can be can be instantiated syllogistically from the facts in facts-at-hand, with one execption. The fact:

(knowledge p (loadedness f unloaded))

cannot be instantiated or counter-instantiated. The user is therefore queried:

SYSTEM: "Did Gordon Good know that the saturday-night special was unloaded?"

USER: "No.:

The counter-instantiation of this fact completes the instantiation of apprehension. The intent component of assault is instantiated directly by the fact f5, just as the intent component of battery was instantiated by fact f6. AS a result, both battery and assault are instantiated in this analysis.

The English discussion of this instantiation record would look something like this:

SYSTEM: Howard Hood appears to be liable to Gordon Good for a battery. Howard Hood appears to be liable to Gordon Good for an assault.

USER: Why is there liability for a battery?

SYSTEM: According to the doctrine adopted in Smith v. Jones, an action in battery is established by three components: contact, intent, and lack of consent. These components appear to be satisfied by the facts at hand.

USER: How is contact satisfied?

SYSTEM: The decision in Roe v. Doe provides an analogy. In that case, John Doe knocked off the hat that Richard Roe was wearing. It was held that when a person strikes an article of clothing on the person of another, there is contact as required for a battery. This comes close to covering the facts at hand, in which Howard Hood grabbed the umbrella that Gordon Good was holding.

USER: How is intent satisfied?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or the apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for a battery. This appears to cover the facts at hand, in which Howard Hood grabbed the umbrella held by Gordon Good with the purpose of making him apprehensive of a physical contact.

USER: How is lack of consent satisfied?

SYSTEM: When a person consents to being contacted, there is consent as will avoid a battery. This appears to be contradicted by the facts at hand, in which Gordon Good did not consent to the grabbing of his umbrella.

USER: Why is there liability for an assault?

SYSTEM: According to the doctrine adopted in Jones v. Smith, an action in assault is established by two components: apprehension and intent. These components appear to be satisfied by the facts at hand.

USER: How is apprehension satisfied?

SYSTEM: One form of apprehension involves a threat with a gun. The decision in Zoe v. Woe is applicable. In that case, Winnie Woe visibly pointed a rifle at Zeke Zoe. The rifle was not loaded, but Zeke Zoe did not know that. It was held that where a person visibly points an unloaded firearm at another person, the other person's apprehension as required for an assault is established if the other person does not know that the firearm is unloaded. This appears to cover the facts at hand, in which Howard Hood visibly pointed a saturday-night special at Gordon Good, and in which the saturday-night special was unloaded, and in which Gordon Good did not know that it was unloaded.

USER: How is intent satisfied?

SYSTEM: When a person acts (a) with the purpose of achieving a physical contact to someone else, or an apprehension of such a contact, or (b) in the belief that such contact or apprehension will be caused by the act, there is intent as required for an assault. This appears to cover the facts at hand, in which Howard Hood pointed the saturday-night special at Gordon Good with the purpose of making him apprehensive of a physical contact.

Chapter 8 Concluding Discussion

In the previous chapter, we examined three examples in which the prototype system was able to produce sensible looking analyses for some simple factual situations. These examples were rather carefully drawn to illustrate those aspects of analysis that the system performs fairly well. To put these examples in perspective, we must try to delineate more carefully the charactersitic abilities and limitations of the prototype system. This will help us to understand the extent to which the prototype can be extended to wider areas of application.

At the heart of this study is the contention that the structural representations that have been developed for the prototype system are particularly well suited for the task of computer-aided legal analysis, as we have defined it. Our structural model is more complex and more comprehensive than many of the early forms of data structure used by researchers in artificial intelligence and in analytical jurisprudence. It is less comprehensive, and less linguistically oriented, than some of the more general purpose comprehension models, like that of the OWL system, that are under current development. This mid-level realm of structural complexity has been largely unexplored. Yet it appears to be well matched to the inherent categorization and structure with which legal doctrine is constructed.

The prototype system contains representations for only two areas of legal doctrine, the intentional torts called battery and assault, and even for these areas, some of the component aspects (like the defense of

privilege) have been omitted. The same structural technique that we have used for battery and assault, however, can be applied throughout large bodies of legal doctrine. False imprisonment, an intentional tort that we did not examine, includes a component situation in which the plaintiff is confined within boundaries fixed by the defendant, in the place of the components of physical contact or apprehension of contact.

Unintentional torts involving negligence comprise four component elements: a duty to behave in a certain manner toward another individual or individuals, a failure to so behave, an actual loss or injury to the other (note that this was not a necessary element of battery or assault), and a causal connection between the breach of duty and the loss or injury. These component elements can then be broken down into sub-component elements and alternatives, in the same way that the components of battery and assault were broken down. There is, for example, a series of alternative types of duty that exist between certain individuals in differing, well defined situations.

In the law of contracts, the same kind of componentization is possible. The creation of a valid contract comprises: a proper offer, a proper acceptance, a common understanding of the terms of the contract ("meeting of the minds"), and consideration (remuneration for whatever is contractually promised). Once again, these components are defined in terms of smaller elements and alternatives.

This same process of breaking legal actions and legal relations into elements and counter-elements (defenses) is used throughout the body of the law. The ultimate embodiment of these finely dissected

sub-sub-sub-components are the examples and counter-examples provided by individual case decisions.

Even if our basic modeling technique is adequate for representing areas of legal doctrine, however, there are other problems that arise in a significantly larger system--problems that we have not had to face in designing the prototype.

For example, consider a more comprehensive system containing representations for the entire area of tort law. We might estimate the size of this system by noticing that Prosser's discussion of battery and assault, upon which we based the representations for the prototype, constitutes about one per cent of his treatise on torts. In a system containing one hundred times as much legal doctrine as is contained in the prototype, it might not be practical to try to instantiate every tort exhaustively, which is what we do here with battery and assault. Using a similar measurement, if our Corpus Juris Mechanicum contained representations for all of the doctrines in the general law encyclopedia Corpus Juris Secundum, we sould need a system ten thousand times larger than the prototype.

Larger systems like these would have to include mechanisms for narrowing the instantiation effort to small areas of doctrine for which successful intentiation is relatively likely. A lawyer performs a similar task when he or she consults only particular sections of Prosser or Corpus Juris Secundum, having determined that these sections are likely to be relevant to a set of facts being analyzed. In Chapter 6, we suggested the used of indicator-elements that could be used as clues to

facilitate instantiation among alternatives of particular situations. A similar technique might be employed in a large system before the instantiation process per se even begins. For example, the situation facts-at-hand could first be searched for the existence of certain key facts that point to areas of likely relevant doctrines doctrines. Facts involving injuries or contacts or their apprehension would instigate instantiation attempts in certain areas of tort or crime. Facts involving contracts, or agreements, or purchases and sales, would focus the system's attention on doctrines of contract. Within the scope of these rather broad areas of doctrine, a further narrowing of the issues might be desirable, and it could be achieved by looking for key facts that differentiate among the segments of each area.

Another possible mechanism is to allow the user to restrict the system's analysis to those areas of doctrine that he or she believes might be relevant. This corresponds closely to the lawyer's selecting particular sections of books in manual analysis. Thus, the user might present the hypothetical facts and then ask the system "What torts?" or "Does this constitute a battery?" Just as is the case in manual analysis, the more narrow the scope of doctrine is set by the user, the greater is the risk of missing unanticipated instantiations.

An important limitation of the prototype system is its reliance on the kind hierarchy as its only representation of the world in general. The system does not really know anything about things called "weapon" or "tool," except that they are both kinds of a thing called "movable-object." Similarly, it does not really understand anything

about the event called "hit," except that it is a kind of event called "strike," which is in turn a kind of "contact-event."

Because our prototype system is concerned with instantiating assault and battery, we included some helpful, but somewhat artificial, categories (like movable-object and contact-event) in the kind hierarchy. For purposes of instantiating legal doctrines in other areas, it might be more helpful to focus on other features of some of the items in the hierarchy. For example, for purposes of contract law, we might be less interested in the fact that a tool is movable than than the fact that it has commercial value. In Chapter 3, we mentioned that it is not necessary to restrict a thing like tool to single kind category. Thus, we could include tool as a kind of commercially-valuable-object as well as a kind of movable-object.

The use of multiple classifications increases the amount that the system knows about things and relations in the kind hierarchy, and makes it more useful for the instantiation of different kinds of facts and situations. We must be aware, however, that multiple classifications can significantly increase the machine time that is consumed in trying to match a thing or relation in facts-at-hand to that contained in a fact that is being instantiated.

Consider the number of comparisons that the machine must make for each such match. We can assume that there would not be more than on the order of 100 things and relations in facts-at-hand. At the start of analysis these might be grouped according to relatively broad categories (persons, physical objects, semantic relations, et cetera) so as to limit

each matching search to the category to which the item to be matched belongs. There would not be more than on the order of 10 things or relations in each group. Each of these is a candidate for a match, and so is the item directly above that item, and so on, for about four levels upward in the kind hierarchy. If no multiple classifications exist, no more than about 40 comparisons need be made in searching for a match. However, wherever an item in the kind hierarchy belongs to more than one category, the search branches out, and the number of comparisons needed increases exponentially according to the number of levels searched. For example, if each item in the kind hierarchy belongs to an average of three categories, the number of comparisons needed for a single match becomes 10×3^4, or approximately 1000. Although this number is large, it is not necessarily prohibitive. A system like OWL, for example, can make several thousand of these comparisons in a second.

As we indicated in Chapter 3, things and relations also can be represented in terms of their features. A tool, for example, might be represented by features like mobility and commercial-value, among others. Using representations of this kind, matching could be attempted by comparing and matching the features of a thing or relation in facts-at-hand with those of the thing or relation contained in the fact to be instantiated. The most effective representation for purposes of matching probably involves the use of a kind hierarchy, a limited amount of multiple classification, together with the use of features not accounted for by the hierarchy. Further research will be necessary to resolve this issue.

The prototype's concept of analogy is significantly simplified. To make a proper analogy with regard to particular facts, it is necessary to generalize some things or relation to a certain degree (e.g., by moving up the kind hierarchy), and then to re-specialize that thing or relation, arguably to the same degree (e.g., by moving down the kind hierarchy). The analogy is valid, however, only if the generalized fact or situation conveys the same meaning as the original. For example, consider the situation: John Doe injured his foot. "Foot" can be generalized to "anatomical object" without changing the basic meaning of the situation. However, a generalization to "physical object" would be improper.

As we indicated in Chapters 2 and 6, determining the validity of legal generalization and analogy involves aspects of reasoning that lie beyond the scope of this study. In the prototype system, analogy is achieved by moving up and down the kind hierarchy, but the system pays no attention to the propriety of its generalizations. Also, because our hierarchy does not have multiple calssifications, generalization can proceed only in a single direction, which may or may not lead to valid analogies. The problem of weeding out invalid analogies is left for the user. The overproduction of analogies is limited, somewhat artificially, by restricting analogies to the lower levels of the kind hierarchy, and by permitting only one level of generalization.

There are other limitations in the prototype system. For example, we have omitted temporal relations and values, which are not as important for representing battery and assault as they are for other areas of law

(like the formation of a contract by mail). These relations and values, however, probably can be handled in a manner similar to that used for spacial relations and values.

In sum, this study does not describe a finished working system, but the starting point for the development of a new means for computer-aided legal analysis. Hopefully, the techniques incorporated in the prototype, together with an awareness of their shortcomings, provide a framework for further research.

References

[1] Benjamin N. Cardozo, The Nature of the Judicial Process (New Haven: Yale U. Press, 1921), pp. 30,31.

[2] Cardozo, pp. 31,49.

[3] Kennedy v. Parrott, 243 N.C. 355, 363, 90 S.E.2d 754, 759 (1956).

[4] Galbraith v. Busch, 267 N.Y. 230, 234, 196 N.E. 36, 38 (1935).

[5] Terry Winograd, Understanding Natural Language (New York: Academic Press, 1972).

[6] Eugene Charniak, Toward a Model of Children's Story Comprehension, Ph.D. Thesis (Cambridge: M.I.T. Department of Electrical Engineering, August 1972); also available as Technical Report AI TR-266 (Cambridge: M.I.T. Artificial Intelligence Laboratory, 1972).

[7] See, e.g., Harold J. Berman, The Nature and Functions of Law (Brooklyn: Foundation Press, 1958), pp. 34-40.

[8] See John Henry Wigmore, Evidence, Chadbourn revision (Boston: Little, Brown and Co., 1974), vol. 5, chapters 45 and 49.

[9] See Morris R. Cohen, "The Place of Logic in the Law" 29 Harv. L. Rev. 622 (1916).

[10] Wesley Newcomb Hohfeld, Fundamental Legal Conceptions as Applied in Judicial Reasoning (New Haven: Yale U. Press, 1919), ed. Walter Wheeler Cook.

[11] Lee Loevinger, "Jurimetrics: The Next Step Forward," 33 Minn. L. Rev. 455 (1949).

[12] Loevinger, p. 483.

[13] Georg Henrik von Wright, Norm and Action (New York: Humanities Press, 1963). For a recent application to legal doctrine, see Herbert Keuth, "On Some Logical Characteristics of Legal Norms," 15 Jurimetrics Journal 160 (Spring 1975).

[14] von Wright, p. 74.

[15] See, e.g., David T. Link, "Law Office Management," in Computers and the Law, Robert P. Bigelow, ed., second ed. (Chicago: American Bar Assoc., 1969), p. 63.

[16] See, e.g., Louis F. Comus, Jr., Use of Computers and Other Automated Processes by the Courts (Washington D.C.: World Peace Through Law Center, 1969).

[17] Roy N. Freed, "Machine Data Processing Systems for the Trial Lawyer," 6 Practical Lawyer 73 (April 1960).

[18] Harry M. Halstead, "Use of Computers in Preparing Tax Returns," in Computers and the Law, supra note 15, p. 77.

[19] Robert N. Cook, "Real Property," in Computers and the Law, supra note 15, p. 111. See also "Symposium on Title Recordation," 22 Am. U. L. Rev. 239 (Winter 1973).

[20] Richard C. Larson, Urban Police Patrol Analysis (Cambridge: M.I.T. Press, 1972).

[21] Layman E. Allen, "Symbolic Logic: A Razor Edge Tool for Drafting and Interpreting Legal Documents," 66 Yale L. J. 833 (1957); Allen and Caldwell, "Modern Logic and Judicial Decision Making: A Sketch of One View," 28 Law & Contemp. Prob. 213 (1963).

[22] A recently developed computer system, call JUDITH, guides an attorney through legal analysis based on this kind of propositional structure. The system asks the user to indicate the presence or absence of each component in Boolean expressions representing segments of legal doctrine. See Walter G. Popp and Bernhard Schlink, "JUDITH, a Computer Program To Advise Lawyers in Reasoning a Case," 15 Jurimetrics Journal 303 (Summer 1975).

[23] John F. Horty, Jr., "The 'Key Word in Combination' Approach," 1962 Modern Uses of Logic in Law (now Jurimetrics Journal) 54 (1962); and "Use of the Computer in Statutory Research and the Legislative Process," in Computers and the Law, supra note 15, p. 53.

[24] Jerome S. Rubin, "LEXIS: An Automated Research System," in Automated Law Research (Chicago: American Bar Assoc., 1973), p. 35. See also Andrew D. Egendorf, Computers and Legal Research--Ready for the Second Generation, Third-Year Paper (Cambridge: Harvard Law School, 1971).

[25] Reed C. Lawlor, "Fact Content of Cases and Precedent (Excerpts)," 12 Jurimetrics Journal 245 (June 1972); and "Analysis and Prediction of Judicial Behavior," in Computers and the Law, supra note 15, p. 174.

[26] John M. Dawson, "Probabilities and Prejudice in Establishing Statistical Inferences," 13 Jurimetrics Journal 191 (Summer 1973); Finkelstein and Fairley, "A Bayesian Approach to Identification Evidence," 83 Harv. L. Rev. 489 (1970). But see People v. Collins, 68 Cal.2d 319, 438 P.2d 33 (1968) for a classic example of the misuse of probability theory in court, and see Laurence H. Tribe, "Trial by Mathematics: Precision and Ritual in the Legal Process," 84 Harv. L. Rev. 1329 (1971) for a discussion of possibly inherent incompatibilities between statistical methods and traditional legal concepts of fact finding.

[27] Including Columbia, Harvard, Michigan, Stanford, Texas, and Virginia. See "Education in Jurimetrics," 13 Jurimetrics Journal 69 (Winter 1972).

[28] Patrick H. Winston, Learning Structural Descriptions from Examples, Ph.D. Thesis (Cambridge: M.I.T. Department of Electrical Engineering, January 1970), esp. pp. 7-10 and 158-162.

[29] For example, research related to the development of the OWL system. See Hawkinson, infra note 32.

[30] See, e.g., C. T. Filmore, "The Case for Case," in Universals in Linguistic Theory, ed., Bach and Harms (New York: Holt, Rinehart and Winston, 1968), p. 1.

[31] Jeffrey A. Meldman and Anatol W. Holt, "Petri Nets and Legal Systems," 12 Jurimetrics Journal 65 (December 1971). Legal applications of Petri nets currently are being examined under an NSF grant at the Boston College Law School.

[32] Lowell Hawkinson, "The Representation of Concepts in OWL," presented at the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, U.S.S.R., Sept. 1975; also available as Internal Memo 17, Automatic Programming Group (Cambridge: M.I.T. Project MAC, 1975).

[33] William L. Prosser, Handbook of the Law of Torts (St. Paul: West, 1941), p. 43.

[34] Prosser, p. 48.

[35] Prosser, p. 53.

[36] Prosser, p. 40.

[37] Prosser, p. 118.

Biographical Note

The author received his S.B. in Electrical Engineering from M.I.T. in 1965, and his J.D. from the Harvard Law School in 1968. In 1970, he was appointed as an Instructor in the M.I.T. Department of Electrical Engineering, receiving his S.M. from that department in the same year. Between 1970 and 1974, he taught subjects in network theory, probability theory, and computer science. In 1973, he began a graduate seminar titled "Information Systems and Law," in which law students as well as M.I.T. students were enrolled.

In 1974, he was appointed as an Assistant Professor of Management Science at the M.I.T. Sloan School of Management, where he is teaching subjects in management information systems, and is continuing his legal seminar.

His publications include "Petri Nets and Legal Systems," J. A. Meldman and A. W. Holt, 12 Jurimetrics Journal 65 (December 1971), and "Centralized Information Systems and the Legal Right to Privacy," 52 Marquette Law Review 335 (Fall 1969).

He is a member of the Bars of Massachusetts and Wisconsin, and is a member of the Massachusetts Security and Privacy Council.

CS-TR Scanning Project
Document Control Form

Date : 11 / 6 / 95

Report # LCS-TR-157

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 216 (220-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☒ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number): FOLLOWS TITLE PAGE

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:

IMAGE MAP: (1-216) UNNUM'ED TITLE, BLANK, ABST, ACK, TABLE OF CONTENTS (2), TABLE OF FIG.S, DEDICATION, 8-215 (217-220) SCANCONTROL, TRGT'S (3)

Scanning Agent Signoff:
Date Received: 11 / 6 / 95 Date Scanned: 12 / 4 / 95 Date Returned: 12 / 7 / 95

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 12/4/1995

M.I.T. Libraries
Document Services

darptrgt.wpw Rev. 9/94