EVALUATION OF COMPUTER SECURITY SYSTEMS USING A FUZZY RATING LANGUAGE

by

D. Clements

Memorandum No. UCB/ERL M77/41

1 August 1977

Evaluation of Computer Security Systems using a Fuzzy Rating Language

by

Don Clements

Memorandum No. UCB/ERL M77/41

1 August 1977

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Evaluation of Computer Security Systems using a Fuzzy Rating Language

Don Clements

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

ABSTRACT

A methodology for the evaluation of computer security systems has been investigated. The central idea is the application of natural language as the vehicle for the expression of imprecise and sometimes subjective evaluations by a "security rater" in the absence of objective measures of security performance. It is suggested that linguistic rather than numeric measurement tools are more appropriate in this environment.

An organizational tool for the evaluation of security systems is proposed in the form of the *basic security system* which is a set theoretic model in the form of a 5-tuple that facilitates the enumeration of security threats, protected objects and security techniques or features. The model is augmented with the mechanism of the *linguistic variable* to combine the enumeration and evaluation processes. The linguistic variable and the concept of imprecise evaluation are concepts grounded in the theory of fuzzy sets. Applicable features of fuzzy set theory are discussed and various techniques for the design of rating languages are presented. The security system rater employs phrases in the rating language to express his evaluation of the security performance of a system component.

Methods are examined for the determination of an overall system security rating given the individual component evaluations. The classical "weakest link" approach is compared with the use of scores based upon an imprecise analog of the numerical weighted mean. The several scoring methods have been tested through the use of a security rating calculator, a software product which is briefly described. The calculator is exercised upon a hypothetical sample computer facility. Although the facility analysis is kept at a very high level in the interest of brevity it is suggested that the methodology easily extends to more detailed security analysis activities.

While the applicability of the linguistic variable and its superiority to numerical methods in complex and imprecise environments is not amenable to a formal proof, the utility of the methodology is strongly indicated by its intuitive appeal and the general agreement of evaluation results with the opinions of security experts.

CR Categories: 2.4, 4.6, 8.1

Keywords: security evaluation, fuzzy-set applications, linguistic variable

---

Research sponsored by National Science Foundation Grant MCS76-09214.

1. Introduction

In the last several years, methods for controlling security in computer systems have become widely known. While in 1972 there were only one or two books and three bibliographies on computer security and privacy, in 1977 there are at least 14 books and 6 bibliographies on the topic. An example of a recent general text is [HOFFMAN 1977]. Case histories of computer abuse [PARKER 1976] indicate that this kind of crime is increasing. Knowledge of the techniques and problems involved in computer security is also widespread, as one might guess from the proliferation of literature. Expanding public concern with the problem is evidenced by a great deal of federal, state and local legislation [US 1974], [NASIS 1974], [BERKELEY 1974], [CBEMA 1975], [HR 1984], [PL 93-579]. Governmental agencies such as the National Bureau of Standards, the Defense Department, and the National Science Foundation are all sponsoring research efforts in the area, as have some private manufacturers [SAFE 1974], [SALTZER 1974], [GOLDSTEIN 1975].

One consequence of this work is that a significant part of the computing community is now aware of techniques for maintaining security in computer systems. Unfortunately, the question of how to measure the costs and effectiveness of the various security methods is still largely unexplored. Only recently has there been any reliable work done on costs of privacy transformations or authentication methods, and researchers have only scratched the surface in investigating metrics for security systems [HOFFMAN 1974]. While some more formal work has begun [BELL 1973], [ANDREWS 1975], [POPEK 1974], [WALTER 1974], [HARRISON 1976], [HARTSON 1976], [DENNING 1976], there has so far been very little useful application of this theoretical work to practical security decisions. One exception is a privacy cost model which has been recently introduced [GOLDSTEIN 1976]. This work assesses the impact of various privacy regulations upon data processing installations which handle personal information. Goldstein addresses both the problem of conversion costs to meet privacy requirements and the increment to ongoing operating costs in the installation. He does not, however, deal with the effectiveness of security techniques directly.

We will address the question of security effectiveness. We propose a methodology for the evaluation of data processing installations with respect to the performance of their security systems. We believe the major obstacle to effective measurement of security performance is the lack of suitable metrics. One difficulty is that many security techniques are difficult to evaluate precisely. Another reason for the lack of physical metrics is the large part the human component of the data processing environment has to play in the making or breaking of security systems. Human performance in the area of security is often difficult to quantify. The problem is further aggravated by the lack of reliable historical data in this area.

In the absence of reliable and objective security performance measures, we propose the use of an evaluation mechanism based upon the responses of a *security rater* who employs objective data in arriving at an evaluation whenever possible while drawing upon his or her experience and expertise in formulating subjective evaluations when necessary. Accordingly, we advocate the use of *a security rating language* rather than numeric measures as the basis for system evaluation. A language is more suited to the human rater because it facilitates the expression of imprecision.

The details of rating language design will be presented in Chapter 3. An APL-based software system for calculating security ratings has been developed and is discussed in Chapter 5. All of this is based upon a set-theoretic security model which serves as a vehicle for evaluating data processing installations with respect to security. The model serves as a basis for the analysis of the entire data processing facility by decomposition into security *elements*. Each element is a "point of interest" from the standpoint of security. It identifies the interaction of some object of value to the system owner with some security threat which attacks that object. The element also identifies a security *feature* which has been provided by the data processing installation to deter potential security compromise. Presumably, both an object and a threat must exist for an element to be of interest with respect to system security. As we shall see shortly, each element will, by definition, possess some security feature even if it is only an ineffective null feature. Our goal is to infer a measure of the effectiveness of the security system as a whole based upon the security properties of the individual elements.

We feel that imprecise measurements are required due to the complexity of security systems and the presence of the human component throughout these systems. We further believe that an application of *fuzzy set theory* [ZADEH 1965] to the problem of imprecision will result in a useful though inexact measure or rating. This theory forms the basis for the design of a *linguistic rating scale* in which words are employed instead of (or in addition to) numerical measures of security effectiveness. We have applied this theory to form a software rating system to be employed by security auditors in the evaluation of system security in data processing installations.

1.1 Scope of the Work

Before presenting our security system model, we wish to establish the boundaries of our study. We do so by way of a set of basic assumptions about the nature of the security evaluation problem. A detailed discussion of these assumptions, including plausibility arguments, is presented in Appendix A. The assumptions are:

1. Current methods for evaluating computer security systems are inadequate, mainly for the reasons outlined above.

2. In the absence of applicable physical measurement techniques and adequate historical data, an evaluation methodology based upon a combination of objective data and subjective evaluations can enhance the current check list procedures for system analysis.

3. A model which facilitates system decomposition is necessary since human evaluators are better able to rate system components individually than to evaluate the security system as a whole. A composite rating based upon an exhaustive detailed decomposition will be less likely to omit significant components.

4. The dependence of so many security techniques upon the human component and the presence of the human adversary in many of the security threats in real-world computer systems introduces both complexity in the system analysis and imprecision in the rating estimates.

5. Classical probability theory is not always applicable in the analysis of security risks because threats are not always random in nature. Additionally, the semantics of the rating language are not necessarily best modeled through a statistical or probabilistic approach since there is not a large body of evidence that human raters evaluate in a probabilistic manner. Zadeh's theory of possibility [ZADEH 1977] may be a better choice since it seems more intuitive. We shall adopt it here with the caveat that there is no more empirical evidence that humans evaluate via a possibilistic mechanism than that they use probabilities. Evaluation language semantics may be modeled using probabilities. We do not choose to do so. We believe that they are better modeled (for this application) using the theory of possibility.

6. A suitable foundation for the construction of a subjective and imprecise rating methodology is the Theory of Fuzzy Sets [ZADEH 1965]. This assumption is closely related to number 5.

7. The model to be presented is quite general and thus may lack accuracy in representing some details of complex real-world security systems. We are sacrificing detail for the flexibility needed to deal with the diverse types of security threats and techniques present in a typical data processing facility.

Before we continue with this development, it is necessary to briefly touch upon some of the problems in the area of security system metrics which are considered to be beyond the scope of our present efforts. With reference to Assumption 2 we wish to emphasize that the rating language of Chapter 3 is a suggested sample or prototype. We will not attempt the design of an "optimal" set of rating terms or semantics. We believe the semantics we have developed to be at least intuitively reasonable. The design of some of the language semantics draws upon existing work (see references in Chapter 3).

Nor do we attempt to present a methodology for the determination of the "best" semantics for a given individual security expert. The calibration of the system to individuals is a very difficult problem. We have given some references which indicate how such a methodology might be approached (see Appendix A). We have not studied methods for comparing raters. Suitable statistical normalization techniques may perhaps be developed or existing methods applied to the fuzzy scores; these issues are beyond the scope of this work. We also ignore the problem of selecting a "competent" security expert to formulate the system ratings.

The system decomposition alluded to in Assumption 3 is not detailed here for real-world data-processing installations. Some work in this area has already been done (see [CLEMENTS 1974] and [MICHELMAN 1977] for example). A hierarchical decomposition tool will greatly enhance the practical usefulness of the methodology we are presenting.

We will make no attempt to establish that people think or formulate ratings in a fuzzy manner. The references in Appendix A under the discussion of Assumption 4 deal with this problem. The results to date are inconclusive but not discouraging.

We are not concerned with the accumulation of statistics on security threats or compromises either accidental or intentional. We do assume such information (however incomplete) is available to the security expert to aid in the formulation of his or her ratings. Some very adequate compilations of security compromises and threats are available to the reader in check list or case history form. Some sources are: [BROWNE 1973], [FARR 1972], [KRAUSS 1972], [PARKER 1976], [WOOLDRIDGE 1973].

Ball and Hora have proposed a risk analysis methodology based upon the use of Bayesian techniques [BALL 1977]. Their work is concerned with the calculation of threat probabilities and compromise costs. Their concept of cost relates directly to our term *object loss value* (see Section 1.3). Results of calculations under the Ball and Hora model could provide input to the security rater for the estimation process. As the authors point out, when the prior probabilities are unknown and/or the cost parameters are uncertain, subjective guesses must be made. Since the model is dynamic, the correct values will be produced in time. We feel that lack of prior information is the rule rather than the exception in the field of security. Furthermore, we have assumed that threats are not always probabilistic (Assumption 5 above). Nevertheless, the use

of an approach such as Ball's might aid the rater in making realistic evaluations.

With reference to Assumption 6, we should point out that we are adopting the existing theory of fuzzy sets essentially as is. We have no major or radical changes to propose in order to fit the theory to our application. We will present some minor extensions when necessary.

Finally, we wish to emphasize that complex theoretical modeling is not the goal of this work. The model to be presented is simple. It is intended as a convenient framework for the development of the rating methodology, not as an entirely faithful simulation of a data processing facility.

1.2 Need for Security Metrics

One of the most difficult decisions for the data processing manager has been security investment. How much of the data processing equipment budget should be allocated to the purchase and maintenance of security features? The decision process is largely subjective because there do not currently exist techniques for objectively and precisely measuring the effectiveness of all of the elements of computer system security. It has sometimes been argued (often by designers of military systems) that security must be a *binary* condition - a system is either secure or completely insecure. However, it is highly unlikely that many commercial computer users could afford total security even if it were possible to achieve that state. Indeed, total security may not be desirable if efficiency of data processing greatly suffers as a result.

Nevertheless, security measurement is becoming necessary in a rapidly increasing number of commercial installations. Sound business practice requires the protection of corporate assets, including the physical computing equipment, the information therein, and (of course) the people involved in system operation. Additionally, increasing national concern over privacy and security in computer systems [U.S. 1974], [H.R. 1984] indicate that future data processing managers will devote even more attention to security issues in their selection and operation of data processing equipment.

However, most every manager must work within the limits of a reasonable security budget. It becomes important to know how much extra security is obtained in choosing one security mechanism over another. Ideally, this security increment should be quantifiable so that cost-effectiveness analyses may be performed. Currently, we are a long way from such precise quantification of the effectiveness of security techniques. This is especially true in the areas of data processing security which are more the province of people than the machine. These are the administrative, legal, and physical security considerations in the data processing facility. In these areas, it is most definitely the integrity and/or competence of *people* which makes security work. We believe the impact of the human component will continue to

frustrate the attainment of precision in security metrics. There are just too many human interfaces in security systems and "people cannot be 'proven secure' in a non-Orwellian world" [HOFFMAN 1974].

We may, however, attempt to formalize a methodology for use in evaluation of a security system which organizes and places bounds upon the imprecision and subjectivity currently employed by computer system evaluators. We propose to base this methodology upon the model described in the following section.

1.3 The Basic System Model

As a first step in the design of security metrics we describe an abstraction of a data processing installation's security system. The description takes the form of a set-theoretic model: *the basic security system.* To date, the application of mathematical modeling to studies of security issues has focused mainly upon software structures functioning within or in conjunction with the operating system in multi-user installations. Such structures function to control user access to information and computing resources [WEISSMAN 1969], [LAMPSON 1971], [GRAHAM 1972], [HSIAO 1974].

Three more recent models are worthy of note. Hartson characterizes a security system as a 5-space [HARTSON 1976]. His access control mechanism is concerned with monitoring and granting access requests which result in changes to the system state within the 5-space. His implementation is for use in data-base management applications. Harrison has presented a model of a system with information sharing [HARRISON 1976] and has shown that the owner of information may lose control of access to that information after granting access to a limited number of users. He has proven that the question of guaranteeing that an unreliable user will not pass on an access right to someone unknown to the original owner is undecidable in general, though it may be decidable for a specific system. Dorothy Denning [DENNING 1976] has provided a framework based upon a lattice model of security classes and information flow. The lattice properties permit concise formulations of the security requirements of an existing system into a mathematical framework which facilitates proofs of certain security characteristics. Her model is concerned with determining if "illegal" information flow paths can exist.

Our model addresses a broader range of security problems. We are interested in all of the ways in which data and/or computing resources may be misappropriated. We adopt a view of the computing environment which has been suggested by Turn [TURN 1974]. His conception of the security environment is illustrated in Figure 1.1. While Turn is interested in the design of security systems subject to user needs and external constraints, the measurement of security effectiveness can be viewed as the evaluation of the *threat domain-security system-protected*

domain interface. We therefore adopt a narrower view of the system environment by eliminating the *user domain* and *external constraints* from consideration.

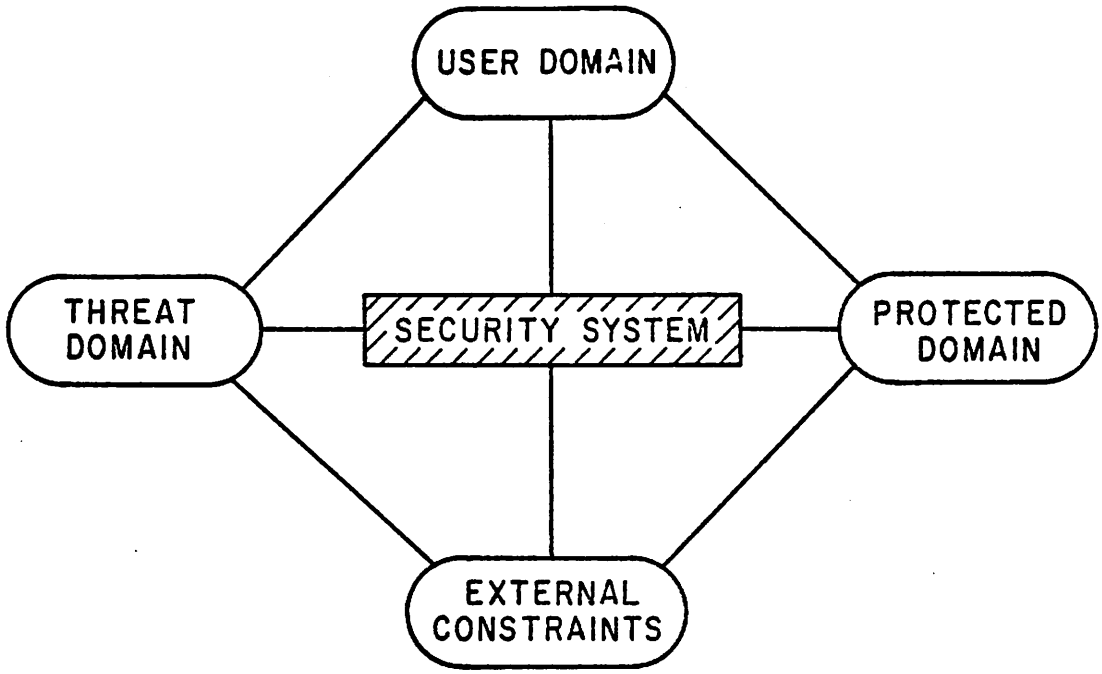

Figure 1.1 Security Environment (from [TURN 1974])

We wish to focus upon those resources within computing systems which are vulnerable to some security threat. In any non-trivial data processing installation there will exist a large variety of such resources, including (but by no means limited to) confidential data, proprietary programs, hardware devices (tapes, terminals, disks, etc.), the operating system, computer time, and even elements of the security system itself (e.g., the password file). We group these elements as the set of security *objects* denoted by the letter *O*. We will use upper case italics to name sets and (possibly subscripted) lower case italics to denote set elements. These security objects display a common characteristic: each possesses a loss *value* to its owner. This value may or may not be precisely quantifiable [GLASEMAN 1977].

Associated with each security object is a number of activities which a potential intruder may employ to gain unauthorized access to that object. We enumerate the potential intrusion activities against each of the security objects to form the set of security *threats* (*T*). Some typical threats are: wire tapping, exploiting operating system trapdoors, reading core residues, theft, misappropriation of computer time, natural disasters, etc. The common characteristic of the threat set is a *likelihood* of occurrence associated with each threat. In a real world environment, these likelihoods are often quantifiable with only a limited degree of precision. In fact, a subjective estimate may often be the best that may be had [GLASEMAN 1977]. Likelihood may

be thought of as a *subjective* probability estimate. We will not use the term *probability* directly since a threat may not always be a random event (see Assumption 5, Section 1.1).

The object-threat relation is modeled as a bipartite directed graph (Figure 1.2) in which edge (t_i,o_j) exists if and only if t_i is a viable means of gaining access to object o_j. It should be noted that the relation of threats to objects is *not* 1 to 1; a threat may compromise an arbitrary number of objects and an object may be vulnerable to more than one threat. The goal of the security game is to "cover" each edge of the graph of Figure 1.2 by erecting a barrier to access along that path.

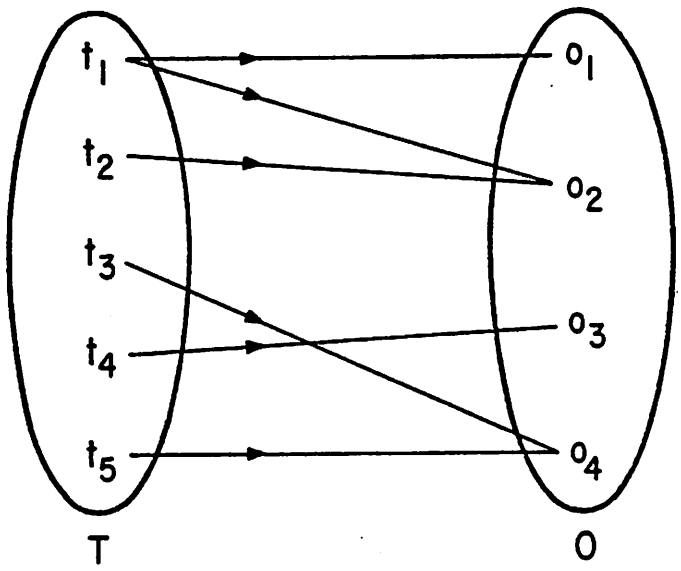

Figure 1.2 The threat-object relation

A third set forms the barriers and completes the model. Its members are the security *features* (F) which are employed as protective devices in the data processing system. Security features include hardware, software, physical, administrative, and legal techniques for countering threats of compromise. Ideally, each feature f_k in F eliminates some edge (t_i,o_j) in the threat-object relation. In reality, a security technique performs a *fire wall* function by presenting some degree of *resistance* to a penetration attempt. This resistance is the common characteristic of the members of F. Some examples of security features are: cryptography, passwords, locks, tape sign-out logs, overwriting core residue and security badges.

The set of security features transforms the bipartite digraph of Figure 1.2 into the tripartite graph of Figure 1.3 if every threat - object edge is "covered" by some feature. Therefore, a "covered" system or sub-system may be defined as one in which all edges in the graph of Figure 1.3 are of the form (t_i,f_k) or (f_k,o_j). An edge of the form (t_i,o_j) identifies an *unprotected* object. It should be noted that a single security technique may counter more than one threat

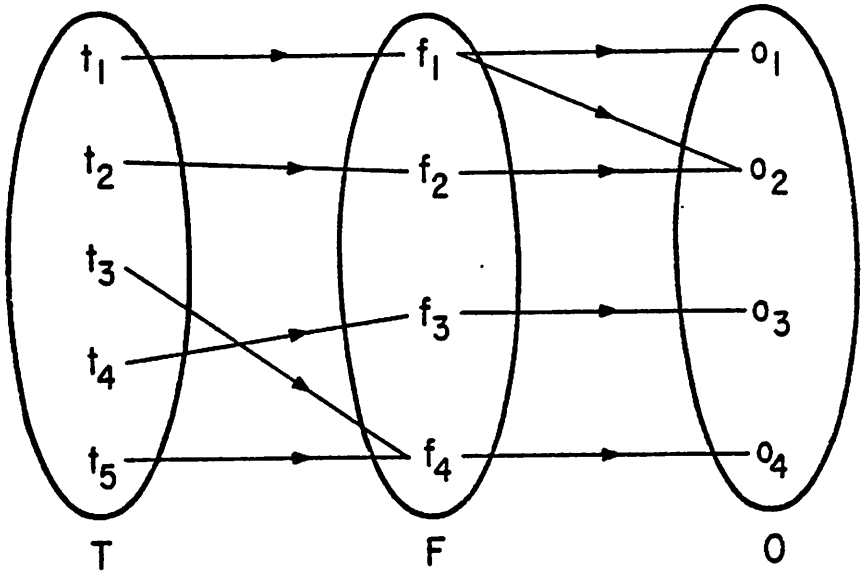

Figure 1.3 The basic security system

(e.g. f_4 in Figure 1.3) and/or protect more than one object (e.g. f_1 in Figure 1.3). We also wish to emphasize that the absence of an edge (t_i,o_j) does not guarantee complete security because of the "fire wall" nature of most security measures. The presence of such an edge does indicate a potential for compromise unless, of course, the likelihood of occurrence of t_i is known to be nil.

We define a *basic security system* as a 5-tuple:

$$S = <O,T,F,V,E> \tag{1.1}$$

where:

O is a set of security *Objects*

T is a set of security *Threats*

F is a set of security *Features*

V is a set of *Vulnerabilities*, a relation in $T\times O$

E is a set of *Elements*, a relation in $V\times F$ (equivalently, a relation in $T\times F\times O$)

In other words, V is the set of ordered pairs (t_i,o_j) (the penetration paths) and E is the set of

ordered triples (t_i, f_k, o_j) which are the points of interest in the data processing system from the security evaluation standpoint.

- Recall that a "covered" system possesses no unprotected objects. In such a system:

$$(t_i, o_j) \in V \rightarrow (\exists k)(t_i, f_k, o_j) \in E. \tag{1.2}$$

If this does not hold the system is not covered and o_j is unprotected for some j.

Of course, there is no way to *completely* enumerate all items of security interest in a real-world computing facility since new threats may arise every day. The best we can do is to identify an *uncovered* system when an object threat pair is found which has no associated security feature. Therefore, coverage is properly considered as a relative property; a system is covered or uncovered relative to some threat set which has been enumerated. Showing that there are no uncovered paths does not imply the system is covered. This is analogous to trying to prove the absence of program bugs through testing. Finding an uncovered system or sub-system is of practical usefulness and is a by-product of the normal enumeration process which is involved in the check-list approach to security analysis. The data processing manager will be interested in any uncovered sub-system since this indicates a security system design oversight. Coverage is also relative to an object set since an uncovered threat is of little interest if there is no associated set of objects which may be compromised by the occurrence of the threat in question.

Although our model appears superficially similar to [HARTSON 1976], the underlying elements and relations greatly differ. Hartson is concerned with access demands and *dynamic* changes in the system state as a result of granting or denying requests. Our model is static and addresses the broader question of overall security performance. Access control by the data base management system (Hartson's object of investigation) might be one security *element* in our model depending upon the level of detail chosen in analyzing a given computer installation. We are concerned with evaluating the *effectiveness* of many types of security features - including the DBMS-type Hartson has developed - and their contributions to the overall security of the computing system. Hartson's goal is the design of a family of protection languages which enable system security administrators to specify ownership, sharing, authorization, access history and other housekeeping parameters in a functioning data base system. Our purpose is to aid the system manager in a post-design security audit or to aid the designer in an iterative design-evaluate process.

1.4 The Problem of Imprecision

If every security feature in our model system possessed an infinite resistance to penetration, our task would be finished. Assuring coverage would be sufficient to assure absolute security. However, real-world security measures present only a finite amount of resistance: passwords have a finite length and are chosen from a finite alphabet, a steel door may be cut given sufficient time, etc. We must therefore address the problem of "measuring", in some sense, these resistances if we are to arrive at an overall evaluation of the degree of security present in the hypothetical data processing facility.

Most security techniques do not lend themselves to reliable and precise quantification. For example, the security of a password scheme may be lower if the password characters are not chosen randomly since clever guessing may quickly discover a password. How much lower is the resulting resistance to penetration? A similar situation exists with respect to the likelihood of a security threat. What is the numerical expectation that a user will lose his or her security badge or have it stolen? Even object values may be calculable only within a range in many cases. Dollar-cost replacement may not be the sole consideration if confidential data is stolen [GLASEMAN 1977]. Loss of client trust or damage to a company's market position may be much more difficult to quantify. In such instances we must rely upon human judgement to provide an *approximate* measure of resistance, likelihood, and value.

The problem is further aggravated when we attempt to combine these individual resistance values to obtain an overall system security rating. The individual security features may interact or depend upon each other. For example, the effectiveness of a password scheme may depend upon the protection mechanism which safeguards the password file. Often these interactions are not well understood. One rating scheme [HOFFMAN 1974] recognizes the need for subjectivity but requires the rater to assign a numerical value to his or her estimates of component resistances. A linear weight and score method is used to produce relative rankings. We feel that such a purely numerically based rating system demands a degree of precision on the part of the system rater which is both difficult to attain and difficult to interpret with confidence. The precision implied in a numerical rating is inconsistent with the complexity of the real world data processing installation. The statement: "XYZ computer installation is .65 secure" may be consistent within the framework of some rating system but is difficult for the outside observer to interpret meaningfully. Further, such a statement is likely to generate more than a little skepticism. Conceptualizing a .65 secure system is much more difficult than visualizing, say, a .65 full cup of coffee. The latter concept is tough enough as it is.

Nevertheless, it is possible to make meaningful measurements of the security of a system relative to another installation or relative to some security standard. We suggest that the appropriate structure for the expression of such measures is the *linguistic variable*

[ZADEH 1975b] - a variable which assumes values which are words rather than numbers. A computing system may be represented by a composite linguistic variable: a structure whose components are themselves linguistic variables termed *attributes.* These attributes might include processing power, cost, amount of storage, security, etc. An exact description of the structure is not necessary here - we are interested only in the security attribute. This component is a simple linguistic variable which takes on values such as **high, low** and **medium.** Appropriate modifiers provide finer resolution by allowing values such as **very high, somewhat low,** and so on. A complete description of the process which produces a linguistic rating scale is given in Chapter 3.

The evaluation of the security of a given data processing system corresponds to the assignment of a value to the attribute "security". This value is the *meaning* of the linguistic rating phrase (see Section 3.1) chosen and is represented by a fuzzy set (see Chapter 2) whose members are values on some (numeric) rating base scale. The rating scale might be an interval on the real line or an integer valued interval. These base scale values are analogous to the numerical ranking values proposed by Hoffman [HOFFMAN 1974]. We sacrifice the precision of a purely numeric approach to gain a higher level of confidence that the final "fuzzy" rating is realistic and easy to interpret.

The overall security rating will be based upon the rater's evaluation of the adequacy of each security element in the system. This overall "score" may be based upon resistances alone, or upon resistances weighted by some combination of object value and threat likelihood. As mentioned earlier, these entities may be quite difficult to quantify precisely. Additionally, every evaluation will be somewhat subjective. For these reasons, we will augment each security element of the basic security system with a composite linguistic variable. There are three components of this variable: *likelihood, resistance* and *value.* Each of these is a simple linguistic variable which may be assigned linguistic values taken from the rating scale. Each is associated with a component of the element triple (t_i, f_k, o_j) in the basic model. Resistance corresponds to the feature f_k, likelihood corresponds to the threat t_i and value is associated with the object o_j.

There are intuitively reasonable interpretations for the dimensions of these three measures. Resistance could be measured in units of time, likelihood could take on units of numeric probability, and value could be measured in dollars. In the interest of simplicity and economy of rating scale design we will assume that all three measures are normalized to dimensionless quantities. Thus, we are interested in relative measures. A single dimensionless base rating scale will be used. For most of the examples in this work we will assume the rating scale to be the real interval (1,9) or the ordered sequence of integers from 1 to 9. This gives the scale an integer valued mid-point at 5. The implementation of our rating calculator (see Chapter 5)

uses the integer valued rating scale.

As mentioned in Appendix A, it would also be possible to model "likelihood" as a *linguistic probability* [ZADEH 1975b]. Linguistic probabilities assume values such as *likely, unlikely, highly likely,* etc. In order to keep this exposition conceptually simple we will employ only one rating vocabulary using variants of **high, low**, etc. as discussed previously. These may be thought of as relative linguistic values which are applied to evaluate resistance, likelihood and value.

The type of scoring function used to arrive at an overall system rating will depend upon the security policy in effect at the installation under analysis. A "dogmatic" security policy [TURN 1974] requires the complete protection of all resources. Such a policy would require a *weakest link* scoring function. A "rational" security policy [TURN 1974] calls for a *weighted score* where the weights are a function of the object values and threat likelihoods. Scoring functions are discussed in detail in Chapter 4.

1.5 The Rated Security System Model

The basic model elements are augmented with composite linguistic variables as outlined above. In order to avoid confusion, we will use (possibly subscripted) upper-case **boldface** letters to denote linguistic variables. The letter **E** denotes the compound structure made up of the composite linguistic variable defined above and the associated triple from the element relation (E) of the basic security system. Consider Figure 1.4 which depicts a single *rated security element.* Note that $\mathbf{E}_r$ identifies the triple which represents e_r in the element relation of the basic model. Additionally, a *likelihood* attribute $\mathbf{L}_r$ (where $\mathbf{L}_r$ is a linguistic variable) is associated with threat t_i. The object component o_j has *value* attribute $\mathbf{V}_r$ and a *resistance* attribute $\mathbf{R}_r$ belongs to the security feature f_k. Note that the subscripts for likelihood (**L**), value (**V**) and resistance (**R**) match the subscript (r) of the *element* (**E**) rather than the individual threat, object, and feature component subscripts. This is to emphasize that these components are evaluated in the context of the *specific security element* which they form. In other words, these measurements are dependent upon the particular object-feature-threat interaction.

As a qualitative example of this interdependence, consider a file containing a proprietary program belonging to a software engineering firm. The loss value of the program may be **pretty high** in the context of a threat of theft by a competitor who may exploit its commercial value. On the other hand, the file may have a **fairly low** value in the context of a threat of accidental erasure (especially if a machine readable back-up copy exists).

Values such as **pretty high** or **fairly low** are assigned to the linguistic variables at each security element in the data processing system. These linguistic values are modeled as fuzzy

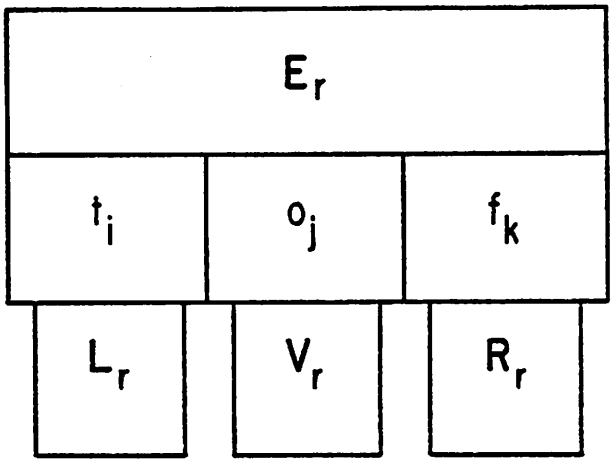

Figure 1.4 The rated security element

sets. Each of the rating base scale values possesses a *grade* of membership in the fuzzy set which is the meaning of one of these linguistic values. This grade of membership is represented by a *compatibility function* which is denoted by the greek letter μ. Fuzzy sets are more completely discussed in Chapter 2. We anticipate that development here only to emphasize the following: fuzzy sets are *not* probability distributions although there are some superficial similarities. A related discussion appears in Appendix A.

Figure 1.5a depicts a possible compatibility function for the phrase **high** over the real base scale (1,9).

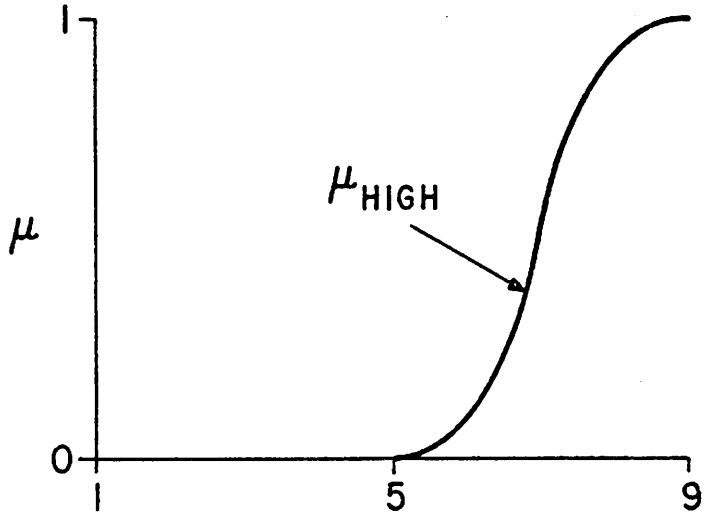

Figure 1.5a A possible compatibility function for **high**

Again, μ_{high} is not a probability although μ will (by definition) range over the (0,1) interval [ZADEH 1965], [ZADEH 1975b]. The proper interpretation of μ_{high} is that it represents the degree to which each of the base values agrees with our conception of the notion of **high**. Similarly, Figure 1.5b illustrates the concept **very high**.

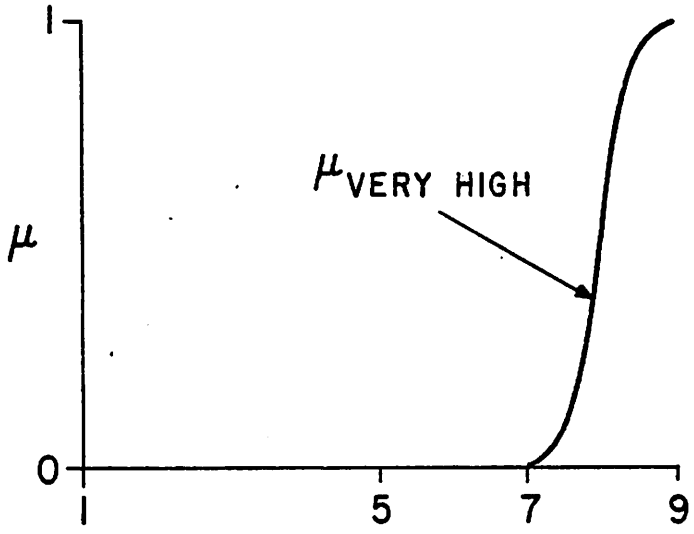

Figure 1.5b A possible compatibility function for **very high**

Neither of these curves should be taken too literally. They are intended only as qualitative illustrations.

As mentioned in the description of the basic security system, there may be unprotected objects in a given installation. In terms of our earlier notation:

$$(\exists i)(\exists j)\left[[(t_i,o_j)\in V]\ \&\ (\forall k)[(t_i,f_k,o_j)\notin E]\right] \tag{1.3}$$

We may easily augment the model to include uncovered objects in the element relation. We introduce the *null security feature* written f_0 and add it to the set of security features in the system. This null feature possesses the property that any security element of the form:

$$(t_i,f_0,o_j)$$

will have a **null** resistance rating (**null** is the lowest rating available) regardless of the specific object and threat in the element. We define an augmented security feature set:

$$F_a=F\bigcup\{f_0\}$$

and replace F with F_a in the model definition (1.1). Now define a set of uncovered elements: E_u which contains those uncovered pairs in E, each of which is augmented with the null

security feature:

$$E_u=\{(t_i,f_0,o_j) \mid (t_i,o_j)\in V \ \&\ (\forall k)(t_i,f_k,o_j)\in E\} \tag{1.4}$$

Then replace the set E in the model with the augmented set $E_a=E \bigcup E_u$.

1.6 The Evaluation Process

Evaluation begins with the analysis of the data processing installation by decomposing it into a set of security elements. At each element, the security auditor or rater assigns a linguistic value to the resistance (R) of the security feature at that point. The rater (or perhaps the installation manager) then assigns a linguistic value to V and also L: the object loss value and the threat likelihood at this particular element. These three values determine the contribution of the element to the overall installation security rating. The resistance value determines the amount by which overall security is enhanced or degraded. Depending upon the scoring function in use (see Chapter 4), this increment/decrement may be weighted by some combination of object value and threat likelihood.

As a crude example consider a system composed of only two elements. The first is:

$$e_1=(t_1,f_1,o_1)$$

where t_1 is a software bug in user A's program which overwrites his or her private data file (o_1) and f_1 is optional read only protection available for the file system. The second element is:

$$e_2=(t_2,f_2,o_1)$$

where t_2 is the threat from user B browsing through the files of user A, o_1 is as before and f_2 is password protection available for private files. Assume a hypothetical rater assigns the following values at the two elements:

$$
\begin{aligned}
\mathbf{E_1}: \ \mathbf{L_1} &= \textbf{fairly high} \\
\mathbf{V_1} &= \textbf{high} \\
\mathbf{F_1} &= \textbf{very high} \\
\\
\mathbf{E_2}: \ \mathbf{L_2} &= \textbf{medium} \\
\mathbf{V_2} &= \textbf{very high} \\
\mathbf{F_2} &= \textbf{medium}
\end{aligned}
\qquad (1.5)
$$

Using "weakest link" scoring based upon the MIN score (see Section 4.1), the system rating is **medium** (due to $\mathbf{F_2}$). If a "fuzzy weighted mean" scoring function (see Section 4.4) is employed and threat likelihood is the weight, $\mathbf{L_1}$ causes $\mathbf{F_1}$ to dominate and the overall rating might be **high**. On the other hand, if loss value is considered the paramount issue, $\mathbf{V_2}$ causes $\mathbf{F_2}$ to dominate and the total score might be **sortof high** or some other value nearer to **medium** than **high** Weights which are combinations of **L** and **V** would generate still different total scores.

1.7 Overview of the Thesis

The next chapter presents a brief review of fuzzy set theory. Only those elements of the theory which are needed for development of concepts used later in this work are presented. Chapter 3 develops a methodology for the design of a *rating language.* This language is used to generate the rating phrases used in assigning values to the various resistances, likelihoods, and loss values in the installation. A set of default compatibility functions which represent the *meanings* of the terms in the rating language is also given. These may be used in the absence of any information about a particular rater's *linguistic bias,* that is, in the absence of empirically gathered data (classified by individual rater) relating numeric rating values to linguistic rating phrases.

Chapter 4 describes some suggested scoring functions based upon the "weakest link" security philosophy and upon the linear weight and score method of Hoffman [HOFFMAN 1974]. Chapter 5 discusses our implementation of a software package for gathering element ratings and

calculating overall system scores. A fictitious data processing installation is rated in Chapter 6 where the various scoring functions are compared. Chapter 7 is a summary and a discussion of some possible extensions to the present work. Appendix A contains a detailed discussion of the major assumptions of the thesis (see Section 1.1). Finally, the actual APL code which is the software for gathering and calculating ratings is displayed in Appendix B.

2. Basics of Fuzzy Set Theory

In this chapter we present a very brief outline of fuzzy set theory. We intend to discuss only those portions of the theory which will be useful in the understanding of the linguistic rating process. For full details on this growing field, the reader may consult the bibliography. See especially Kaufmann's very complete text [KAUFMANN 1975] for a starting point. Material more closely related to our specific applications may be found in [ZADEH 1975b].

2.1 Definition of a Fuzzy Set

We begin with a *universe of discourse* which is just some collection of items of interest. We will deal with two such universes: an interval of the real line between and including 1 and 9 which we will label: R and the set of integers from 1 through 9 which we label: I. We will often refer to the universe of discourse as the *base scale* and its elements as base values. A fuzzy set is a subset of some universe of discourse which is defined by a *membership function.* This membership function is a mapping from the universe of discourse to the real valued interval(0,1). It represents the *grade* of membership of each point in the universe of discourse in the fuzzy set. A membership function is usually labeled with the Greek letter μ. Thus, if X is a fuzzy subset of R :

$$\mu_X\colon\ (1,9)\rightarrow(0,1) \tag{2.1}$$

Notice that a classical (non-fuzzy) set (labeled Y for example) may be considered to possess a *binary* membership function:

$$\mu_Y\colon\ U\rightarrow\{0,1\} \tag{2.2}$$

where {0,1} is the set of values 0 and 1 rather than an interval. Therefore, a fuzzy set may be viewed as an extension of the notion of a classical set. The membership function is often also referred to as the *compatibility* function. The notion of compatibility is more intuitive for our purposes since it emphasizes the idea that set membership may be a subjective thing. We will use the two terms interchangeably. A non-fuzzy finite set is often denoted explicitly as:

$$A=\{a_1,a_2,\ \cdots,a_n\} \tag{2.3}$$

where the a_i are values in U where $\mu_A = 1$. We will represent a finite fuzzy set explicitly as:

$$A=\{\mu_1[a_1],\mu_2[a_2],\ldots,\mu_n[a_n]\} \tag{2.4}$$

We will omit the braces when no confusion will result. A general form which displays the functional property of μ is:

$$A = \bigcup_{i \in U} \mu_A(i)[i] \tag{2.5}$$

where μ_A(i) indicates the result of applying the compatibility function to the singleton: *in the universe of discourse.* This yields a value in (0, 1) which is the μ-value (membership value) for that singleton. This μ-value is paired with its base value as indicated by the $[i]$ notation. The union of these *fuzzy singletons* yields the fuzzy set: A.

This notation holds an advantage for us over the form which most often appears in the literature (see for example [ZADEH 1975b]). There the standard notation employs the "/" (slash) to separate compatibility and base values (e.g. .2/5) and union is often represented by "+". We wish to reserve these symbols for use in discussing fuzzy arithmetic. As an example, let I be our universe of discourse. We may represent the set labeled *slightly more than 2* (with {} omitted) as:

$$1[3],\ .7[4],\ .5[5],\ .1[6] \tag{2.6}$$

These values are arbitrarily selected for illustration purposes. When the fuzzy set is infinite the membership function will be given in closed form. For example, we might represent **about 9** in R as:

$$\bigcup_{x \in (7,9)} 0.5 \times (x-7)[x] \tag{2.7}$$

where "×" represents conventional multiplication and the union is taken over the (infinite) set of values in (7,9). It is understood that base values outside the interval (7, 9) are assigned membership values of zero (see Figure 2.1). We shall deal mainly with finite sets in this work and will use fuzzy sets on the continuum solely for purposes of illustration. We shall also employ closed forms on occasion in describing operations upon fuzzy sets.

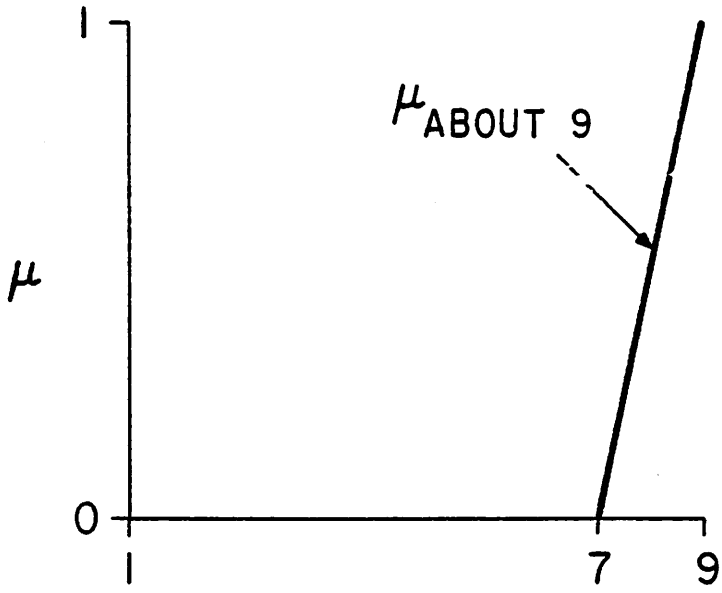

Figure 2.1 The simple membership function of (2.7)

2.2 Properties of Fuzzy Sets

For a given fuzzy set: X, the set of points in U for which μ_X is non-zero valued is called the *support* of X. The *height* of X is defined as the maximum value of μ_X. A fuzzy set is *normal* if its height is 1. A sub-normal set may be made normal by dividing μ_X by the height of the set. We shall find this *normalization* process useful later.

Another important property of certain fuzzy sets is *convexity*. A general definition of convexity is given in [ZADEH 1965]. For the special cases in which the universe of discourse is a segment of the real line or an interval of integers, we can define convexity quite simply. A fuzzy set $X \in U$ is convex if:

$$(\forall i)(\forall j)(\forall k)(i \leqslant j \leqslant k \rightarrow \mu_X(j) \geqslant (\mu_X(i) \wedge \mu_X(k))) \tag{2.8}$$

Here the "$\wedge$" operator signifies the taking of the minimum. Figure 2.2 illustrates convexity and normality.

Convexity is an important consideration when modeling linguistic ratings as fuzzy sets. For example, a phrase such as **about 2 or about 8** (see Figure 2.3a) indicates a value which is simultaneously near both end points of the rating scale but not near the center. While this is a perfectly good fuzzy concept, it is difficult to make intuitive sense out of such a value as a rating or score. In words, the rater is saying: "The score is either near 2 or near 8." On the other hand, Figure 2.3b illustrates a valid (though perhaps overly vague) rating value which might be

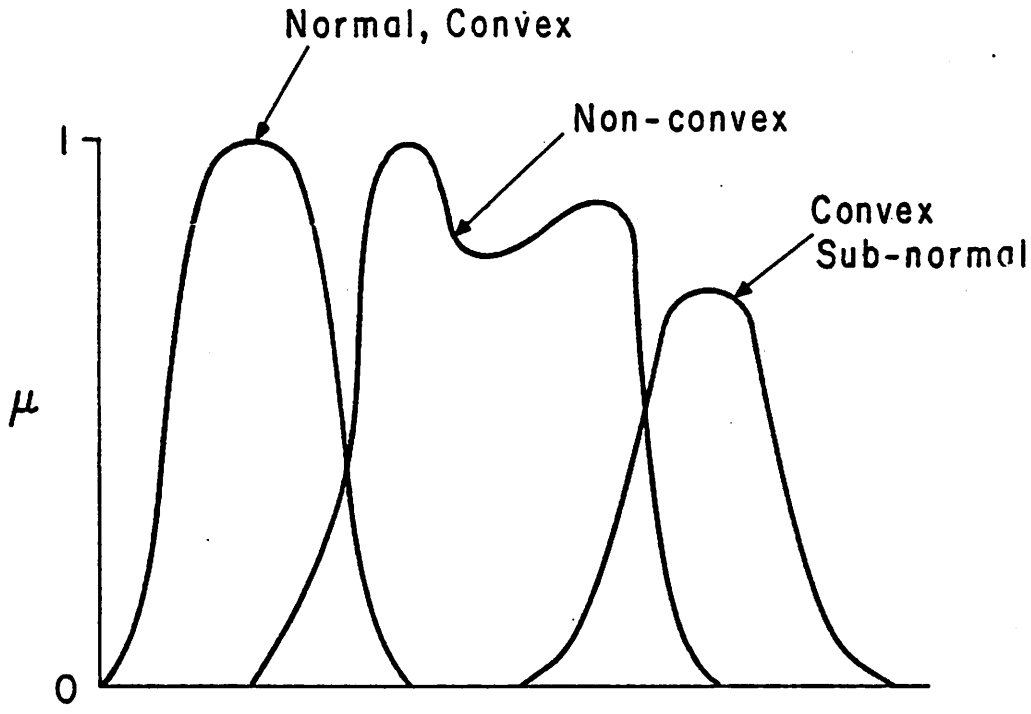

Figure 2.2 Normal convex, non-convex, and sub-normal fuzzy sets

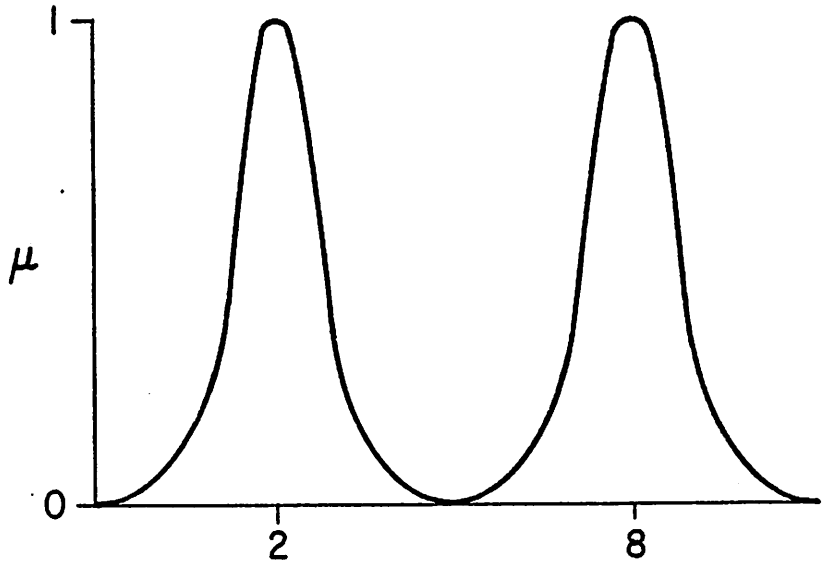

Figure 2.3a Semantics of **about 2 or about 8**

termed **about 2 to about 8**. The rater is saying "I'm uncertain about the exact score but it is not lower than about 2 and not higher than about 8". Notice that the meaning of this phrase is a convex fuzzy set.

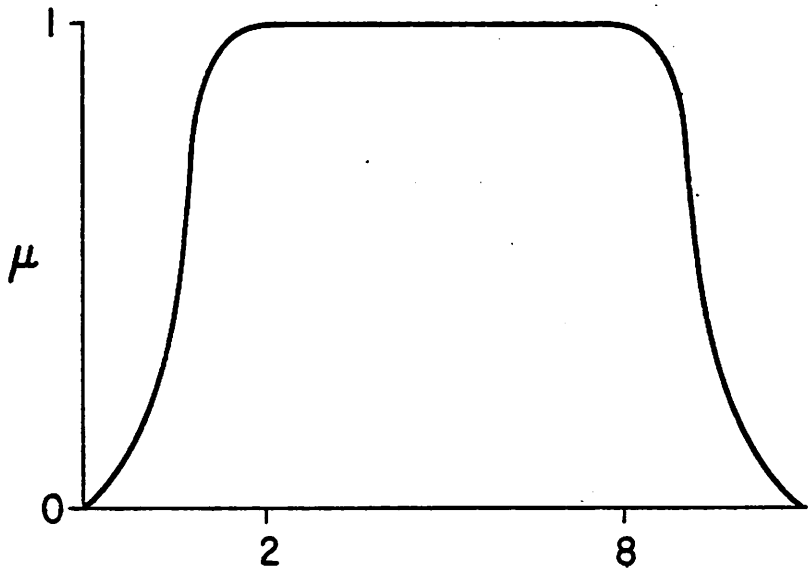

Figure 2.3b Semantics of **about 2 to about 8**

Of course, normal, convex sets represent only a small portion of the possible forms for the compatibility function which the general fuzzy set theory allows. As we have seen, there is a certain intuitive justification to requiring a linguistic rating to possess a meaning which is a convex fuzzy set. There is also justification for maintaining normality. As rating phrases become more and more complex (see Figure 3.7 in Chapter 3), peak compatibility values may get smaller and smaller until linguistic approximation (see Chapter 5) becomes very difficult. Normalization assures the final rating "curve" in normal, convex fuzzy sets in this work. will belong to the family of curves which makes up the rating language. Thus convexity and normality greatly simplify the task of the linguistic approximator without sacrificing the flexibility needed to express a linguistic rating value. Accordingly, we will be mainly interested

2.3 Classical Set Operations

Three basic operations upon fuzzy sets are complementation, union, and intersection. Let X be a fuzzy set over U. The *complement* of A may be denoted as A' and is defined as:

$$A' = \bigcup_{a \in U} (1 - \mu_A(a))[a] \tag{2.9}$$

where "-" represents the usual subtraction operation. Here the notation $\mu_{A(a)}$ indicates the result of applying the compatibility function associated with set A to the base value a. After subtraction from unity, this new μ-value is associated with a (indicated by the $[a]$ notation)

and the process is repeated over all values in the universe of discourse. Complementation implements the linguistic operator **not** and thus represents negation as in **not high** (see Section 3.3).

The *intersection* of two fuzzy sets is defined by:

$$A \cap B = \bigcup_{c \in U} (\mu_A(c) \wedge \mu_B(c))[c] \tag{2.10}$$

where "$\wedge$" indicates the minimum of the two values is to be taken. The *union* of two fuzzy sets is defined by:

$$A \cup B = \bigcup_{c \in U} (\mu_A(c) \vee \mu_B(c))[c] \tag{2.11}$$

where "$\vee$" indicates that the maximum of the two values is to be taken. These three basic operations are illustrated graphically in Figure 2.4.

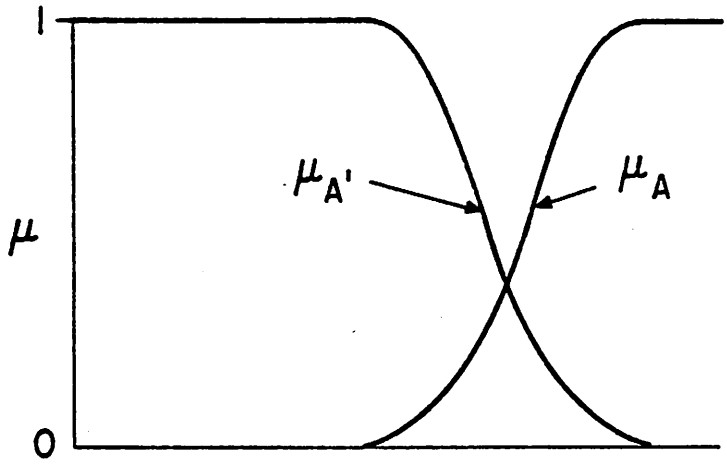

Figure 2.4a Complementation

An in-depth discussion and intuitive justification of these definitions may be found in [BELLMAN 1973]. We will make use of these operations in rating language design (see Chapter 3).

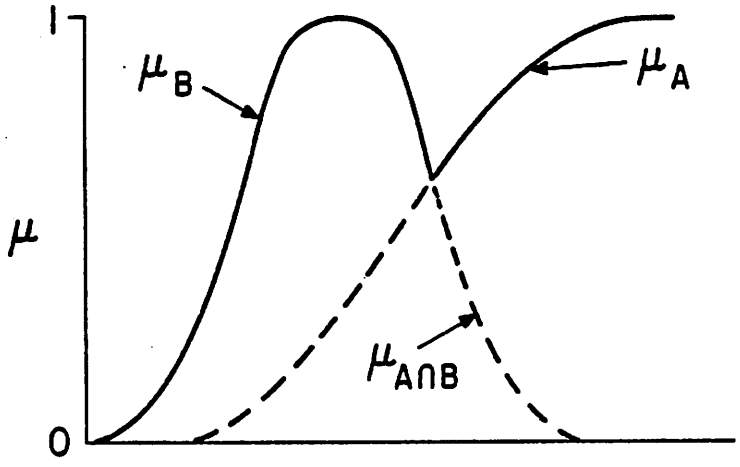

Figure 2.4b Intersection

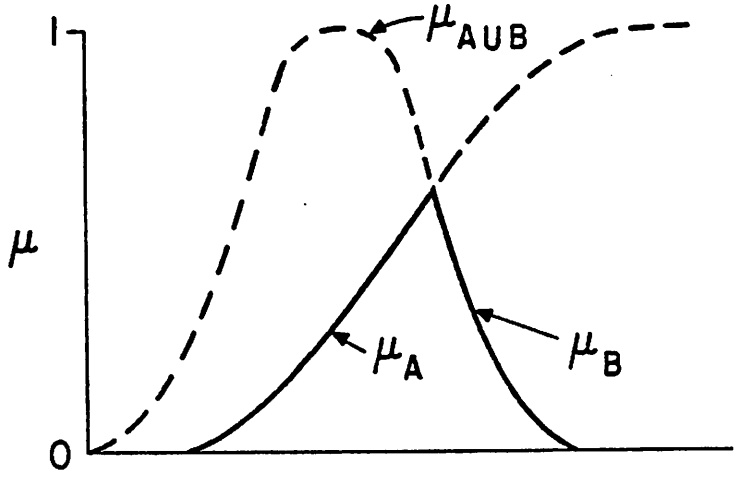

Figure 2.4c Union

2.4 Compatibility Function Modifiers

In Section 3.3 we discuss the notion of a *hedge:* a unary operator that modifies the *meaning* of a linguistic value. That meaning is represented as a fuzzy set. In the construction of hedges, we will find the operations of *concentration* (CON) and *dilation* (DIL) useful. These operations are defined (see [ZADEH 1972]) as:

$$\mathrm{CON}(A)=A^2$$

$$\mathrm{DIL}(A)=A^{0.5} \tag{2.12}$$

where A is a fuzzy set and in general,

$$A^c=\bigcup_{a\in U}(\mu_A(a))^c[a] \tag{2.13}$$

Thus, CON and DIL are operators which, in a sense, decrease and increase the fuzziness of a set since μ-values are modified to a degree which depends upon their distance from unity. The operation of *intensification* [ZADEH 1972] is somewhat akin to concentration in this respect but is more emphatic in separating high and low μ-values. Intensification (INT) is defined as follows:

$$\mu_{\mathrm{INT}(A)}=\begin{cases}2\times(\mu_A(a)^2) & \text{if } \mu_A(a)\leqslant 0.5\\ 1-2\times(1-\mu_A(a))^2 & \text{if } \mu_A(a)>0.5\end{cases} \tag{2.14}$$

A qualitative comparison of concentration dilation and intensification is given in Figure 2.5.

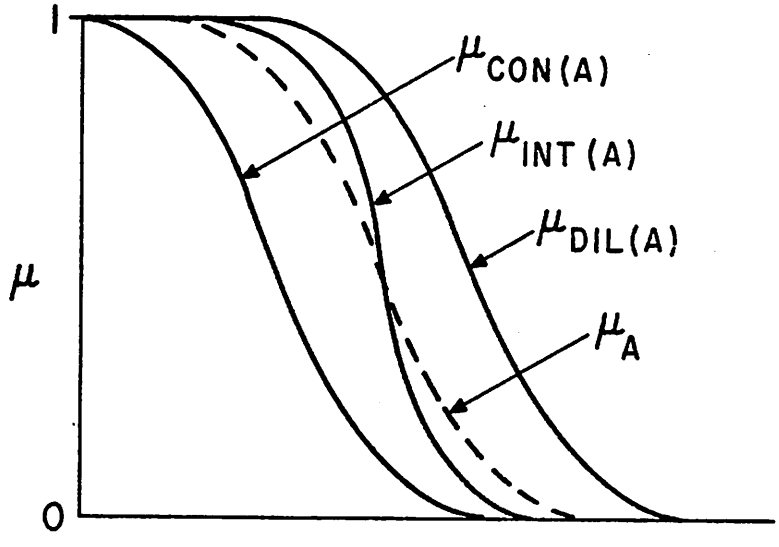

Figure 2.5 Concentration, dilation and intensification

The operation of *fuzzification* has an effect opposite to that of intensification in that the fuzziness of a set is increased. In the case of a non-fuzzy set, fuzzification transforms the set into a fuzzy one. Fuzzification is discussed in great detail in [ZADEH 1972]. We will be concerned with one form: an operation which modifies the *support* of a fuzzy or non-fuzzy set. The

fuzzifier FUZ accepts two parameters: a fuzzy set A and a set of *kernels* K each of which is a fuzzy set generated by applying FUZ to a non-fuzzy singleton in the universe of discourse. FUZ has the effect of performing a scalar multiplication of the membership function of A at each point in U over the kernel at that point.

The operation is easier to illustrate than to explain. Let I (integers 1 to 9) be the universe of discourse. Let:

$$K(1)=\{1[1],\ .5[2]\}$$

$$K(9)=\{.5[8],\ 1[9]\} \tag{2.15}$$

$$K(i)=\{.5[i-1],\ 1[i],\ .5[i+1]\},\ 2\leqslant i\leqslant 8$$

Each $K(i)$ is a distinct fuzzy set which is used in the fuzzification process as follows. Let:

$$A=\{.4[5],\ 1[6],\ .6[7]\} \tag{2.16}$$

We represent scalar multiplication of compatibility function values by "×":

$$\mathrm{FUZ}(A;K)=\bigcup_{i\in I}\mu_A(i)\times K(i)[i] \tag{2.17}$$

$$=\{.4\times K(5),1\times K(6),.6\times K(7)\}$$

$$=\{.4\times(.5[4],\ 1[5],\ .5[6]),1\times(.5[5],\ 1[6],\ .5[7]),.6\times(.5[6],\ 1[7],\ .5[8])\}$$

$$=\{.2[4],\ .4[5],\ .2[6],\ .5[5],\ 1[6],\ .5[7],\ .3[6],\ .6[7],\ .3[8]\}$$

$$=\{.2[4],\ .5[5],\ 1[6],\ .6[7],\ .3[8]\}$$

The behavior of FUZ upon a non-fuzzy singleton is of particular interest. Let $B=\{4\}=\{1[4]\}$ and let K be given by (2.15). Then:

$$\mathrm{FUZ}(B;K)=\{1\times K(4)\} \tag{2.18}$$

$$=\{1\times(.5[3],\ 1[4],\ .5[5])\}$$

$$=\{.5[3],\ 1[4],\ .5[5]\}$$

This enables us to transform a non-fuzzy set (in this case a singleton) into a fuzzy set. Fuzzification is useful in the generation of fuzzy numbers. For instance, the results of the last example could be interpreted as the fuzzy number **about 4** (see Section 3.3).

2.5 Fuzzy Relations

An n-ary *fuzzy relation* [ZADEH 1971], [ZADEH 1975b] is a fuzzy subset of the (non-fuzzy) Cartesian product of n universes of discourse. We will confine our attention to binary fuzzy relations in $I \times I$ where I is the integer base scale defined in Section 2.1 above. Fuzzy relations in real n-space are discussed in the references. A binary relation R has a bivariate membership function over the ordered pairs (i,j) in $I \times I$. The general descriptive form is:

$$R = \bigcup_{(i,j) \in I \times I} \mu_R(i,j)[i,j] \tag{2.19}$$

We shall depict binary relations explicitly as n by n matrices where n is the cardinality of the universe of discourse. In the interest of brevity, we choose a universe of discourse: $J=\{1,2,3\}$. The relation **similar** in $J \times J$ might be defined as:

$$\textbf{similar} = \begin{matrix} 1 & .5 & .02 \\ .5 & 1 & .5 \\ .02 & .5 & 1 \end{matrix} \tag{2.20}$$

For a pair (i,j) in **similar**, i denotes a row and j denotes a column.

In words, the *meaning* of a relation R is just a statement of the compatibility of pairs in $I \times I$ under the notion which R labels. Thus in the case of (2.20), **similar**(2,1)=.5 means that 0.5 is the degree to which 2 is compatible with 1 under this particular notion of similarity. In other words, 2 is "similar" to 1 with degree 0.5 in this case.

In general, if R is a relation from U to V (i.e. $R \in U \times V$) and S is a relation from V to W, then the *composition* of R and S yields a relation from U to W. In this work, we will be using a single base "rating" scale (see Chapter 3) in defining the fuzzy sets and fuzzy relations which make up our rating language. Therefore, for our purposes, U, V, and W are all identical to I. Modifying the above definition apply to our special usage, a composition of binary relations will yield a relation in $I \times I$. Composition is denoted by the symbol "O" and is defined (for discrete fuzzy sets) in terms of the max-min inner product [ZADEH 1971] as:

$$R \circ S = \bigcup_{(i,j) \in I \times I} \bigvee_{j \in I} (\mu_R(i,j) \wedge \mu_S(j,k))[i,k] \tag{2.21}$$

where "$\vee$" represents taking the maximum μ-value and "$\wedge$" represents taking the minimum. Note that "$\vee$" applies over all $j \in I$. In words, the compatibility of i and k in the resultant relation is the maximum of the minimums with j as the linking component. As Zadeh points out, composition is just the max-min matrix product when the universes of discourse are finite sets. The max-min product is the analog of the classical matrix "dot" product with "$\vee$" replacing addition and "$\wedge$" replacing multiplication. Since our implementation models the rating scale as a set

of integers, we need consider only this form of composition. We borrow an example from [ZADEH 1975b]. Here the universe of discourse is simply: {1,2} so that R, S, and ROS are defined over $\{1,2\}\times\{1,2\}$. We represent these ordered pairs as (labeled) rows and columns of the relation matrices:

$$R = \begin{matrix} .3 & .8 \\ .6 & .9 \end{matrix} \tag{2.22}$$

$$S = \begin{matrix} .5 & .9 \\ .4 & 1 \end{matrix}$$

Then the composition of these relations is given by the matrix product (where "$\vee$" replaces summation and "$\wedge$" replaces multiplication in the more familiar matrix product definition):

$$ROS = \begin{matrix} (.3\wedge.5)\vee(.8\wedge.4) & (.3\wedge.9)\vee(.8\wedge 1) \\ (.6\wedge.5)\vee(.9\wedge.4) & (.6\wedge.9)\vee(.9\wedge 1) \end{matrix} \tag{2.23}$$

$$= \begin{matrix} .4 & .8 \\ .5 & .9 \end{matrix}$$

2.6 Compositional Inference

We shall find use for composition in combining fuzzy relations and fuzzy sets. The effect of a relation upon a value which is a fuzzy set is described by the *compositional rule of inference* [ZADEH 1975b] which is a generalization of the familiar modus ponens rule of the propositional calculus:

FROM: p

AND: $p \rightarrow q$ (2.24)

INFER: q

Analogously, consider the application of a function f:

GIVEN: $x=a$

AND: $y=f(x)$ (2.25)

INFER: $y=f(a)$

Now suppose a in (2.25) is a fuzzy set in I. An alternative way to view a fuzzy set is as an *assignment* to a unary (fuzzy) relation in I. Let $X(i)$ be such a relation. Then $X(i)=a$ may be thought of as the relational assignment corresponding to $x=a$ in (2.25). Let f be a fuzzy set in $I\times I$ and let $F(i,j)$ be a binary (fuzzy) relation in $I\times I$. Then the compositional rule of inference states that we may infer a unary relation $Y(j)\in I$ as the result of the two relational assignments and the composition given by:

FROM: $X(i)=a$

AND: $F(i,j)=f$ (2.26)

INFER: $Y(j)=X(i)\circ F(i,j)$

where "$\circ$" is the composition operator of (2.21). The unary relation $Y(i)$ defines a fuzzy set y in I. When we are dealing with finite fuzzy sets, composition is just the max-min product of a unary and a binary relation:

$$\mu_Y(j)=\bigcup_{j\in I}\bigvee_{i\in I}(\mu_X(i)\wedge\mu_F(i,j))/j \tag{2.27}$$

This gives us a semantic rule for determining the effect of a relation upon a fuzzy value. As a quick example, let $F=R$ where R is defined in (2.22) and let $a=\{.4[1], .7[2]\}$. Then $X(i) = .4 \quad .7$ and:

$$Y(j) = .4 \quad .7 \circ \begin{matrix} .3 & .8 \\ .6 & .9 \end{matrix} \tag{2.28}$$

$$= (.4\wedge.3)\vee(.7\wedge.6) \quad (.4\wedge.8)\vee(.7\wedge.9)$$

$$= .6 \quad .7$$

that is, the result is the fuzzy set: $y=\{.6[1], .7[2]\}$. Note that (2.28) was arbitrarily constructed to illustrate the *mechanics* of compositional inference. Our use of this rule is detailed in connection with the relation **lower** in Section 3.5 where its operation is more intuitively appealing.

2.7 The Extension Principle

Finally, we describe the *extension principle* [ZADEH 1975b] which allows a mapping or relation which operates in some universe of discourse U to apply to fuzzy subsets of U. In essence, the extension principle asserts that the result of applying a mapping f to a fuzzy subset A obtains from applying f to each base element of A in turn:

$$\begin{aligned} A &= \{\mu_1[a_1],\ldots,\mu_n[a_n]\} \\ f(A) &= \{f(\mu_1[a_1],\ldots,\mu_n[a_n])\} \\ &= \{\mu_1[f(a_1)],\cdots,\mu_n[f(a_n)]\} \end{aligned} \tag{2.29}$$

For an analogous formulation when U is a continuum see [ZADEH 1975b].

In Chapter 4 we develop scoring functions for finding composite security system ratings. These scoring calculations depend upon the application of standard arithmetic operations to fuzzy sets. The extension principle for binary operators is presented in [ZADEH 1975b]. If $\bullet$ is an arbitrary binary operator in U and A and B are fuzzy sets in U:

$$A=\bigcup_{i\in U}\mu_A(i)[i]$$

$$B=\bigcup_{j\in U}\mu_B(j)[j] \tag{2.30}$$

By the extension principle:

$$\begin{aligned} A\bullet B&=\left(\bigcup_{i\in U}\mu_A(i)[i]\right)\bullet\left(\bigcup_{j\in U}\mu_B(j)[j]\right) \\ &=\bigcup_{(i\bullet j)}(\mu_A(i)\wedge\mu_B(j))[i\bullet j] \end{aligned} \tag{2.31}$$

Here the tacit assumption is that:

$$\mu_{A,B}(i,j)=\mu_A(i)\wedge\mu_B(j) \tag{2.32}$$

That is, the result is a subset of the fuzzy Cartesian product of A and B. The "$\wedge$" operator is characteristic of taking the fuzzy Cartesian product [ZADEH 1975b].

2.8 Summary

Reviewing the major operations upon fuzzy sets which have been discussed in this chapter, recall that for fuzzy sets A, B in some universe U:

$$A' = \bigcup_{a \in U} (1-\mu_A(a))[a]$$

$$A \cup B = \bigcup_{c \in U} (\mu_A(c) \vee \mu_B(c))[c]$$

$$A \cap B = \bigcup_{c \in U} (\mu_A(c) \wedge \mu_B(c))[c]$$

$$\mathrm{CON}(A) = A^2$$

$$\mathrm{DIL}(A) = A^{0.5}$$

$$\mu_{\mathrm{INT}(A)} = \begin{cases} 2\times(\mu_A(a)^2) & \text{if } \mu_A(a) \leqslant 0.5 \\ 1-2\times(1-\mu_A(a))^2 & \text{if } \mu_A(a) > 0.5 \end{cases}$$

For a fuzzy set A and a kernel function K in I:

$$\mathrm{FUZ}(A;K) = \bigcup_{i \in I} \mu_A(i) \times K(i)[i]$$

For fuzzy relations R, S in $I \times I$:

$$R \circ S = \bigcup_{(i,j) \in I \times I} \bigvee_{j \in I} (\mu_R(i,j) \wedge \mu_S(j,k))[i,k]$$

For a fuzzy set a in U, a fuzzy set f in $U \times U$, unary relations X, Y and binary relation F:

FROM: $X(i) = a$

AND: $F(i,j) = f$

INFER: $Y(j) = X(i) \circ F(i,j)$

For a general binary operator "$\bullet$":

$$A \bullet B = (\bigcup_{i \in U} \mu_A(i)[i]) \bullet (\bigcup_{j \in U} \mu_B(j)[j])$$

$$= \bigcup_{(i \bullet j)} (\mu_A(i) \wedge \mu_B(j))[i \bullet j]$$

This introduction to the theory of fuzzy sets is by no means an exhaustive treatment of the subject. The interested reader will find a wealth of literature beginning with the references in the present work. We have purposely limited our discussion to those features of the theory which will prove useful in the development to follow. Specifically, in the next chapter we will employ the above tools in the design of a sample security rating language. This language will be the vehicle for the construction of fuzzy security ratings.

3. Designing a Security Rating Language

We now outline the general method for constructing a rating tool based upon words and phrases rather than numbers. As discussed in Section 1.6, we must evaluate three components at each security *element* in the data processing facility: threat *likelihood,* object *value,* and feature *resistance.* In a completely general system, three separate rating scales would be constructed since the units of the base variable scale would differ for each component. For example, subjective probability, dollars, and expected safe time might be chosen as the underlying units. We will make the simplifying assumption that each of the three base scales is first normalized to range over a dimensionless scale. Then, without loss of generality, we need consider only one rating scale for each of the 3 components.

3.1 The Base Scale

Specifically, the base variable will range over the interval (1,9). In the interest of clarity, some of the illustrations in this and other chapters will feature continuous compatibility functions (i.e mappings onto the real line). Therefore, in the general case the rating base variable is real-valued. Certain theoretical matters discussed in Chapter 4 also require the use of fuzzy sets upon the real line. However, our implementation (see Chapter 5) models fuzzy sets over the integers. Additionally, examples are often easier to construct and understand when the integers from 1 through 9 are used. Our base variable will be real or integer valued as convenient. We emphasize that the real line is the general case; the integers are used to simplify our implementation and examples.

In the next section we discuss the various terms which make up our rating mini-language. Some of the terms are *primary,* that is their definition in the form of a fuzzy set is considered axiomatic. Other terms operate upon the primary terms to yield composite terms or phrases which yield finer shades of meaning. Neither the names of the rating terms nor the forms of their fuzzy sets are to be considered as the only choices. Our implementation allows the rater to optionally supply his or her own term set and provide the fuzzy sets and functions which represent the *meaning* of the rating terms. The language presented here is a standard default provided with the rating calculator described in Chapter 5.

3.2 The Primary Linguistic Values

We choose as primary rating terms the values **low, high,** and **medium.** A possible representation for μ_{high} is given in Figure 3.1. The values of μ_{low} and μ_{medium} are shown also.

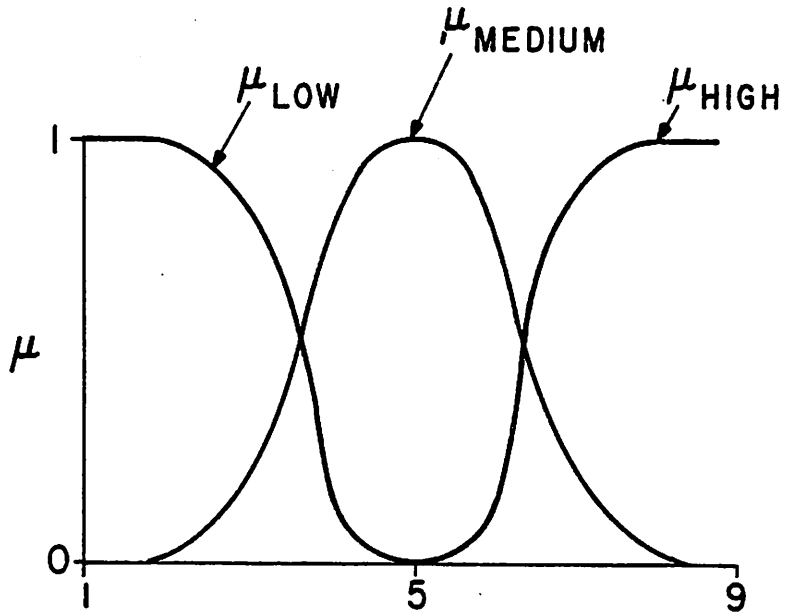

Figure 3.1 The primary linguistic values

Our selection of these forms is based largely upon an intuitive notion of the underlying *meaning* of the primary terms **low, high,** and **medium.** We are employing fuzzy sets to model the semantics of these terms in a way which makes more explicit the approximate nature of the concept. For example, consider the notion **high.** Each value $\mu_{\text{high}}(i)$ represents the degree to which we subjectively believe that the base value i (in the context of the limits of the base scale) is compatible with the notion of a **high** value. In other words, i belongs to the class of **high** ratings (where in this case 9 is the highest possible rating). The degree to which i belongs is given by $\mu_{\text{high}}(i)$.

In the absence of empirical data which would indicate the way in which people internalize concepts such as **high** with reference to a numerical rating scale, we might reason in the following way. We would expect a maximum at the upper end point of the rating scale since (on a scale of 1 to 9) one would surely perceive 9 as **high.** It seems reasonable to expect that values near the mid-point and lower would be excluded from the notion of **high.** The shape of the curve in between is more difficult to justify but it seems reasonable to expect that the μ-values will decrease slowly at first as we move away from the upper end of the base scale and then fall off more rapidly. G. Lakoff presents a more detailed and compelling analysis of this curve form in conjunction with the concept "tall" [LAKOFF 1973].

Similar reasoning may be applied to justify the forms of the μ-functions of **low** and **medium**. Since these concepts are so subjective, our implementation provides these primary fuzzy sets as default values. The user is free to substitute his or her own μ-functions as well as names other than **high**, **low**, and **medium**.

We now present two approaches to the problem of defining primary terms. In the next section we discuss three general methods for defining linguistic modifiers. These various techniques are provided as options and defaults in the implementation (see Chapter 5). The first scheme is due to Zadeh. He has formulated parameterized closed forms which may be used to generate fuzzy sets for primary terms of this type. These canonical forms are called S and π functions [ZADEH 1976]. The S-function is defined (we have changed Zadeh's notation somewhat):

$$S(v;\, z,c,p) = \begin{cases} 0 & , \quad v \leqslant z \\ 2\times(\frac{(v-z)}{(p-z)})^2 & , \quad z < v \leqslant c \\ 1-2\times(\frac{(v-p)}{(p-z)})^2 & , \quad c < v < p \\ 1 & , \quad v \geqslant p \end{cases} \tag{3.1}$$

where:

v is the base value
z is the v at which $s = 0$ (Zero)
p is the v at which $s = 1$ (Peak)
c is the v at which $s = .5$ (Crossover)

Normally the "crossover" would be halfway, that is $c=\frac{(z+p)}{2}$. Then the **high** and **low** curves of Figure 3.1 might be generated by:

$$\mu_{\text{high}}(u) = S(u;\, 5,7,9) \tag{3.2}$$

$$\mu_{\text{low}}(u) = 1-S(u;\, 1,3,5) \tag{3.3}$$

The π function generates a pulse which may be used to form the meaning of **medium**. It is defined in terms of the S function, a peak parameter (p), and a bandwidth (b). The bandwidth is just the distance on the base scale between crossover points.

$$\pi(v;\, b,p) = \begin{cases} S(v;\, p-b,p-\frac{b}{2},p) & , \quad v \leqslant p \\ 1-S(v;\, p,p+\frac{b}{2},p+b) & , \quad v > p \end{cases} \tag{3.4}$$

With reference to Figure 3.,1 **medium** might be generated by:

$$\mu_{\text{medium}}(u) = \pi(u;\, 2,5) \tag{3.5}$$

The S and π functions are convenient because they make it easy to specify the limits and likely "interesting" points of the generated compatibility functions. However, other forms which approximate these curves are possible. For example, Shaket has experimented with the exponential as a canonical form for generation of fuzzy sets [SHAKET 1975]. An exponential which produces a form much like Zadeh's π function (we will call it the E function) is:

$$E(v;\, p,s) = \exp^{-(\frac{(v-p)}{s})^2} \tag{3.6}$$

where:

v is the base value

p is the "peak" value of E

s is the "spread" ($E=\frac{1}{e}$ at $p+s$ and $p-s$)

The typical E curve is shown in Figure 3.2 which is adapted from [SHAKET 1975].

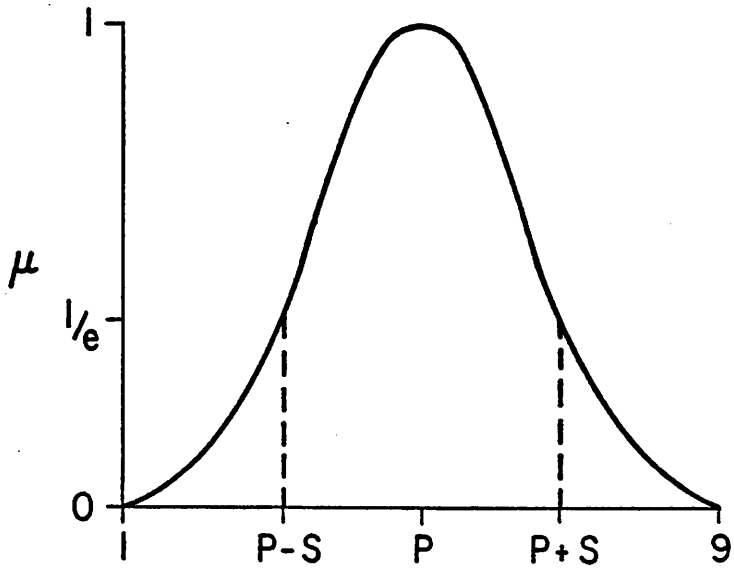

Figure 3.2 An exponential-based compatibility function

Note that "spread" is analogous to "bandwidth" as defined earlier. The difference is that spread gives the $\mu=\frac{1}{e}$ points rather than $\mu=0.5$ points. There are no $\mu=0$ points on the E curve since its behavior is asymptotic. This property may be of some use in limiting cases. At any rate, suitable rounding functions exist in our implementation to optionally force the very low μ-

values to 0 when desired.

If p=1 or p=9, the E curve peaks at one of the boundaries of the base scale and one half of the curve disappears. The E curve then approximates the S function described above. Thus the E function gives us some computational simplicity at the expense of some of the flexibility and intuitive appeal of the S and π curves. All of these canonical forms are available as default functions in our implementation. In the general case the primary terms of the rating system will be specified by the user and the fuzzy sets which represent the "meanings" will probably be given by the user explicitly or determined empirically (see Section 7.6).

3.3 Linguistic Hedges

Given the three primary terms **(low, high, medium)** we next define a set of *modifiers* each of which operates upon a primary term to effect a change in meaning, either through a modification of the form of the curve (that is the fuzzy set) or through a shift of its position on the rating scale or both. Modifiers of this type are classified as *hedges.* The subject of hedges is treated very completely in the references. The interested reader should especially note [ZADEH 1972], [ZADEH 1973] and [LAKOFF 1973]. As we shall see shortly, hedges work best with the end point primaries **high** and **low**. We have included **medium** as a primary because it seems to be a concept which is as intuitively basic as **low** or **high**. The fact that it is difficult to use hedges with **medium** and get semantically meaningful composite terms suggests that it may in reality be more of a complex notion.

One modifier which is very useful is negation which is represented by the adverb **not** as in **not high**. We use the complement operator as presented in Section 2.3 to model **not**:

$$\textbf{not } P = P' \tag{3.7}$$

where P is a primary term such as **high** and "'" represents the taking of the complement of the fuzzy set which is the meaning of P. We shall often use the same symbol (here P) to represent the label of a linguistic value and the fuzzy set which is its meaning when the context removes ambiguity. We shall retain the convention of representing actual linguistic values in **boldface.**

Using the primary functions shown in Figure 3.1, the hedged value **not high** is illustrated in Figure 3.3 along with **high** and **low**. Notice that **not high** has quite a different meaning than **low**. Negation is usually considered to be an operator rather than a hedge but since its semantics are very similar in operation to a hedge we will consider it as such for the purposes of rating language design.

The basic operations CON and DIL presented in Section 2.4 enable us to define hedges which increase and decrease fuzziness in primary terms. The hedge **very** has the effect of

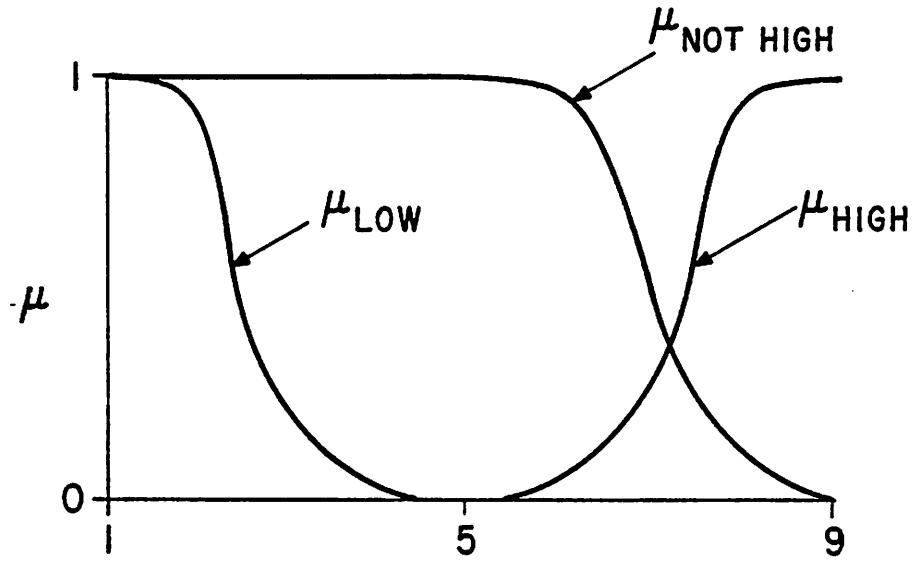

Figure 3.3 Comparison of **low** *and* **not high**

steepening the slope of a term such as **high**. All compatibility values less than 1 are reduced with the net effect that base values which had a low compatibility with the concept of **high** will have a much lower compatibility with the concept of **very high**. In other words, fewer base values will have a strong association with the hedged value. **Moreorless** has an opposite effect. All μ-values less than 1 are increased. Since more base values attain a strong association with **moreorless high** than with **high** the effect is to increase the fuzziness of the concept. In terms of our basic operations (see Chapter 2):

$$\textbf{very } P = \text{CON}(P) \tag{3.8}$$

$$\textbf{moreorless } P = \text{DIL}(P) \tag{3.9}$$

Incidentally, **moreorless** is written without intervening blanks to emphasize that the phrase behaves semantically as a single indivisible operation. The effect of these two hedges is illustrated in Figure 3.4.

As Lakoff points out [LAKOFF 1973], the power of 2 in CON (see Equation 2.12) is chosen somewhat arbitrarily. In fact, we can gain a more satisfactory degree of generality by defining **very** in terms of a *parameterized* concentration function (CONC) which is a slightly liberalized version of the CON function of Zadeh. Thus:

$$\mu_{\text{CONC}}(X,a)=(\mu_X)^a \tag{3.10}$$

Now we may define **very** as before (3.5) and additionally define the hedge **extremely** as an even stronger concentration:

$$\textbf{very } P = \text{CONC}(P, 2) \tag{3.11}$$

$$\textbf{extremely } P = \text{CONC}(P, 3) \tag{3.12}$$

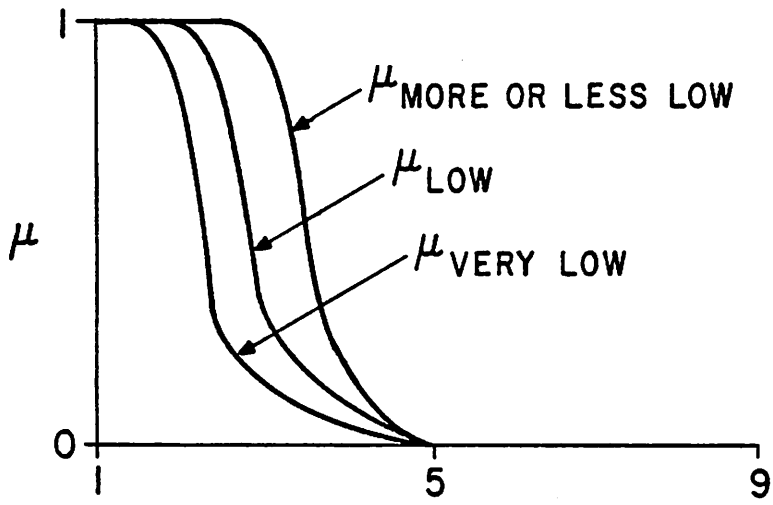

Figure 3.4 Comparison of **very** *and* **moreorless**

It is possible to similarly parameterize the DIL operator but we will not do so here. Operationally, the parameterized CON and DIL functions are identical except that CON accepts powers greater than 1 while DIL is concerned with fractional powers. This redundancy is useful in making clear the *meaning* of these operators, so we will retain it.

We should point out that **moreorless** has also been interpreted as a *fuzzifier* in the literature [ZADEH 1972]. As pointed out in Section 2.4, this operation makes a non-fuzzy set fuzzy or increases the fuzziness of a fuzzy set by modifying the support of the set. Rather than define two versions of **moreorless** we introduce a new hedge **about** which is specifically intended to fuzzify a numerical rating. Thus the phrase **about five** means that the non-fuzzy singleton:

$$\textbf{five} = 1[5] \tag{3.13}$$

is fuzzified by the hedge **about**. Our definition of **about** is in the same spirit as Zadeh's use of "approximately" in connection with the definition of fuzzy numbers (see [ZADEH 1975b].) In our implementation (see Chapter 5) a default value for the kernel of the fuzzification is provided but may easily be changed by the user. For the purposes of illustration in this chapter, we will use the following formulation:

$$\text{KERNEL}(i) = .4[i-2],\ .8[i-1],\ 1[i],\ .8[i+1],\ .4[i+2] \tag{3.14}$$

where values beyond the end points of the rating scale are dropped (that is, $\mu_j=0$, $j<1$ or $j>9$). Then:

$$\begin{aligned} \textbf{about five} &= \text{FUZ}(\textbf{five};\ \text{KERNEL}(5)) \\ &= 1\times(.4[3],\ .8[4],\ 1[5],\ .8[6],\ .4[7]) \\ &= \{.4[3],\ .8[4],\ 1[5],\ .8[6],\ .4[7]\} \end{aligned} \tag{3.15}$$

Fuzzification is illustrated in Figure 3.5.

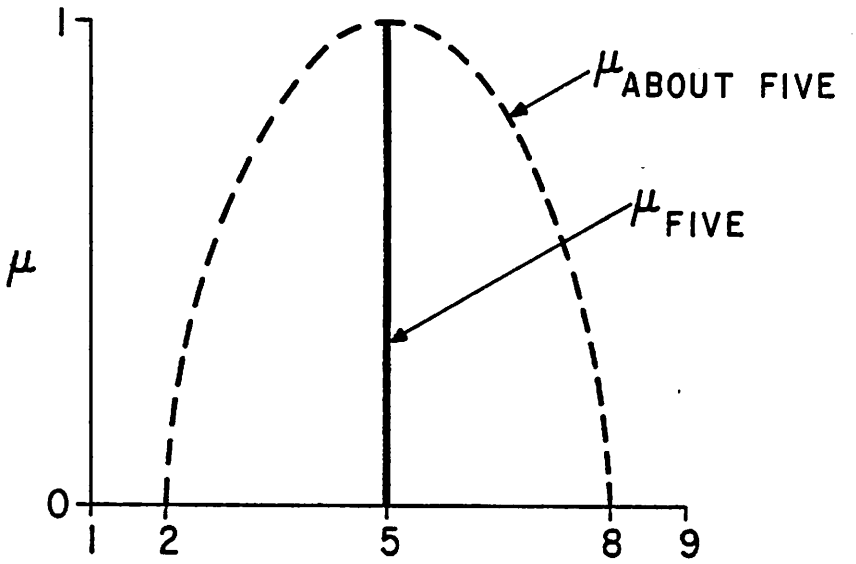

Figure 3.5 The fuzzification operation

For the purpose of contrast intensification we define the hedge **indeed** as simply:

$$\textbf{indeed}\ P = \text{INT}(P) \tag{3.16}$$

This interpretation of **indeed** is due to Wenstop [WENSTOP 1975]. The effect of **indeed** differs from **very** in that it modifies membership values which differ significantly from 0.5 (the "cross-over points"). The hedge **very** concentrates the entire set about its peak value. Figure 3.6 displays both of these hedges for comparison.

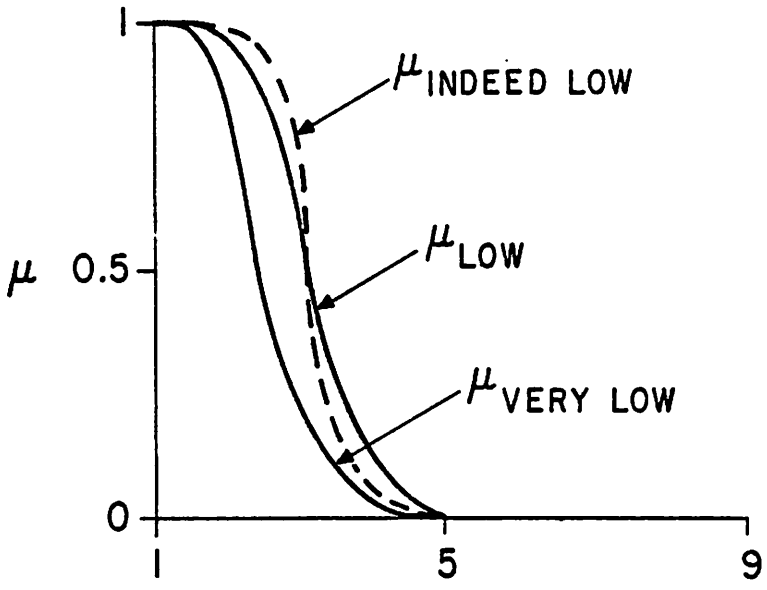

Figure 3.6 Comparison of **very** *and* **indeed**

3.4 Connectives

These basic hedges may be combined to form more complex functions which will cause peak shifts of the primary values. In this way we may generate rating values which "lie between" the primary fuzzy values on the rating scale. First we need to add connectives to our rating language.

Our most useful connective will be **and** which is simply defined as:

$$A \text{ and } B = A \cap B \tag{3.17}$$

where A, B are (possibly hedged) primary values. Zadeh has shown [ZADEH 1965] that the intersection operation preserves convexity. Similarly, we could define **or** as the *union* of two fuzzy sets. This definition can be found in [ZADEH 1972]. However, the connective **or** is not useful in forming complex rating terms because it does not preserve convexity. For example, **high or medium** would peak at 5 and 9 (using the earlier definitions of the primary terms). Since μ-values between the peaks would be less than 1, convexity is lost. This issue of convexity is discussed in greater detail in Section 2.2 where the operation of **or** is graphically illustrated (Figure 2.3).

The simplest solution is to make all of the μ-values between these peaks equal to the peak values so that a (very broad) convex curve results. The corresponding intuitive argument is that when we say an object value (or feature resistance, or whatever) is **high or medium** we really mean the value is less compatible with base values below 5 and above 9 but equally compatible with all points between these bounds. In other words we are giving the end points of an

interval which is interpreted as a range of equally compatible values. In order to avoid confusion with the more usual notion of **or** we shall employ the connective **to** as in **high to medium**. This connective will be defined as the union with the result corrected to maintain convexity.

Through the use of combinations of basic hedges and connectives it is possible to define more complex hedges. These will operate upon the end point primary values to yield terms which have meanings (in the sense of fuzzy sets) which peak at base values between the primary terms. This allows greater flexibility in specifying a fuzzy rating.

We define the hedge **pretty** to perform a very slight peak shift in the end point curve. For example, **pretty high** will have a maximum at 8 on the 9 point rating scale used in our implementation. Using the hedges we have already discussed, the definition of **pretty** is:

$$\textbf{pretty}\ P = \text{NORM}\ \textbf{moreorless}\ ((\ \textbf{extremely}\ P\ )\ \textbf{and moreorless not very}\ P) \qquad (3.18)$$

where evaluation proceeds from right to left except when overridden by parentheses. The process is illustrated in Figure 3.7; the reasoning proceeds as follows. The hedge **very** creates a curve lying to the right of **high**. The hedge **extremely** yields a curve which is even further displaced toward the high end of the base scale (Figure 3.7a).

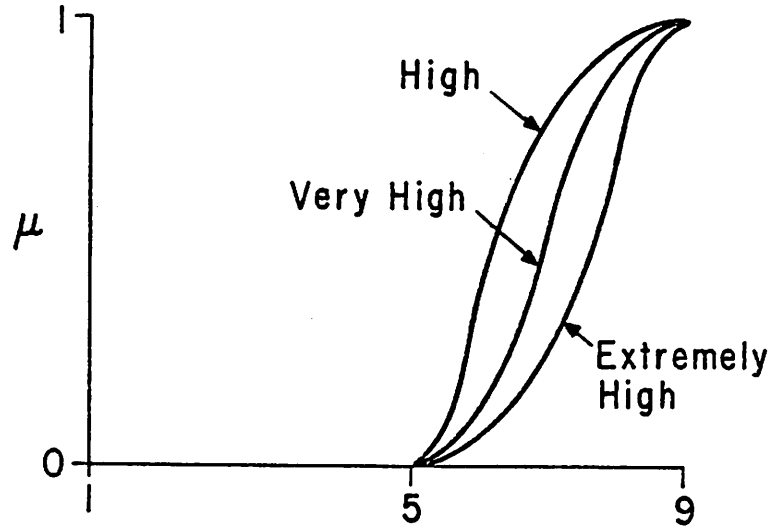

Figure 3.7a Forming the portions of **pretty high**

This latter curve will form the left half of the result (before dilation and normalization). Applying **not** to **very high** results in the dashed curve of Figure 3.7b. The hedge **moreorless** shifts the dashed line slightly right. Applying the connective **and** yields the intersection of the two fuzzy sets represented by the heavy solid lines in Figure 3.7b. This result is labeled **X** in Figure 3.7c. The leftmost **moreorless** acts upon the result and it is in turn normalized. Normalization must be applied since **and** does not preserve normality in general. We assume all

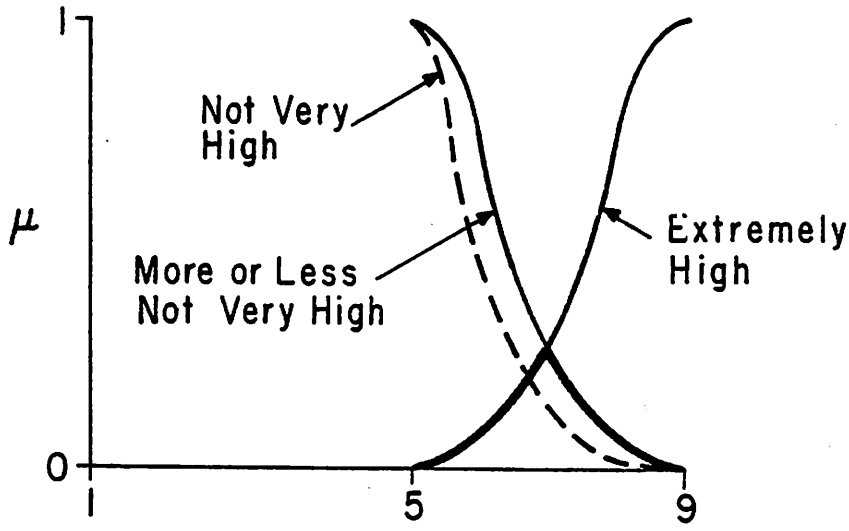

Figure 3.7b Negation forms the left half of the phrase

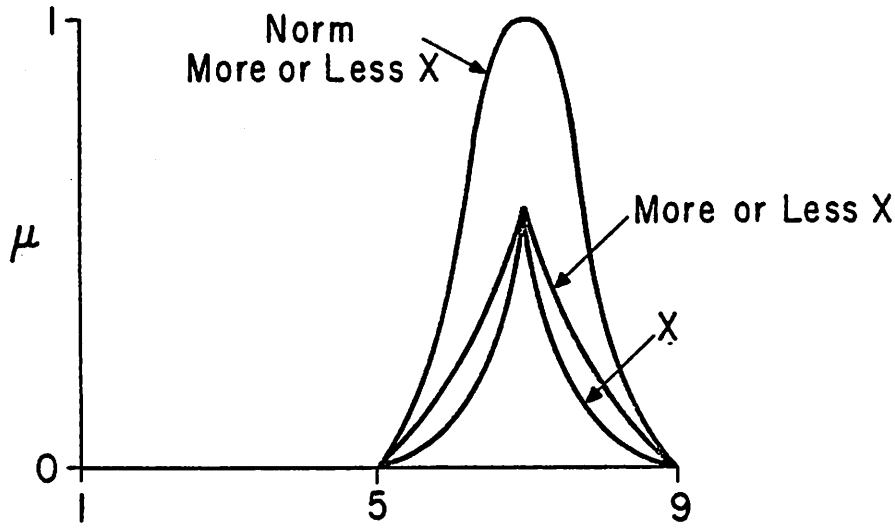

Figure 3.7c Combining portions to arrive at the final phrase

rating phrases to be modeled by normal fuzzy sets. Since the scoring functions preserve normality and convexity (see Chapter 4), the linguistic approximation process is greatly simplified. The final rating will be similar in shape (normal and convex) to the meanings of terms which are generated by the rating language grammar (see Section 3.8).

We give the definitions of two more hedges without detailed explanation. Their relationship to each other and the primary term **low** is illustrated in Fig. 3.8. The two hedges are **fairly** which shifts further than **pretty** and **sortof** which shifts further than **fairly** but still remains to one side of **medium**:

fairly P=NORM (**indeed** P)and **not indeed very** P (3.19)

sortof P=NORM (**moreorless moreorless** P)and **moreorless not** P (3.20)

Recall that **indeed** is an intensifier and steepens the slope of the curve in a slightly different manner from very.

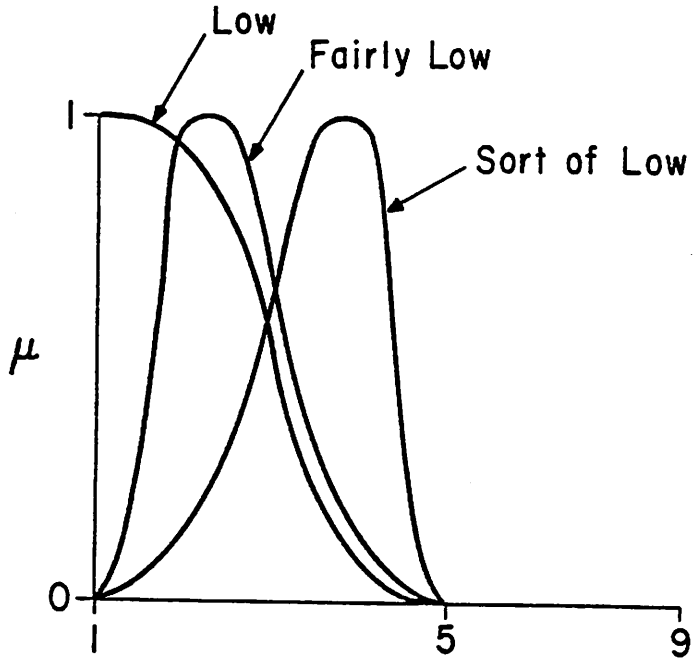

Figure 3.8 Effect of the hedges **fairly** *and* **sortof**

This method of constructing hedges was inspired by Lakoff [LAKOFF 1973] although we have modified his formulations somewhat. A difficulty with hedges designed in this way is that they tend to be sensitive to small changes in the membership functions of the primary values. Additionally, the complexity of the formulations can grow rapidly and soon become counter-intuitive and difficult to analyze. Shaket has suggested a method of hedge construction [SHAKET 1975] which is simple computationally although perhaps less linguistically satisfying. Recall that primary terms in his system are based upon the exponential.

Shaket defines a hedge by explicitly declaring the μ-value of the primary term at which the hedged primary should reach it's maximum. An exponential is then generated which peaks at the base value at which the primary possesses the μ-value of interest.

For example, suppose:

$$\textbf{high} = E(I;\, 9, 2.5)$$

$$= \{.02[4],\ .08[5],\ .24[6],\ .53[7],\ .85[8],\ 1[9]\} \tag{3.21}$$

where I is the universe of discourse made up of the integers 1 to 9, 9 is the peak, and 2.5 is the spread. We have rounded to two places for simplicity. In actuality, there would be no zero μ-values due to the asymptotic behavior of the exponential. We may then define **pretty** to peak when $\mu_{\textbf{high}}$=0.8 and to behave exponentially. Then (see [SHAKET 1975] for more detail):

$$\mu_{\textbf{pretty high}} = E(\mu_{\textbf{high}};\, 0.8, 0.25)$$

$$= .01[6],\ .32[7],\ 1[8],\ .55[9] \tag{3.22}$$

The result has been normalized and rounded for simplicity. The spread value of .25 is given for purposes of illustration and represents one possible choice. Notice that the E function here operates upon the fuzzy set for **high** rather than upon the base scale. Similarly, we can define **fairly** and **sortof**:

$$\mu_{\textbf{fairly}}\, P = E(\mu_P;\, 0.5, 0.25) \tag{3.23}$$

$$\mu_{\textbf{sortof}}\, P = E(\mu_P;\, 0.2, 0.25) \tag{3.24}$$

While this method is conceptually simple, its effective usage requires a deep knowledge of the specific membership functions of the scale end point primary terms. We now suggest a third alternative hedge formulation which operates more directly upon information about the base scale. Suppose we define **pretty** as a hedge which shifts its input fuzzy set one unit *on the base scale.* Thus, **pretty high** will peak at a base value of 8 while **pretty low** peaks at 2. We define the function SHF which takes a shift count and a fuzzy set as input. SHF determines the base value at which the input set peaks, shifts that value toward mid-scale by the count given and generates a fuzzy set of the same form as the input (that is the primary term) with the new peak. For example, using the exponential system:

$$\mu_{\textbf{pretty low}} = \mathrm{SHF}(\mu_{\textbf{low}}, 1)$$

$$= E(I;\, 2, 2.5) \tag{3.25}$$

Of course, this method may also be applied when S and π functions are being used by simply generating a π-pulse at the proper peak.

While less linguistically intuitive since the fuzzy sets themselves are not operated upon directly, this method has the advantage of conceptual simplicity. Additionally, the hedged values will possess fuzzy sets with similar "shapes thus making things easier for the linguistic approximator.

It is likely that none of these methods will provide the ultimate answer to hedge formulation. Further study is needed to determine which (if any) of these schemes most closely models the hedge process in humans. It may be that hedges will ultimately require an empirical formulation of some sort. Therefore, our system allows the user to define his or her own hedge operators and/or primary terms.

3.5 Fuzzy Relations

We now discuss the use of fuzzy relations to gain additional flexibility in formulating ratings. Two particular binary relations are of interest: **lower** and **higher**. We wish to investigate the effect of applying a relation such as **lower** to a (possibly hedged) primary term in the rating system. Recall that in Section 2.6 we defined the meaning of a binary relation applied to a fuzzy set to be the composition of the set with the relation. For example, let the primary term be denoted by P and the relation be labeled R. Then the meaning of a phrase such as **lower than medium** is given by the solution of:

$$P = \textbf{medium}$$

$$R = \textbf{lower}$$

$$\textbf{lower than medium} = P \circ R \tag{3.26}$$

$$= \textbf{medium} \circ \textbf{lower}$$

We therefore define **than** to be a binary operator which performs the composition of its right operand with its left operand. Allowing the **than** operator to perform the function of order reversal allows us to formulate relations in the familiar row-to-column matrix format while retaining the ordering of natural English in writing rating phrases. The idea of associating **than** with composition is due to Wenstop [WENSTOP 1975].

In the interest of simplicity, we will define **higher** and **lower** explicitly and model these relations with matrices. In order to keep the examples short, let us choose the integer interval 1 to 6 as the rating scale (universe of discourse). We define **lower** as:

$$\text{lower} = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & .1 & 0 & 0 & 0 & 0 \\ 1 & .5 & .1 & 0 & 0 & 0 \\ 1 & .8 & .5 & .1 & 0 & 0 \\ 1 & 1 & .8 & .5 & .1 & 0 \\ 1 & 1 & 1 & .8 & .5 & .1 \end{matrix} \tag{3.27}$$

The **higher** relation may be thought of as a matrix which is symmetrical to the above about the diagonal. Notice that we are interpreting **lower** here in a fuzzy manner. For instance, $\mathbf{lower}_{3,3}$=0.1 whereas a non-fuzzy model of **lower** would contain a zero entry for that pair. Also, the property of being **lower** increases with the separation of base scale values. Finally, we recognize the fact that there is a minimum base scale value by setting $\mathbf{lower}_{1,1}$=1. Therefore a more exact interpretation of this relation might be "lower or to a lesser extent similar or minimal". The reader may find these additional properties counter-intuitive to the familiar notion of "lower". As we shall see shortly, a fuzzy notion of **lower** as in (3.27) simplifies the design of *relational hedges.*

We do feel compelled to indicate an alternative formulation, however. One may begin with the familiar notion of "lower":

$$\text{lower} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{matrix} \tag{3.28}$$

and then define relational hedges as *fuzzifiers* (see Section 2.4) to get fuzzy relations such as **slightly lower** or **much higher**. This approach is illustrated briefly in [ZADEH 1972]. Either method of relation definition will serve in general. We choose to work with a fuzzy relation for lower in the interest of simplicity of operation. Since the original relation is fuzzy, we may employ relational hedges which are as simple as the value hedges defined above. The intermediate step of fuzzification is avoided. Additionally, a fuzzy definition of **lower** allows us to model the effect of increasing membership values as the base values move further apart. In other words, the notion of "lower" becomes stronger for base values which are further toward the low end of the scale from the input value when composition is performed.

3.6 Relational Hedges

Returning to our original definition of **lower** (3.27), we next define hedges which operate upon relations in a manner analogous to the primary term hedges discussed earlier. Since our relations are simply fuzzy sets in $U \times U$, many of our earlier hedge formulations can be adapted easily. For example, **not** is just the element by element complement:

$$\textbf{not lower} = \begin{matrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & .9 & 1 & 1 & 1 & 1 \\ 0 & .5 & .9 & 1 & 1 & 1 \\ 0 & .2 & .5 & .9 & 1 & 1 \\ 0 & 0 & .2 & .5 & .9 & 1 \\ 0 & 0 & 0 & .2 & .5 & .9 \end{matrix} \tag{3.29}$$

Notice that **not lower** is not the same thing as **higher** just as **not high** is not identical in meaning to **low**. We may also equate **much** with **very** in meaning. Then for a relation R:

$$\textbf{much } P = \text{CON}(R) \tag{3.30}$$

or equivalently:

$$\mu_{\textbf{very}}\, R = \mu_R^2$$

The compatibility function is now bivariate since R is bivariate. In other words, the hedged relation is also defined in $U \times U$:

$$\textbf{much lower} = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & .01 & 0 & 0 & 0 & 0 \\ 1 & .25 & .01 & 0 & 0 & 0 \\ 1 & .64 & .25 & .01 & 0 & 0 \\ 1 & 1 & .64 & .25 & .01 & 0 \\ 1 & 1 & 1 & .64 & .25 & .01 \end{matrix} \tag{3.31}$$

Similarly, we may define **moreorless lower** in terms of the DIL operator.

As a more complicated example, consider the hedge **slightly** which we define as an analog of the primary term hedge **fairly**:

$$\textbf{slightly } R = \text{NORM}(\textbf{indeed } R) \text{and } \textbf{not indeed much } R \tag{3.32}$$

where **indeed** is just the intensification operation described earlier and **and** is fuzzy set intersection. All these operations are extensions of the simple fuzzy set operators to relations. This just means the base values are ordered pairs rather than singletons. Our implementation of **slightly** (see Appendix B) is somewhat more complicated in that **slightly** (R) is automatically adjusted to have a $\mu_{1,1}$ value of 1 for **lower** or (with the present base scale of 1 to 6) a $\mu_{6,6}$ value of 1 for **higher** so that we preserve the "slightly lower or minimal" property.

Additionally, we round to two places as a matter of course. The result is:

$$\textbf{slightly lower} = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & .04 & 0 & 0 & 0 & 0 \\ 0 & 1 & .04 & 0 & 0 & 0 \\ 0 & .52 & 1 & .04 & 0 & 0 \\ 0 & 0 & .52 & 1 & .04 & 0 \\ 0 & 0 & 0 & .52 & 1 & .04 \end{matrix} \tag{3.33}$$

In each of these cases, the fact that the diagonal entries have been non-zero reflects the fuzziness of the value which will be composed with the relation as illustrated below. If the value to be operated upon by **lower** were non-fuzzy, a relation of the form (3.28) would be more appropriate since a number is not lower than itself. When the number is not precisely known, however, the division between "lower" and "equal" is not as clear. Hence the small but non-zero diagonal values in (3.27), (3.29), (3.31) and (3.33).

We can now give an example of a relation operating upon a rating value. Using our 1 to 6 scale, we define **high**:

$$\textbf{high} = .2[4],\ .8[5],\ 1[6] \tag{3.34}$$

The meaning of the phrase **slightly lower than high** is given by the composition of the primary term (3.34) and the hedged relation (3.33):

$$\textbf{slightly lower than high} = \textbf{high} \circ (\textbf{slightly lower}) \tag{3.35}$$

$$= \begin{matrix} 0 & 0 & 0 & .2 & .8 & 1 \end{matrix} \circ \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & .04 & 1 & .52 & 0 & 0 \\ 0 & 0 & .04 & 1 & .52 & 0 \\ 0 & 0 & 0 & .04 & 1 & .52 \\ 0 & 0 & 0 & 0 & .04 & 1 \\ 0 & 0 & 0 & 0 & 0 & .04 \end{matrix}$$

$$= \begin{matrix} 0 & .2 & .52 & .8 & 1 & .04 \end{matrix}$$

which might approximate to **fairly high** or **pretty high** depending upon the exact form of those hedges.

3.7 Compound Rating Phrases

These relations may be employed along with the previously defined connective **and** to create compound terms which convey finer shades of meaning than a hedged primary alone. As a simple but typical example, consider the compound rating phrase: **higher than - medium and lower than pretty high**. The process is depicted graphically in Figure 3.9 where the curves are to be viewed as qualitative approximations for the purposes of illustration.

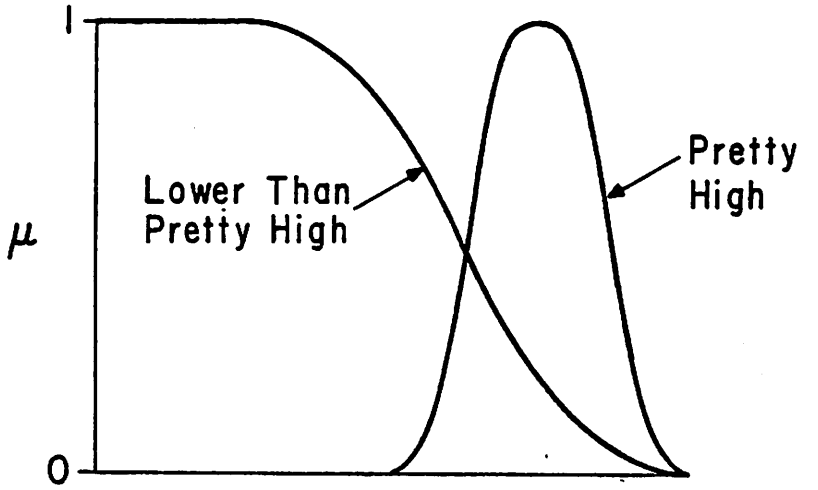

Figure 3.9a Meaning of **lower than pretty high**

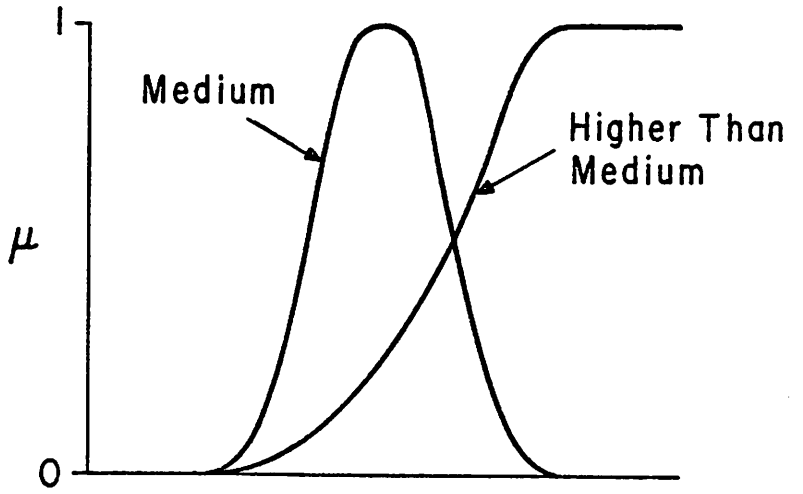

Figure 3.9b Meaning of **higher than medium**

Notice in Figures 3.9a and b that the application of these particular relations generates a pair of monotonic compatibility functions. When these are intersected via **and** and the result

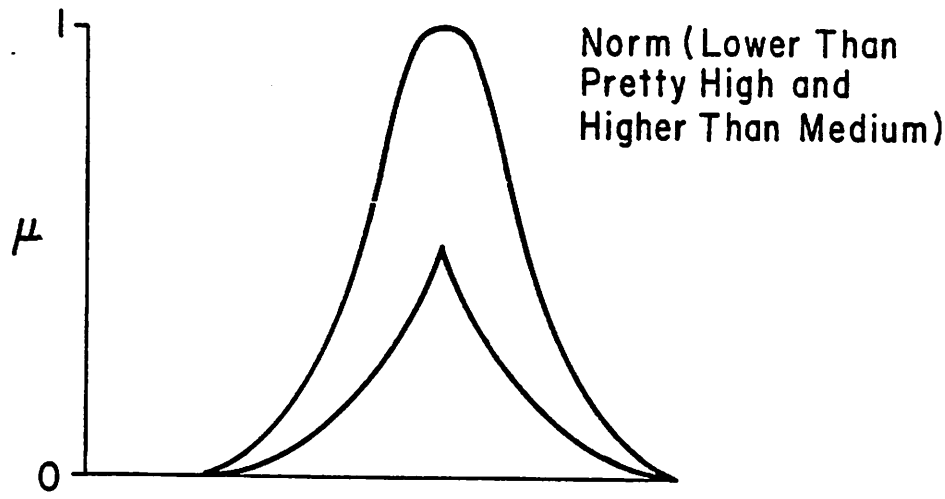

Figure 3.9c The final compound phrase

normalized (see Figure 3.9c) a function which peaks *between* the two original curves is created. In fact, one could define a connective "between" to operate in just this manner although we will not do so here.

3.8 A Sample Rating Language Grammar

Let us now organize all of the above information into a simple rating language. We give the language syntax in the familiar Backus-Naur Form [NAUR 1963]:

<sentence> ::= <compound phrase> | <simple phrase>
<compound phrase> ::= <conjunctive phrase> | <range phrase>
<simple phrase> ::= <relational phrase> | <hedged primary>
<conjunctive phrase> ::= <relational phrase> **and** <relational phrase>
<range phrase> ::= <hedged primary> **to** <hedged primary>
<relational phrase> ::= <composite relation> **than** <hedged primary>
<composite relation> ::= <relation hedge> <relation> | <relation>
<relation hedge> ::= **not** | **much** | **slightly**
<relation> ::= **lower** | **higher**
<hedged primary> ::= <hedge> <primary> | <primary> | <fuzzy number>
<hedge> ::= **not** | **very** | **moreorless** | **fairly** | **pretty** | **sortof** | **really** | **extremely** | **indeed**
<primary> ::= **low** | **high** | **medium**
<fuzzy number> ::= <fuzzifier> <number>
<fuzzifier> ::= **about**
<number> ::= **one** | **two** | **three** | **four** | **five** | **six** | **seven** | **eight** | **nine**

Some of the rating phrases which may be generated with this grammar are:

high
low
medium
not high
moreorless medium
indeed low
medium to sortof high
about four to about six
slightly lower than pretty high
not higher than medium
higher than low and lower than sortof high

3.9 Limitations of the Sample Rating Language

Our sample language consists of a finite number of unique rating "sentences". We have chosen not to include any recursive productions in order to keep the exposition simple. In addition, we allow only a single hedge application in forming a <hedged primary> or a <composite relation>. There are certainly reasonable intuitive arguments for removing these restrictions but doing so leads to some semantic difficulties. A recursive formulation such as **very very high** makes good sense while **pretty pretty low** is much less intuitively satisfying. Good combinations of hedges are not always easy to find. Intensification works well (as in **indeed pretty high**) and negation works well with concentration (**not very high, not extremely low**). Other combinations are uncomfortable: **fairly pretty high, very pretty high**. We adopt the view that our simple grammar should avoid recursion rather than try to isolate semantically meaningful instances. We feel the resulting loss of power will not be critical in most real-world uses of the rating system.

A more sophisticated grammar would sub-categorize the various hedges to prohibit certain combinations. We wanted to give the user maximum flexibility to define the semantics of his or her hedges without having to be concerned with combinations or recursive applications which could yield semantically invalid or intuitively alien phrases. An alternative approach would involve designing hedge functions which are context-sensitive in the semantic sense. Thus a hedge might behave differently when applied to a primary term such as **high** than when applied in combination or recursively.

A related phenomenon results from our interpretation of **medium** as a primary term. While the concept "medium" seems intuitively as simple as **high** or **low** its behavior with hedges argues that it may indeed be a more complex notion. While **moreorless medium** makes sense as a means of increasing fuzziness a phrase such as **pretty medium** implies a peak shift of some sort due to the way the hedge is defined. This is almost certainly not what is desired. Assuming **pretty medium** makes intuitive sense (a debatable proposition in itself) the effect is probably analogous to the application of dilation or concentration. The worst combination is **not medium** which makes semantic sense ("either high or low") but does not have a reasonable interpretation in the context of a rating since it is non-convex. A more complex but less general grammar would probably treat medium as a special case. We will restrict ourselves to this prototype language for the remainder of the presentation.

4. Determination of the Overall Security System Rating

Let us assume a hypothetical rater has analyzed a data processing facility using the rated security system model presented in Section 1.5. We assume a *resistance* has been assigned to each security *feature* in the system using the rating language described in Chapter 3. Given the individual resistances, we wish *infer* a rating for the security system as a whole. This system rating should be expressed using the same rating language which was employed to rate the system elements.

4.1 The MIN Function

One possible method of rating calculation would be to employ worst case analysis. This is in keeping with the "weakest link" philosophy (see Section 1.4). Given a set of resistance values, we compute a composite of the lowest points of each. Therefore, we need a function which takes two fuzzy sets over the same base scale as inputs and returns the "minimum" (in a fuzzy sense). We call this the MIN function. Thus in Figure 4.1, we want the result to be the linguistic approximation of μ_x. This is *not* the same operation as the element by element computation of minimum μ-values. That operation would yield the *intersection* (see Section 2.3) as indicated by the broken lines in the figure. In this case μ_x MIN μ_y will yield μ_x. As we shall see, when the inputs are not well separated the MIN function will yield a composite of the two.

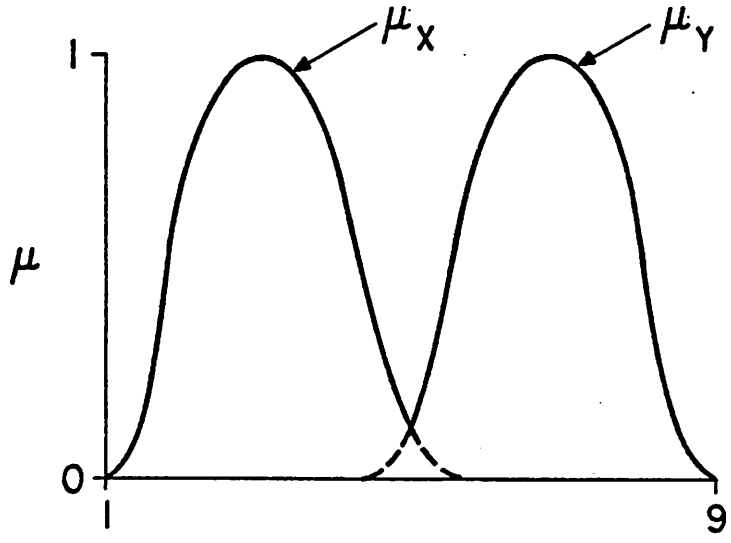

Figure 4.1 The MIN is not the intersection

If we begin with the classical definition of minimum as a binary operator upon non-fuzzy points in some universe of discourse - U, we may employ the extension principle as discussed in Section 2.7 to arrive at a definition of MIN as a binary operator upon *fuzzy subsets* of U. We

select the set of non-negative integers as our *U*. This more is convenient for the construction of examples than using the real numbers. Furthermore, the implementation discussed in Chapter 5 employs the integers as the universe of discourse. Recall that in Section 3.1 we defined our normalized rating scale as the set of integers from 1 to 9. This will be the base scale for all examples in this chapter. Let MIN represent the binary operation which returns the minimum of two fuzzy ratings. The MIN operation will map two fuzzy subsets of *I* back into *I*. We use MIN instead of the more familiar "$\wedge$" because we have used the latter symbol for operations upon μ-values. Computationally, the two operations are identical; it is the conceptual difference we wish to emphasize. Let X and Y represent two fuzzy ratings. Their membership functions are given by:

$$X = \bigcup_{i \in I} \mu_X(i)[i]$$

$$Y = \bigcup_{i \in I} \mu_Y(j)[j] \tag{4.1}$$

Applying the extension principle (see Section 2.7):

$$X \text{ MIN } Y = \bigcup_{i,j \in I} (\mu_X(i) \wedge \mu_Y(j))[i \text{ MIN } j] \tag{4.2}$$

Recall that "$\bigcup$" as used here implies a maximum operator acting upon the membership values of identical base values (see Section 2.3). As an example, consider the MIN of two fuzzy numbers:

$$\textbf{about } 4 = \{.4[3],\ 1[4],\ .6[5]\}$$

$$\textbf{about } 8 = \{.7[7],\ 1[8],\ .3[9]\} \tag{4.3}$$

$$\textbf{about } 4 \text{ MIN } \textbf{about } 8 = \bigcup \begin{Bmatrix} .4[3 \text{ MIN } 7],\ .4[3 \text{ MIN } 8],\ .3[3 \text{ MIN } 9], \\ .7[4 \text{ MIN } 7],\ 1[4 \text{ MIN } 8],\ .3[4 \text{ MIN } 9], \\ .6[5 \text{ MIN } 7],\ .6[5 \text{ MIN } 8],\ .3[5 \text{ MIN } 9] \end{Bmatrix} \tag{4.4}$$

$$= .4[3],\ 1[4],\ .6[5]$$

$$= \textbf{about } 4$$

A security system rated with this "weakest-link" function is presented in Chapter 6.

We have also implemented a variant of the MIN score which we call the "preselected weakest link" score. The rater must provide a *threshold* which is a phrase chosen from the rating language used to assign loss values and likelihoods. This threshold is used as a selector to weed out unimportant security elements in the following manner. In addition to feature resistance, we assume the security system rater has assigned a loss *value* to each security object and a *likelihood* to each threat in the system (see Section 1.5). Considering each security *element* in turn, we first examine the "value" and "likelihood" of that element. Since object values and threat likelihoods are expressed using linguistic ratings (fuzzy sets), we operate upon them with a MAX function which is defined in a manner analogous to the MIN function of (4.2):

$$X \text{ MAX } Y = \bigcup_{i \in I} (\mu_X(i) \wedge \mu_Y(j))[i \; MAX \; j] \tag{4.5}$$

Using MAX we calculate the fuzzy maximum of the object value and threat likelihood for a given element. If this maximum is less than the above mentioned "threshold", the element is ignored in computing the overall system rating. Those elements which remain form a subset of the installation. Intuitively, this subset contains elements of some minimum degree of importance from the standpoint of loss value or threat likelihood. A MIN score is then calculated upon this subset. This "preselection" process allows the user to employ information about the "importance" of the various security elements in his or her installation; less vital security objects will not affect the overall rating. An example of a "preselected weakest link" rating is given in Chapter 6.

The decision to include or reject an element is a binary one which must be made in the presence of uncertainty. The function LT (see Appendix B) performs the "less than" comparison by using the MIN function described above to determine the fuzzy minimum of the threshold and the previously calculated maximum of loss value and threat likelihood. The lesser of the threshold and the maximum is defined to be the one which most closely matches the MIN of the two. This definition is necessary since the fuzzy minimum may actually be a composite of the two. Although this is not the only possible definition of "less than" in a fuzzy environment, it will suffice for our purposes.

4.2 The Fuzzy Mean

Let us now consider alternative scoring functions which are more optimistic and hopefully more realistic for real-world security system analysis. Instead of returning the lowest resistance value, our scoring function will return an "average" resistance which is defined as the analog of the classical numeric mean. The desired situation for two resistance values is illustrated in Figure 4.2. The figure is intended as an intuitive illustration rather than a precise graphical calculation.

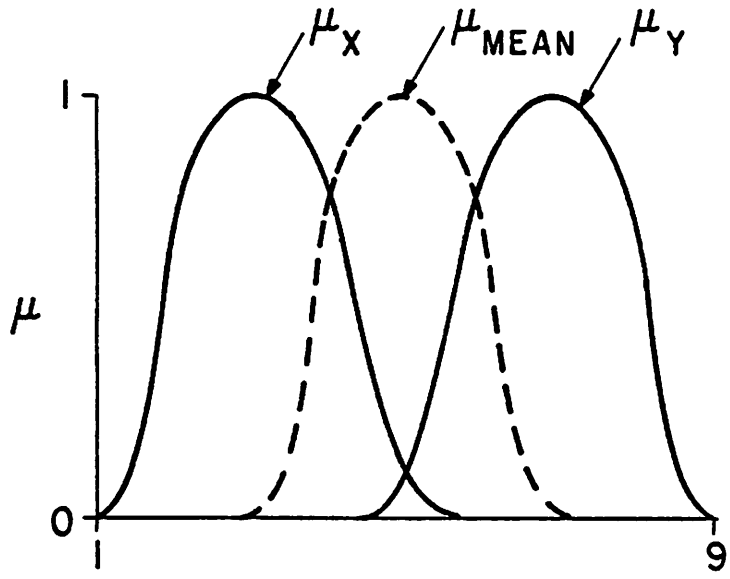

Figure 4.2 A qualitative illustration of the fuzzy mean

If we were working with non-fuzzy values, the mean would be defined as:

$$\bar{r} = \frac{\sum_{i=1}^{n} r_i}{n} \tag{4.6}$$

where each r_i is a (non-fuzzy) resistance rating and n is the number of elements in the system being rated. In order to calculate a "fuzzy mean", however, we must extend the addition operation to accept fuzzy sets. Applying the extension principle (see Section 2.7):

$$X+Y = \bigcup_{i+j \in U} (\mu_X(i) \wedge \mu_Y(j))[i+j] \tag{4.7}$$

where, as usual, "$\wedge$" signifies the taking of the minimum. Recall that the union of fuzzy sets implies a maximum operator acting upon the membership values of identical base values (see Section 2.3).

As an example, consider the addition of the two fuzzy numbers in (4.3):

$$\textbf{about } 4 + \textbf{about } 8 = \bigcup \left\{ \begin{matrix} .4[3+7],\ .4[3+8],\ .3[3+9], \\ .7[4+7],\ \ 1[4+8],\ .3[4+9], \\ .6[5+7],\ .6[5+8],\ .3[5+9] \end{matrix} \right\} \tag{4.8}$$

$$= \{.4[10],\ .7[11],\ 1[12],\ .6[13],\ .3[14]\}$$

which might be interpreted via the process of linguistic approximation as **about** 12.

Therefore, the first step in the calculation of the fuzzy mean is the summation of the set of resistances using fuzzy addition. The result is then divided by the number of elements rated which is a non-fuzzy value. Appealing once more to the extension principle, this division is actually a division of each of the base values by a non-fuzzy singleton since n may be represented as $\{1[n]\}$. Division in this case has no effect upon the μ-values of the fuzzy sum since (by definition) for any μ-value "x" : $1 \wedge x = x$. Thus division by a non-fuzzy singleton amounts to a change in base values only. This is not the general case with fuzzy division as we shall see in Section 4.3 below. Continuing with example (4.8), the fuzzy mean of **about** 4 and **about** 8 is:

$$\frac{\textbf{about } 4 + \textbf{about } 8}{2} = .4[\tfrac{10}{2}],\ .7[\tfrac{11}{2}],\ 1[\tfrac{12}{2}],\ .6[\tfrac{13}{2}],\ .3[\tfrac{14}{2}] \tag{4.9}$$

$$= \{.4[5],\ .7[5.5],\ 1[6],\ .6[6.2],\ .3[7]\}$$

which might be approximated as **about** 6.

A difficulty with the division operation is that it has mapped our result onto the real line. We have chosen to deal with the set of non-negative integers both for simplicity in presenting examples and in anticipation of the discussion of our rating system implementation in Chapter 5. However, the extension principle can be formulated for situations in which the base scale is a continuum such as the real line. Such a formulation is given in [ZADEH 1975b]. We could then choose the real line as our universe of discourse and define our rating scale as the real interval (1,9). The fuzzy mean operation would then map back into this interval. Thus, problems of scale changes and issues of convexity (to be discussed shortly) are more pragmatic than theoretical issues. We will examine these problems in further detail in Chapter 5 when we describe our implementation.

By way of review, the fuzzy mean is defined (in our context) as:

$$\overline{\mathbf{R}} = \frac{\sum_{i=1}^{n} \mathbf{R}_i}{n} \tag{4.10}$$

where the **R**'s are fuzzy resistance ratings and n is the number of elements in the security system under consideration. Unlike (4.6), the $\sum$ operator is here understood to represent fuzzy summation via the addition operation of (4.7). The division by n may be thought of as simply an operation upon base values or (more generally) as a degenerate case of fuzzy division (see Section 4.3).

4.3 The Weighted Fuzzy Mean

One of the shortcomings of the fuzzy mean as a scoring function for rating security systems is that it deals with security feature resistance *only*. We wish to take advantage of other information about the security elements which our model of the *rated security system* (see Section 1.5) provides. In particular, each element possesses *value* and *likelihood* components. We can think of these component values as representing subjective evaluations of the *importance* of each particular element in the system to the overall rating. In other words, value and likelihood are *weights* on resistance. Suppose we assign non-fuzzy numbers from our integer rating scale (1 to 9) to object value and threat likelihood at each element. Assume for simplicity that each weight w is given by:

$$\text{w} = \text{maximum (value, likelihood)} \tag{4.11}$$

where maximum has the usual definition for non-fuzzy numbers. We may then define a *weighted fuzzy mean* scoring function where each of the fuzzy resistance values are multiplied by a non-fuzzy weight before fuzzy addition:

$$\overline{\mathbf{W}} = \frac{\sum_{i=1}^{n} w_i \times \mathbf{R}_i}{\sum_{i=1}^{n} w_i} \tag{4.12}$$

The operation which is represented by "×" is a multiplication of the base values of the various **R**'s. Each weight may be thought of as a non-fuzzy singleton $\{1[w_i]\}$ with the result that multiplication here operates analogously to division in the case of the fuzzy mean. Again, this can be thought of as a limiting case of fuzzy multiplication (see Equation 4.15). Notice the summation in the numerator is a fuzzy operation as in (4.10). The summation in the denominator represents classical addition of the (non-fuzzy) weights which serves to normalize the weighted fuzzy mean into the (real) interval (1,9). This normalization is desirable so that the resultant score ranges over the original base scale thus simplifying the task of the linguistic approximator (see Section 5.6).

As a simple example, using **about** 4 and **about** 8 as defined in (4.3), let:

$$\begin{aligned} \mathbf{R}_1 &= \textbf{about } 4 \\ \mathbf{R}_2 &= \textbf{about } 8 \\ w_1 &= 4 \\ w_2 &= 2 \end{aligned} \tag{4.13}$$

Then the weighted fuzzy mean is given by:

$$\frac{(4 \times \textbf{about } 4) \; + \; (2 \times \textbf{about } 8)}{(4+2)} \tag{4.14}$$

$$= \frac{\{.4[4\times3],\ 1[4\times4],\ .6[4\times5]\} + \{.7[2\times7],\ 1[2\times8],\ .3[2\times9]\}}{6}$$

$$= \bigcup \frac{\left\{\begin{array}{l} .4[12+14],\ .4[12+16],\ .3[12+18], \\ .7[16+14],\ \ 1[16+16],\ .3[16+18], \\ .6[20+14],\ .6[20+16],\ .3[20+18] \end{array}\right\}}{6}$$

$$= \{.4[\tfrac{26}{6}],\ .4[\tfrac{28}{6}],\ .7[\tfrac{30}{6}],\ 1[\tfrac{32}{6}],\ .6[\tfrac{34}{6}],\ .6[\tfrac{36}{6}],\ .3[\tfrac{38}{6}]\}$$

$$= \{.4[4.5],\ .4[4.67],\ .7[5],\ 1[5.4],\ .6[5.67],\ .6[6],\ .3[6.33]\}$$

which might approximate to **about** 5.4 if the final base scale were the real line. Our implementation (see Chapter 5) assumes the base scale is integer valued, however. An approximation under such assumptions would be somewhat less precise. Possible linguistic approximations are: **slightly higher than about** 5 or **about** 5 to **about** 6 or perhaps medium.

4.4 Fuzzy Weighting

In the general case, one would expect the object value and threat likelihood components to be assigned fuzzy values from the same normalized base variable scale as the resistance values. Therefore, weights may also be fuzzy. This leads us to define a more general scoring function which we name the *fuzzy weighted mean.* Here both the resistance and weight values are fuzzy so that multiplication must operate upon fuzzy sets. Furthermore, the sum of the weights will be a fuzzy value as will of course the weighted sum. Evidently, we require definitions for fuzzy multiplication and fuzzy division. Fortunately, both operations may be defined via the extension principle (see Section 2.7) in a manner analogous to the earlier definition of fuzzy addition. Let X and Y be fuzzy ratings as above. Then multiplication is defined via the extension principle as:

$$X \times Y = \bigcup_{i \times j \in U} (\mu_X(i) \wedge \mu_Y(j))[i \times j] \tag{4.15}$$

Similarly, division is defined as:

$$\frac{X}{Y} = \bigcup_{\frac{i}{j} \in R} (\mu_X(i) \wedge \mu_Y(j))[\frac{i}{j}] \tag{4.16}$$

Note that division does not map into the set of positive integers as does addition and multiplication. We signify this by the use of R instead of U in the above definition. This presents no theoretical difficulties if the rating scale is taken as the positive reals but does present pragmatic implementation problems since we assume the base scale is integer valued. Essentially, some of the information resulting from fuzzy division is lost under an integer implementation where the fuzzy sets are necessarily discrete. This issue is treated further in Section 5.4.

The following two examples use the values **about** 4 and **about** 8 from (4.3). First consider multiplication of fuzzy values:

$$\textbf{about } 4 \times \textbf{about } 8 \tag{4.17}$$

$$= \bigcup \left\{ \begin{matrix} .4[3\times7],\ .4[3\times8],\ .3[3\times9], \\ .7[4\times7],\ \ 1[4\times8],\ .3[4\times9], \\ .6[5\times7],\ .6[5\times8],\ .3[5\times9] \end{matrix} \right\}$$

$$= \{.4[21],\ .4[24],\ .3[27],\ .7[28],\ 1[32],\ .3[36],\ .6[35],\ .6[40],\ .3[45]\}$$

which might be approximated as **about** 32. Now consider division:

$$\frac{\textbf{about 8}}{\textbf{about 4}} = \bigcup \left\{ \begin{array}{l} .4[\frac{7}{3}],\ .4[\frac{8}{3}],\ .9[\frac{9}{3}], \\ .7[\frac{7}{4}],\ 1[\frac{8}{4}],\ .3[\frac{9}{4}], \\ .6[\frac{7}{5}],\ .6[\frac{8}{5}],\ .3[\frac{9}{5}] \end{array} \right\} \tag{4.18}$$

$$= \{.4[2.33],\ .4[2.67],\ .3[3],\ .7[1.75],\ 1[2],\ .3[2.5],\ .6[1.4],\ .6[1.6],\ .3[1.8]\}$$

which might approximate as **about** 2.

Notice that in applying multiplication and division to these fuzzy sets over a discrete base scale, convexity is lost. In the multiplication example we have as a portion of the answer set:

$$1[32],\ .6[35],\ .3[36],\ .6[40] \tag{4.19}$$

The compatibility value for 36 is lower than the μ-values at 35 and 40. Similarly, the result of (4.18) shows a value of .3 for the compatibility of 2.5 which is lower than the μ-values at 2.33 and 2.67. We mentioned in Section 2.2 that we would be dealing with fuzzy ratings expressed as normal convex sets. Since the linguistic approximator expects to match the fuzzy score to some combination of linguistic phrases (see Section 5.6), the process will be greatly simplified if the score is represented by a normal convex set. Furthermore, for the reasons outlined in Section 3.4, there is an intuitive appeal to restricting ratings to possess convexity in the fuzzy sets which are their meanings.

Fortunately, the loss of convexity in the above examples is due to the use of a discrete base scale. Suppose we had defined our ratings as fuzzy sets in the real line. Mizumoto and Tanaka have shown [MIZUMOTO 1976] that addition, multiplication, and division (as defined above) are normality preserving. They also prove that addition and multiplication are convexity preserving. Finally, division is convexity preserving if the base scale does not include 0. We may be confident that our calculated rating will possess a final form which is similar to that of the group of linguistic values we employ if we operate with a base scale over some segment of the non-negative reals.

For ease of implementation, we will model fuzzy sets over the non-negative integers. The problem of maintaining convexity becomes a pragmatic rather than theoretical issue. We will employ a form of interpolation to restore convexity when necessary (see Section 5.4).

We are now in a position to define the fuzzy weighted mean. The formula of (4.12) is modified to include fuzzy weights:

$$\overline{W} = \frac{\sum_{i=1}^{n} W_i \times R_i}{\sum_{i=1}^{n} W_i} \tag{4.20}$$

Here "$\sum$" represents fuzzy summation in both the numerator and denominator. The multiplication indicated by "×" is the fuzzy operation of (4.15) and the division which normalizes the score is defined by (4.16).

As a final example, we extend (4.14) to include fuzzy weights and calculate a fuzzy weighted mean. Let:

$$\textbf{about } 2 = \{.5[1],\ 1[2],\ .5[3]\} \tag{4.21}$$

and let **about** 4 and **about** 8 be taken from (4.3). We assume the rater has made the following rating assignments in a security system of two elements:

$$\begin{aligned} \mathbf{R}_1 &= \textbf{about } 4 \\ \mathbf{R}_2 &= \textbf{about } 8 \\ \mathbf{W}_1 &= \textbf{about } 4 \\ \mathbf{W}_2 &= \textbf{about } 2 \end{aligned} \tag{4.22}$$

Then the fuzzy weighted mean of this two-element security system is:

$$\frac{(\textbf{about } 4 \times \textbf{about } 4) + (\textbf{about } 2 \times \textbf{about } 8)}{\textbf{about } 4 + \textbf{about } 2} \tag{4.23}$$

$$= \frac{\bigcup \begin{Bmatrix} .4[3\times3],\ .4[3\times4],\ .4[3\times5], \\ .4[4\times3],\ 1[4\times4],\ .6[4\times5], \\ .4[5\times3],\ .6[5\times4],\ .6[5\times5] \end{Bmatrix} + \bigcup \begin{Bmatrix} .5[1\times7],\ .5[1\times8],\ .3[1\times9], \\ .7[2\times7],\ 1[2\times8],\ .3[2\times9], \\ .5[3\times7],\ .5[3\times8],\ .3[3\times9] \end{Bmatrix}}{\bigcup \begin{Bmatrix} .4[3+1],\ .4[3+2],\ .4[3+3], \\ .5[4+1],\ 1[4+2],\ .5[4+3], \\ .5[5+1],\ .6[5+2],\ .5[5+3] \end{Bmatrix}}$$

$$= \frac{\bigcup \begin{Bmatrix} .4[9],\ .4[12], \\ .4[15],\ 1[16], \\ .6[20],\ .6[25] \end{Bmatrix} + \bigcup \begin{Bmatrix} .5[7],\ .5[8],\ .3[9], \\ .7[14],\ 1[16],\ .3[18], \\ .5[21],\ .5[24],\ .3[27] \end{Bmatrix}}{\{.4[4],\ .5[5],\ 1[6],\ .6[7],\ .5[8]\}}$$

Now addition will yield 54 non-zero μ-values some of which will combine during the union operation of (4.7). Division will then yield approximately 270 values for the fuzzy set which is the final score. In the interest of keeping the example as brief as possible, we will choose only

three fuzzy set elements for each operand. Since we are working with convex sets we will choose elements close to the peak μ-value. This will minimize any loss of information which might otherwise render the example useless. This is done only to simplify this example. In our implementation such intermediate simplifications are not used. A hypothetical computer installation is rated in Chapter 6 using the "fuzzy mean" and "fuzzy weighted mean" as defined here without intermediate simplifications. Choosing the "best" three values in each case:

$$= \frac{\{.4[15],\ 1[16],\ .6[20]\} + \{.7[14],\ 1[16],\ .5[21]\}}{\{.5[5],\ 1[6],\ .6[7]\}}$$

$$= \frac{\left\{\begin{matrix} .4[15+14],\ .4[15+16],\ .4[15+21], \\ .7[16+14],\ 1[16+16],\ .5[16+21], \\ .6[20+14],\ .6[20+16],\ .5[20+21] \end{matrix}\right\}}{\{.5[5],\ 1[6],\ .6[7]\}}$$

$$= \frac{\{.4[29],\ .4[31],\ .6[36],\ .7[30],\ 1[32],\ .6[34],\ .5[37],\ .5[41]\}}{\{.5[5],\ 1[6],\ .6[7]\}}$$

Once again we choose three values from the above fuzzy sum which are nearest the base scale value of 32 and use these in a reduced complexity calculation: calculation: which are near the base value of 32:

$$= \frac{\{.7[30],\ 1[32],\ .6[34]\}}{\{.5[5],\ 1[6],\ .6[7]\}}$$

$$= .5[\tfrac{30}{5}],\ .7[\tfrac{30}{6}],\ .6[\tfrac{30}{7}],\ .5[\tfrac{32}{5}],\ 1[\tfrac{32}{6}],\ .6[\tfrac{32}{7}],\ .5[\tfrac{34}{5}],\ .6[\tfrac{34}{6}],\ .6[\tfrac{34}{7}]$$

$$= .5[6],\ .7[5],\ .6[4.29],\ .5[6.4],\ 1[5.4],\ .6[4.57],\ .5[6.8],\ .6[5.67],\ .6[4.86]$$

which has a μ-value of 1 at 5.4 and so might be approximated as **slightly higher than about** 5. Notice the similarity between these results and the example on the (non-fuzzy) weighted fuzzy

mean. The fuzzy weights serve to add more terms to the resultant fuzzy set which the linguistic approximation process may sometimes use to advantage. We may expect in general a decrease in the sharpness of the peak of the resultant fuzzy set (viewed as a curve). It is intuitively reasonable that these "fuzzy arithmetic" operations increase the fuzziness of the final rating. This is the penalty we pay for the use of fuzzy values throughout the rating process. We are trading precision for utility and flexibility of expression. The end goal is the production of a score which is realistic though inexact.

5. Overview of the Implementation

A complete listing of the *APL* code which constitutes our Security Rating Calculator is given in Appendix B. In this chapter we wish to give the reader a general idea of the system operation and to discuss some of our design decisions and their consequences. We assume the reader has a basic knowledge of programming in *APL* [GILMAN 1974], [GREY 1973], [WIEDMANN 1974]. In addition we employ certain features of *APL*PLUS* [STS 1974], in particular the file subsystem.

5.1 Fuzzy Sets as *APL* Vectors

Our most fundamental design decision was to model fuzzy sets as vectors (i.e. arrays of one dimension). This familiar data structure is simple and yet powerful enough to facilitate the implementation of all of the operations upon fuzzy sets described in Chapter 2. The decision to employ discrete rather than continuous fuzzy sets has some implications for the fuzzy arithmetic operations of Chapter 4. Within certain limits, these difficulties are manageable. We discuss this problem in detail below. Since vectors are the fundamental entity in our implementation, the attractiveness of *APL* is obvious. All of the basic mathematical operations in *APL* extend to multi-element variables in a natural way. Both unary and binary *APL* operators act upon vector operands by performing the operation in question on an element by element basis [GILMAN 1974]. This is exactly the effect of the majority of the fuzzy set operations we have discussed.

The contents of an *APL* vector is usually displayed as a sequence of values:

$$0\ 0.2\ 1\ 0.5\ 0\ 0\ 0.1\ 0\ 0 \tag{5.1}$$

The index of the leftmost value is assumed to be 1 and the remaining indices are just the integers in ascending order. Vector element values may be any set of valid *APL* constants [GILMAN 1974]. We will restrict ourselves to values in the real line interval (0,1). As mentioned in Section 3.1, we assume a normalized rating scale which is dimensionless. It is therefore convenient to consider the *APL* index of each vector element to be a base value and the element value to be its grade of membership (μ-value) in the fuzzy subset which the vector represents. Under this interpretation, the above vector represents the fuzzy set (using our earlier notation):

$$\{0.2[2],\ 1[3],\ 0.5[4],\ 0.1[7]\} \tag{5.2}$$

5.2 Linguistic Modifiers in *APL*

There are additional advantages in using *APL* to model a linguistic rating system. The syntax of *APL* functions is such that monadic functions expect a parameter immediately to the right of the function name and dyadic functions expect a parameter on the left and the right (infix notation). Blanks are used as separators. If we use the various *linguistic values* in our rating language (see Chapter 3) as the *names* of *APL* variables and functions, the writing of a complex rating phrase corresponds directly to a series of *APL* function evaluations. That is, a rating phrase is also a legal *APL* expression. Furthermore, expression evaluation in *APL* proceeds strictly right to left with no operator precedence. This is quite natural for most of the linguistic expressions we shall use. Exceptions of interest occur in the use of the connectives (**and**, **to**) and in composition (**than**) where parentheses are sometimes needed. For example, suppose *HIGH* is the name of an *APL* variable which is currently assigned a value which is, say, a nine element vector of μ-values. Let *VERY* be the name of a monadic *APL* function which returns the square of its input (see Appendix B). Then execution of the *APL* statement:

$$VERY\ HIGH \tag{5.3}$$

yields a nine-element vector each element of which is the square of the corresponding element value in the vector *HIGH*. This is exactly the semantic operation we wish to model for the hedge **very** (see Section 3.3). Similarly, *AND* is the name of an dyadic *APL* function which applies the "minimum" operator to its two inputs thus achieving the intersection operation which represents the semantics of the **and** connective (see Section 3.4).

Operating with fuzzy relations is also quite easy in *APL*. In Section 2.5 we modeled a relation in a discrete universe of discourse as a matrix (array of two dimensions). Such arrays are directly available as data structures in *APL*. All basic *APL* operators act upon arrays on an element by element basis [GILMAN 1974]. Thus, relational hedges are simply modeled as monadic functions which receive a matrix (i.e. a relation) as input. Furthermore, the definition of composition of relations via the max-min inner product [ZADEH 1975b] is quite straightforward to implement in *APL*. Recall that **than** is the linguistic operator which performs this composition (see Section 3.5). The syntax of composition (3.35) requires that the operand order be reversed:

$$\textbf{lower than high} = \textbf{high} \circ \textbf{lower} \tag{5.4}$$

where "$\circ$" represents composition (see Section 2.5). We implement "$\circ$" as a max-min inner product. In *APL* the operator "." performs a generalized matrix inner product operation

[GILMAN 1974]. The operator is "generalized" in the sense that any binary operator pair may be chosen for the row and column operations. Recall that " × " represents multiplication in *APL* [GILMAN 1974]. Then conventional matrix multiplication may be formulated in *APL* as:

$$A \circ B \leftarrow A +.\times B \tag{5.5}$$

If we replace "+" by "⌈" which is the *APL* maximum operator and " × " by "⌊" (the *APL* minimum operator) we have a formulation of the max-min product. Our implementation of relational composition takes the form of a dyadic *APL* function named *THAN* (see Appendix B). If A and B are fuzzy sets (vectors or matrices):

$$A \ THAN \ B \leftarrow B\lceil.\lfloor A \tag{5.6}$$

performs the desired composition.

5.3 Implementation of Fuzzy Arithmetic

The modeling of the fuzzy arithmetic operators of Chapter 4 is slightly more complex. The additional complexity is due to the fact that the universe of discourse changes. Recall that in Section 4.2 we defined fuzzy addition as a mapping from the base scale to the set of all possible sums of the base scale elements with themselves (i.e. a"×"product). Thus the fuzzy sum of two sets defined on a base scale of 1 to 9 results in a set (vector) over the range 2 to 18. This turns out to be straightforward to implement in *APL*. The main problem is the proper accumulation of intermediate "sums". As an aid to the reader we present a brief example of the operation of the *APL* function *PLUS* (see Appendix B). In the interest of brevity we will assume a base scale of 1 to 4. Let:

$$A = .5 \ \ 1 \ \ .4 \ \ 0$$

$$B = .2 \ \ .5 \ \ .8 \ \ 1 \tag{5.7}$$

The execution of A *PLUS* B proceeds as follows. First A_1 is combined with each element of B using the "∧" (minimum) operator. This implements the definition of *PLUS* using the extension principle (see Section 4.2). The (intermediate) result is:

$$.2 \ \ .5 \ \ .5 \ \ .5 \tag{5.8}$$

The base scale of the intermediate result is simply the result of the integer addition of the vector indices involved in this step: (1+1), (1+2), (1+3), (1+4) or the interval (2 to 5). Next

A_2 is combined with each element of B yielding the intermediate result:

$$.2 \quad .5 \quad .8 \quad 1 \tag{5.9}$$

However, the base scale has shifted since the index sums are: (2+1), (2+2), (2+3), (2+4) giving a new interval (3 to 6). The μ-values of corresponding base values in each of these intermediate results must be combined using the "$\vee$" maximum operator in accordance with definition (4.7). We accomplish this in *APL* by first adding leading and trailing zeros to properly align the vectors to a scale of (1 to 6) where the zero for index 1 is included because *APL* normally assumes vectors are indexed from 1. Then:

$$\begin{array}{c} 0 \quad .2 \quad .5 \quad .5 \quad .5 \quad 0 \\ \vee \\ 0 \quad 0 \quad .2 \quad .5 \quad .8 \quad 1 \end{array} \tag{5.10}$$

yields:

$$0 \quad .2 \quad .5 \quad .5 \quad .8 \quad 1$$

Next $A_3 \wedge B$ yields a vector over (4 to 7):

$$.2 \quad .4 \quad .4 \quad .4 \tag{5.11}$$

which is adjusted to:

$$\begin{array}{c} 0 \quad 0 \quad 0 \quad .2 \quad .4 \quad .4 \quad .4 \\ \vee \\ 0 \quad .2 \quad .5 \quad .5 \quad .8 \quad 1 \quad 0 \end{array} \tag{5.12}$$

which yields:

$$0 \quad .2 \quad .5 \quad .5 \quad .8 \quad 1 \quad .4$$

Since $A_4 = 0$ it offers no contribution to the fuzzy sum other than the lengthening of the vector to conform to the interval of the final result (1 to 8):

$$0 \quad .2 \quad .5 \quad .5 \quad .8 \quad 1 \quad .4 \quad 0 \tag{5.13}$$

Notice that our result has a maximum at a base value of 6 which is intuitively plausible since A exhibits a maximum compatibility value at 2 and B has a maximum at 4.

Fuzzy multiplication is implemented in an analogous fashion. In this case the scale shifts are more complex than for addition. For instance, given the A and B values of (5.7), $A_1 \wedge B$ now maps to (1×1), (1×2), (1×3), (1×4) and $A_2 \wedge B$yields (2×1), (2×2), (2×3), (2×4). We must combine intermediate results which are defined over (1,2,3,4) and (2,4,6,8). Therefore, it is necessary to *insert* zeros in the resultant vectors as well as adding them to the beginning and end. The reader may examine the code for the function *TIMES* (see Appendix B) to see how this is done. We will show only the intermediate results of multiplication using (5.7) as operands. These values are repeated here for convenience:

$$A = .5 \quad 1 \quad .4 \quad 0 \tag{5.7}$$

$$B = .2 \quad .5 \quad .8 \quad 1$$

A TIMES B =

.2 .5 .5 .5 0 0 0 0 0 0 0 0 0 0 0 0

∨

0 .2 0 .5 0 .8 0 1 0 0 0 0 0 0 0 0

∨ (5.14)

0 0 .2 0 0 .4 0 0 .4 0 0 .4 0 0 0 0

∨

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

= .2 .5 .5 .5 0 .8 0 1 .4 0 0 .4 0 0 0 0

The fuzzy product shows a maximum μ-value at 8 on the base scale. The result is thus at least intuitively reasonable since A peaks at 2 and B peaks at 4.

5.4 Convexity

Notice that the result of (5.14) is *not* a convex fuzzy set according to definition (2.8). In particular, there are zero μ-values at 5, 7, 10, and 11 which should be non-zero. This occurs because no product of two integers in the interval (1 to 4) will yield these values. We inserted zeros in these positions when shifting the intermediate result vectors before combination with the "$\vee$" operator. As pointed out in Section 4.4, the results of Mizumoto and Tanaka [MIZUMOTO 1976] show that convexity is preserved when the universe of discourse is the real line. Therefore, we may be assured that our loss of convexity is a purely pragmatic consequence of modeling fuzzy sets over a discrete universe. Accordingly, we will restore convexity using "interpolation" for the benefit of the linguistic approximator and to insure that successive operations (e.g. multiplication followed by division) will not intensify the problem.

There are several ways in which this loss of convexity may be remedied. The most conservative approach would be to adjust all low μ-values which lie *between* higher values in the vector which represents the fuzzy set. From the definition of convexity (2.8):

$$(i \leqslant j \leqslant k) \rightarrow \mu_X(j) \geqslant (\mu_X(i) \wedge \mu_X(k)) \tag{5.15}$$

we can assume that all zero values must be corrected to be at least as large as the smallest of their nearest surrounding values in the vector. Applying this form of "interpolation" to the result of example (5.14):

$$A \ TIMES \ B = .2 \ .5 \ .5 \ .5 \ .5 \ .8 \ .8 \ 1 \ .4 \ .4 \ .4 \ .4 \ 0 \ 0 \ 0 \ 0 \tag{5.16}$$

Of course, given more information about the shape of the compatibility functions of A and B, more complex types of interpolation might be employed. If it is known that the fuzzy sets used will always exhibit smooth and monotonic behavior on either side of the peak point(s), it would be possible to use linear interpolation (or non-linear curve fitting techniques) to change the "stair step" function of (5.16) to a smoother form. This might make things easier on the linguistic approximator but such an approach requires certain (perhaps unwarranted) assumptions about the shape of the compatibility functions which define the semantics of the rating language. We prefer to change the intermediate fuzzy sets as little as possible. The code for our "interpolation routine *CONVEX* is given in Appendix B.

5.5 Fuzzy Division and the Fuzzy Weighted Mean

We now turn to some of the difficulties involved in the implementation of fuzzy division in an integer environment. Our motivation for defining fuzzy division is to make possible the calculation of the fuzzy weighted mean (see Section 4.4). It is important to emphasize that we are not trying to exactly duplicate the operation of the classical mean or weighted mean. Rather, we are trying to construct a mechanism which will act upon fuzzy values in an analogous way. The result of the fuzzy calculation (when properly interpreted through linguistic approximation) should possess properties which are qualitatively similar to the results of applying the classical mean or weighted mean when the environment is precise numbers.

Fuzzy division is implemented in a manner analogous to multiplication and addition. In the case of division we do not have to deal with an increase in vector size. This is due to the fact that division maps onto the real line (see Section 4.4) and so the increase in the number of set elements is due to the generation of non-integer base values by the division process. Since we cannot model non-integer base values directly, we choose to ignore those quotients which do not fall back into the original rating scale integer set.

There is, of course, a loss of information in this process. We feel the problem is not too severe from a practical standpoint since we use division only once in the rating calculation process and then only to normalize the fuzzy sum (if calculating a fuzzy mean) or fuzzy weighted score back to the original rating scale interval. We do this so that the linguistic approximator may operate with a single group of rating language terms which are defined over the original integer scale. The code for the function *DIVBY* is given in Appendix B. Our next example illustrates division using the result of (5.14):

(A TIMES B) DIVBY A =

$$.2\ .5\ .5\ .5\ 0\ .8\ 0\ 1\ .4\ 0\ 0\ .4\ 0\ 0\ 0\ 0$$

$$\textit{DIVBY} \qquad (5.17)$$

$$.5\ 1\ .4\ 0$$

Looking at the effects of the first three divisor elements:

$$.2\ .5\ .5\ .5\ 0\ .5\ 0\ .5\ .4\ 0\ 0\ .4\ 0\ 0\ 0\ 0$$

$$\vee$$

$$.5\ .5\ .8\ 1\ 0\ .4\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

$$\vee$$

$$.4\ .4\ .4\ .4\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

We ignore the contribution of A_4 which will be a vector of all zeros. Taking the maximums yields:

$$.5\ .5\ .8\ 1\ 0\ .5\ 0\ .5\ .4\ 0\ 0\ .4\ 0\ 0\ 0\ 0$$

which might be approximated as **about** 4. In practice, the linguistic approximator assumes the result lies entirely in the interval which defines the original rating phrases. In examples (5.14) and (5.17) the base scale is 1 to 4. We therefore disregard μ-values associated with base values above 4 when performing linguistic approximation. The information which is lost is not really of use in finding a rating phrase which most nearly fits the resultant fuzzy set in meaning since all combinations of rating phrases still result in a fuzzy set in the original base interval. When the result of (5.17) is truncated for linguistic approximation we have:

$$(A\ TIMES\ B)\ \ DIVBY\ A\ =\ .5\ \ .5\ \ .8\ \ 1 \qquad (5.18)$$

It is particularly important to note that we do not arrive at our original value for B:

$$B\ =\ .2\ \ .5\ \ .8\ \ 1\ \neq\ .5\ \ .5\ \ .8\ \ 1 \qquad (5.19)$$

That is, fuzzy division is not the exact inverse of fuzzy multiplication. In fact, fuzzy numbers do not possess inverses. This is proven in detail in [MIZUMOTO 1976]. We must therefore rely upon the robustness of the linguistic approximator. The approximation process must not demand that the fit of our final result and the closest rating phrase be too exact.

Another practical difficulty with fuzzy arithmetic operations is that their repeated application tends to increase the fuzziness of the accumulated score. The fact that the resultant "curve" may be quite broad makes it difficult for the linguistic approximator to arrive at a relatively simple phrase which closely matches the result in "meaning". This is especially true in calculating a score in which the resistances and weights are all fuzzy.

There is a certain amount of intuitive justification for this. We would expect a score based upon some combination of fuzzy values to be at least as fuzzy as any individual value.

Nevertheless, from the practical standpoint of performing linguistic approximation, it may be necessary to apply concentration (see Section 2.4) during the process to allow the approximator to make a reasonable guess. A good rule of thumb might be to apply concentration (if necessary) to reduce the breadth of the resultant "curve" so that it is about as fuzzy as the fuzziest of the original rating values assigned during the scoring process. We have not investigated this problem in great detail and therefore do not apply such modifications to the final result in our implementation. In the case of the real-world security systems evaluated by the student raters (see Chapter 6), the results of fuzzy weighted mean scoring were found to be quite complex linguistic phrases. Nevertheless, the raters reported that the results were not so broad as to defy interpretation. The problem of increasing fuzziness requires much more study to determine its impact upon practical rating systems.

5.6 Linguistic Approximation

A detailed description of the linguistic approximator will not be presented here. The interested reader should examine the *APL* code in Appendix B. For now, we are interested in its overall operation. The approximator accepts two inputs. The first is a vector (fuzzy set) which is the overall rating of the security system calculated by some scoring function. The second input is a character matrix which is a list of the linguistic terms in the rating vocabulary and their syntactic categories. Thus, the rating vocabulary is parameterized. The grammar is fixed in the current implementation and corresponds to the rating language presented in Section 3.8. Extending the linguistic approximator to handle a user defined grammar is discussed in Section 7.4.

The underlying assumption in the operation of the linguistic approximator is that the final rating phrase will be some syntactically legal combination of terms from the rating vocabulary. First all of the primary terms (e.g. **high**, **low**, **medium**) are considered. Each is executed as an *APL* expression and the resulting vector is compared on an element-by element basis with the "score" vector which is the first input mentioned above. The sum of the squares of the differences in μ-values of corresponding vector elements is computed and retained. Henceforth, "nearest" will mean minimal in the sum of squares deviation. The primary term on the low side of the rating scale which is nearest to the unknown score is remembered as well as the nearest primary on the high side. The idea is to bracket the unknown score and attempt to move nearer to it by applying modifiers to the primary terms.

Next the hedges are executed one-by-one as *APL* functions with the low side primary term as hedge input. The hedged primary which is nearest is remembered along with the new sum of squares deviation and the process is repeated with the high side hedge. Currently, only

one pass is made through the hedges. It should be quite easy to allow combinations of hedges as well as recursive use of hedges if the hedges are properly constructed.

The best low hedged primary and best high hedged primary are joined with the **to** operator and the result compared with the unknown. If no improvement in the sum of squares error is found this compound phrase is rejected. Next the best low hedged primary (possibly a primary itself if no improvement resulted from hedging) is composed with the **higher** relation. The same process occurs with the upper hedged primary and the **lower** relation. The two relational phrases are connected via the **and** connective and tested against the unknown. If no improvement in error is found this phrase is ignored. Each of the two relational phrases is combined with the set of relational hedges such as **slightly** and **much** and tested against the unknown. In all cases the phrase which shows the best fit is retained as the current approximation.

Thus the approximator may output one of the following types of phrases:

primary	**(high)**
hedged primary	**(ratherlow)**
compound phrase	**(high to sortof low)**
relational phrase	**(lower than pretty high)**
compound relational phrase	**(lower than high and higher than medium)**
hedged relational phrase	**(slightly higher than medium)**

The approximator will thus investigate a "reasonable" subset of the phrase combinations allowed by the (non-recursive) grammar of Section 3.8. In all cases the best single combination is output. It may be that the approximator arrives at a very close solution early in the search. The process may be terminated early when the sum of squares deviation is less than *EPSILON*, a global scalar variable which may be set by the user.

5.7 Using the Rating Calculator

Our software rating calculator is a file driven system. The *APL*PLUS* file subsystem functions [STS 1974] are used to manipulate two files for each user. One file holds the user's personal set of *APL* functions which transform linguistic values into fuzzy sets. That is, these are the functions which model primary terms, hedges, relations, etc. A default file is provided by our system based upon the grammar of Section 3.8 and the semantics forms discussed in Chapter 3. The second file contains the rating phrases assigned to the security elements of the computer facility being evaluated.

The first phase of security system evaluation comprises the construction of *APL* functions for the rating language semantics. The user may execute *GETSEMANTICS* (see Appendix B) which will enable him or her to construct a file of personal semantics in the form of *APL* functions. These are loaded into the rating calculator by executing The function *SEMANTICS*. This routine reads the file of semantics in as character matrices and converts them to active *APL* functions. The symbol table for the linguistic approximator is also constructed at this time.

The second phase constitutes the input of the rating phrases which are the subjective measures of object value, threat likelihood, and feature resistance at each security element (see Section 1.6) in the data processing installation being rated. The user executes the function *GETRATINGS* to accomplish this. Each triple of rating phrases is written onto a user-named file as a separate record. Thus each record represents an element in the rated system. In the current implementation no syntax or semantic checking of input phrases is performed. Input checking is an area for future work (see Section 7.5). It was decided to record the ratings as character phrases so that a user could preserve his or her own list of element scores, possibly run several scoring functions, change the rating language semantics by changing his or her semantics file and rerun the scoring functions without constructing a duplicate rating file.

In the third phase one of the scoring functions described in Chapter 4 is executed. The *APL* code for these functions may be found in Appendix B. The ratings phrases are read from the user's file. Each resistance value (for example) is in the form of a phrase from the rating language and is executed as an *APL* statement. This results in the execution of the various semantic functions supplied by the user in phase 1. The resultant vector (the meaning of the resistance as a fuzzy set) is input to the proper scoring function and the process iterates until the entire ratings file is read.

The output of the chosen scoring function serves as the input to the linguistic approximator function *LAPPROX* (see Appendix B) during the fourth and final phase of the scoring process. The action of the linguistic approximator has been discussed above. The output of the approximator is a phrase in the original rating language which represents the overall linguistic security rating of the computer installation under study.

6. Sample System Ratings

We now present an example of the use of the rating calculator in evaluating a data processing installation. The rating calculator was exercised by several students in conjunction with a graduate course in computer security at University of California, Berkeley during the winter term of 1977. Each of the students selected a "real-world" data processing installation with which he or she had a high degree of familiarity and rated the various security "elements" (see Section 1.6). The students performed the decomposition of the installation to be rated via a hierarchical analysis program designed by Michelman [MICHELMAN 1977] which served as an input/output "front end" to our ratings calculator.

It was not the purpose of these exercises to establish a "correct" evaluation for any of the installations rated. Since some subjectivity is involved in the rating process, the notion of a single "correct" rating is not very meaningful or important to the problem of security evaluation. We felt it *was* important, however, to establish the practicality of using the model of Chapter 1 as a vehicle for the analysis and evaluation of security systems. We were also interested in getting some feel for the difficulty involved in developing a rating language and "training" a rater to a language. A "standard" rating language was designed based upon the grammar of Section 3.8 with the stipulation that minor changes in vocabulary would be accommodated. For example, one user preferred the terms **less** and **greater** to **higher** and **lower**. The "standard" semantics (i.e. compatibility functions) of Chapter 3 were also employed rather than attempt to have each student rater provide his or her own semantics. This approach seemed advisable given the current primitive state of the art in determining language semantic functions specific to the individual.

While it is impossible to draw very strong conclusions from the small amount of testing which the student exercises provided, some useful observations can be made. None of the raters expressed strong objections to the rating language. There was some disagreement about the relative effect of some of the hedges which indicates that, in the long run, an individual language approach may be best. Nevertheless, all of the raters were able to adjust their thinking to conform when necessary. This indicates that the language design problem, while certainly non-trivial, should not be insurmountable. It was encouraging to discover that none of the raters found the calculated scores for their systems to be bizarre or very different from their own intuitive estimates of the overall performance of their systems. More importantly, several indicated that when they found scores to be lower or higher than expected, a closer examination of their system analysis revealed elements that they had overlooked or rated erroneously. There were comments that the way they were forced to enumerate all of the system elements was quite useful. This exhaustive property of our methodology is, we feel, a very important feature.

We resist the temptation to present the description and ratings of one of the installations studied during the student exercises discussed above. A set of 100 or more security elements was quite common in the analysis of these installations. Such a presentation would be quite lengthy and difficult for the reader to assimilate. It is our intention here to give the "flavor" of security analysis without losing the reader in a vast amount of detail. Our example should be easily understandable and illustrative. We therefore present a small and fictitious system analysis which is based upon and typical of the installations which were used in the student study. In using a totally hypothetical installation as an illustration here we avoid the need to justify the validity of the individual object value, threat likelihood, and feature resistance ratings as exact real-world component scores. Accordingly, the linguistic values to be presented should be viewed as arbitrary but typical.

6.1 The Sample Installation

The security objects in our sample installation are listed in outline form in Figure 6.1. The installation has been divided into three main areas more or less by function. The components listed under each of the main areas taken together make up the set O (see Section 1.3).

1. Hardware

- **1.1** CPU
- **1.2** Main memory
- **1.3** Secondary memory
- **1.4** Input/output devices

2. Software

- **2.1** Operating system
- **2.2** Applications Programs
- **2.3** Data

3. The Computer Center

- **3.1** Operations area
- **3.2** Data input/output area

Figure 6.1 Sample installation - the security objects

Of course it is likely that there would be many more objects to consider in a complex real facility. This sample analysis may be thought of as a high level or breadth first initial look at the overall installation. Using a hierarchical system for specifying security elements

[MICHELMAN 1977], the rater may easily pursue whatever level of detail is most appropriate.

Recall that in the *rated system model* (see Section 1.5) the "element" relation is made up of the objects (O) and their associated "values" (V) along with the set of threats (T) and their associated "likelihoods" (L) and, finally, the security features (F) with their "resistances" (R). The element relation for our sample system is presented in Figure 6.2 in tabular form. In order to make the example more concrete and realistic, we employ functional descriptions instead of our original subscripted letter notation in identifying set elements. The assigned ratings appear in **boldface** in accordance with our convention for identifying linguistic values. These ratings might vary considerably from installation to installation and perhaps from rater to rater. A specific installation may possess any or all of these security elements as well as many more which we have omitted in the interest of brevity.

Element	Object Name	Value	Threat Name	Likelihood	Feature Name	Resistance
1	CPU	**high**	Malicious Destruction	**pretty low**	Guard	**medium**
2	CPU	**high**	Hardware Tampering	**fairly high**	Alarmed Cabinets	**high**
3	Main Memory	**medium**	Hardware Tampering	**fairly high**	Alarmed Cabinets	**high**
4	Secondary Memory	**pretty high**	Human Error	**low**	Volume Labels	**low**
5	Secondary Memory	**pretty high**	Unauthorized Read	**very high**	Volume Labels	**pretty high**
6	I/O Devices	**fairly low**	Hardware Tampering	**medium**	Keys on Terminals	**pretty low**
7	Operating System	**high**	Modifying OS Routines	**pretty high**	Privileged Programs	**sortof high**
8	Operating System	**high**	Defective Implementation	**high**	Validation Programs	**medium**
9	Applications Programs	**low**	Improper Operation	**medium**	Explicit Documentation	**pretty high**
10	Data	**very high**	Unsecured Storage Media	**high**	Library Facility	**fairly low**
11	Data	**very high**	Exposed Output	**high**	Physical Security	**sortof high**
12	Operations Area	**medium**	Natural Calamities	**low**	Building Construction	**sortof high**
13	Operations Area	**medium**	Manmade Disasters	**medium**	Contingency Plans	**fairly high**
14	Data I/O Area	**fairly low**	Unauthorized Intruders	**fairly low**	Guard	**high**

Figure 6.2 Sample Installation - the element relation

Figure 6.2 illustrates the need for a true relation rather than a single-valued mapping in specifying elements. A portion of the relation (elements 1, 2, 3, and 6) has been extracted and converted to the graphical form of the original model in Figure 6.3 to emphasize this.

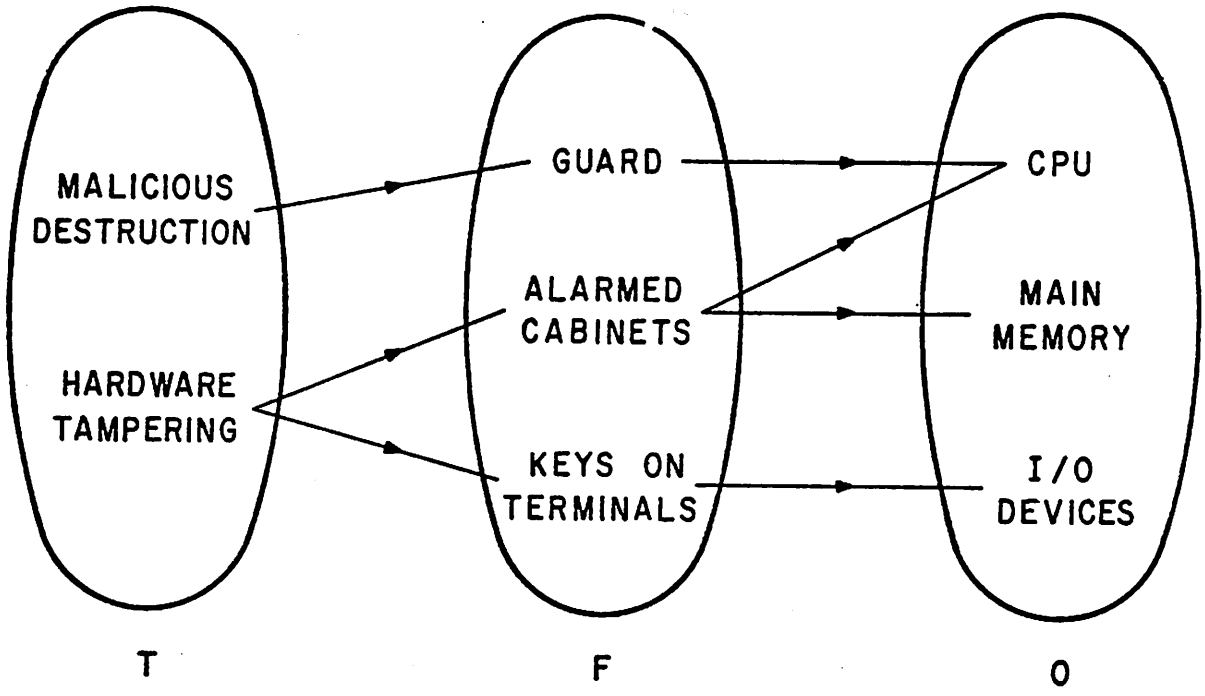

Figure 6.3 A portion of the sample system element relation

Notice that the "alarmed cabinets" device is an example of a security feature which protects more than one object from a single threat. Similarly, the threat of "hardware tampering" evokes a different feature depending upon the object being protected. It is also worth pointing out that the feature "volume labels" is used in elements 4 and 5 with different resistance values; the rating is dependent upon the context of the entity being rated.

6.2 Ratings for the Sample System

The scenario for use of the rating calculator was outlined in Section 5.7 and will not be repeated here. We assume the value, likelihood, and resistance of each element have been entered and stored in a file. We will employ the semantic functions discussed in Chapter 3 as the rating language for this sample system. Overall scores for this system are displayed in Figure 6.4 which shows the results of invoking four of the scoring functions described in Chapter 4.

One possible method of rating calculation would employ worst case analysis in keeping with the "weakest link" philosophy. From the file of component ratings which define the element relation, only the resistance linguistic values are chosen. The minimum resistance in the installation is calculated by the *WEAKLINK* scoring function (see Appendix B) as is illustrated in Figure 6.4a.

```
WEAKLINK
RATINGS FILE NAME? SAMPSYS
LOW
```

Figure 6.4a Sample Installation - Weakest Link Rating

The reader may verify by inspection of the list of resistances in Figure 6.2 that the overall score of **low** is correct.

It may be the case that a weakest link score is desired with only high value objects and high likelihood threats considered. We have termed this scoring function the "preselected weakest link" (see Section 4.1). In this case a threshold must be supplied in the form of a linguistic value as shown in Figure 6.4b.

```
SELWEAKLINK
RATINGS FILE NAME? SAMPSYS
THRESHOLD: PRETTY HIGH
FAIRLY LOW
```

Figure 6.4b Sample Installation - Selected Weakest Link Rating

For each element in the installation, the fuzzy maximum of the object value and threat likelihood is computed. If this result lies above the threshold the element is considered in the weakest link calculation which proceeds as before. In this example we have chosen **pretty high** as

the threshold value. Visual inspection of Figure 6.2 indicates that elements 5, 7, 8, 10, and 11 are to be considered. Since element 10 has the lowest resistance in this subset, we expect a score of **fairly low**. This is indeed the score returned by the rating calculator (Figure 6.4b).

We now consider an alternative scoring function which is more optimistic and possibly more realistic for real-world security system analysis. Instead of returning a rating based upon the worst case, we select the fuzzy mean scoring function which returns an "average" resistance value. As explained in Section 4.2, this is the analog of the classical numeric mean. As in the case of the "weakest link" score, only resistance values are considered but now a "mean" is returned as the score (Figure 6.4c).

```
MEAN
RATINGS FILE NAME? SAMPSYS
SORTOF HIGH
```

Figure 6.4c Sample Installation - Fuzzy Mean Rating

Since the the majority of resistance values in our sample system lie above **medium** we would expect the mean to lie in the high end of the scale. On the other hand, the mean will not lie very far above mid-scale since most of the ratings are hedged **high** values which fall nearer to the middle of the scale than the upper end. For example, three of the resistances are rated **sort-of high** which ranges just above **medium** in the example grammar. The actual score returned (see Figure 6.4c) agrees with our intuition in this case.

Finally, we exhibit the result of scoring with the fuzzy weighted fuzzy mean. For variety, we will evaluate the software subsystem (see Figure 6.1) rather than the entire installation. This subsystem consists of elements 7, 8, 9, 10 and 11 in Figure 6.2; the subsystem rating is shown in Figure 6.4d. Recall that this is a fuzzy mean weighted by the maximum of object value and threat likelihood (see Section 4.4) and is thus the counterpart of the preselected weakest link. We would expect elements 10 and 11 to dominate somewhat because of the **very high** values or likelihoods there. The resistance values for these elements essentially balance each other and so it appears the score will be near mid-scale. The calculated score (see Figure 6.4d) is quite imprecise and displays a wider range of values than any of the previous scores. In particular, the **moreorless medium** component indicates a large degree of fuzziness which is due to the

```
      WMEAN
RATINGS FILE NAME? SAMPSUB
(SLIGHTLY HIGHER ) THAN MOREORLESS MEDIUM
```

Figure 6.4d Sample Installation - Fuzzy Weighted Mean Rating

way in which fuzzy sums, products, and quotients are calculated (see Chapter 4). This increase in fuzziness is in agreement with our intuition since it is reasonable to assume an aggregate of fuzzy values will be less precise than any of its components. It is important to emphasize that this penalty for flexibility of expression must be paid. The fuzzy ratings must be recognized for what they are: gross estimates based upon imprecise knowledge.

The scores presented should give the reader a flavor for the operation of the rating calculator. We have only implemented a few of the possible variants of scoring calculations. In particular, Michelman has implemented scoring of subsystems and composite scores based upon subsystem results [MICHELMAN 1977]. The use of the present rating calculator requires the rater to be familiar with APL and use many or our individual system functions directly. We are currently implementing an improved user interface based upon the work of Michelman and the suggestions of the student raters mentioned above.

7. Summary and Extensions

There is little existing work aimed at formalizing security design guidelines. Turn has proposed one view [TURN 1974] of the computer security environment (Figure 1.1). We have used his work as a starting point for the development of our model (see Section 1.3). Much literature exists concerning the abstraction of protection mechanisms within operating systems and data base management systems, including [HARTSON 1976], [HARRISON 1976], [HSAIO 1974], [LAMPSON 1971], [POPEK 1974], [DENNING 1976]. The various costs of ensuring compliance with privacy regulations has been investigated [GOLDSTEIN 1976]. We know of no attempts to apply the methodology of mathematical modeling via fuzzy set theory to the *measurement* of computer security. With the exception of [HOFFMAN 1974], possible mechanisms for comparative rankings of data processing security systems have not been explored. We believe fuzzy security ratings will provide an important first step in the development of security metrics.

While we are firmly convinced of the need for such security metrics, we are just as certain that, given the state of the art, ratings must be based upon human judgement. There are just too many complex and poorly understood aspects of the security problem. We are skeptical of numeric rankings based upon qualitative (and usually subjective) evaluations by human auditors. Such numeric rankings suggest more precision than reasonably exists and are difficult to interpret meaningfully. One definition of this dilemma is the *principle of incompatibility:* As system complexity increases, analytical precision decreases (paraphrasing [ZADEH 1975b]).

We therefore believe the present limits of security engineering dictate an evaluation methodology which is as exhaustive as is practical in locating system vulnerabilities, but somewhat imprecise in estimating the relative effectiveness of various security techniques and their contribution to the overall security design goal. Accordingly, we have introduced the concept of a *fuzzy rating* as a tool for formalizing the usually subjective and imprecise process of evaluating the security of a data processing installation. We know of no other attempts to build a rating scheme around the linguistic variable although Wenstop has applied the linguistic variable to the problem of simulation in the analysis and evaluation of organizations [WENSTOP 1975].

We presented the basic and rated security system models as abstract vehicles to aid in the organization of the analysis and rating processes. We recognize that these models are simple and therefore perhaps inadequate for exactly representing real-world data processing facilities. Simplicity was required to attain the generality needed to deal with the diverse types of technical, administrative, and physical security techniques in use today. We discuss extensions to the rated security model below.

Through the use of Zadeh's extension principle we have proposed scoring functions for use in inferring the overall security rating of a system based upon the security contributions of its components. The scoring functions introduced were:

Weakest Link
Preselected Weakest Link
Fuzzy Mean
Weighted Fuzzy Mean
Fuzzy Weighted Mean

We wish to emphasize that these scoring functions are not to be considered the final word in calculating composite ratings. Further research will doubtlessly produce more complex methods for the inference of overall security system ratings. We do believe however, that the basis for these improved scoring functions will be linguistic ratings because of their ability to remain meaningful in the face of imprecision and to facilitate the human evaluation process.

7.1 Status of the Security Evaluation Field

Our interpretation of security system evaluation is diagrammed in Figure 7.1 as a "flow chart" for the entire process. Those portions of the evaluation methodology which have been fully or partially developed in this work are enclosed in boxes.

Referring to the upper left of Figure 7.1, the security system is analyzed through the mechanism of the security audit [FARR 1972], [KRAUSS 1972], [WOOLDRIDGE 1973]. A hierarchical organization of the data [MICHELMAN 1977] aids in the decomposition of the system into a form compatible with the Basic Model of Section 1.3 of the present work. Building upon work in measurement scale design [TORGERSON 1958], [THURSTONE 1967], [COOMBS 1970] and the vast amount of research in natural language theory, a rating language must be designed. The language allows flexible expression of judgmental measurements over a "linguistic rating scale" as discussed in Chapter 3. The linguistic variable [ZADEH 1975b] serves as the focal point for the development of such a measurement language. The design of a "standard" or "optimal" (in some yet undefined sense) rating language is a topic for future work.

Armed with the above tools, the security rater must perform the analyses shown in Figure 7.1 to arrive at an evaluated set of components which is exemplified by the Rated System Model (see Section 1.5). In the case of risk analysis we include probabilistic techniques [BALL 1977] as well as subjective estimates of the attractiveness of a particular vulnerability to an intelligent interloper operating under a "protector-intruder" scenario [TURN 1972]. Values, likelihoods, and resistances are all assumed to be relative values on the single rating scale

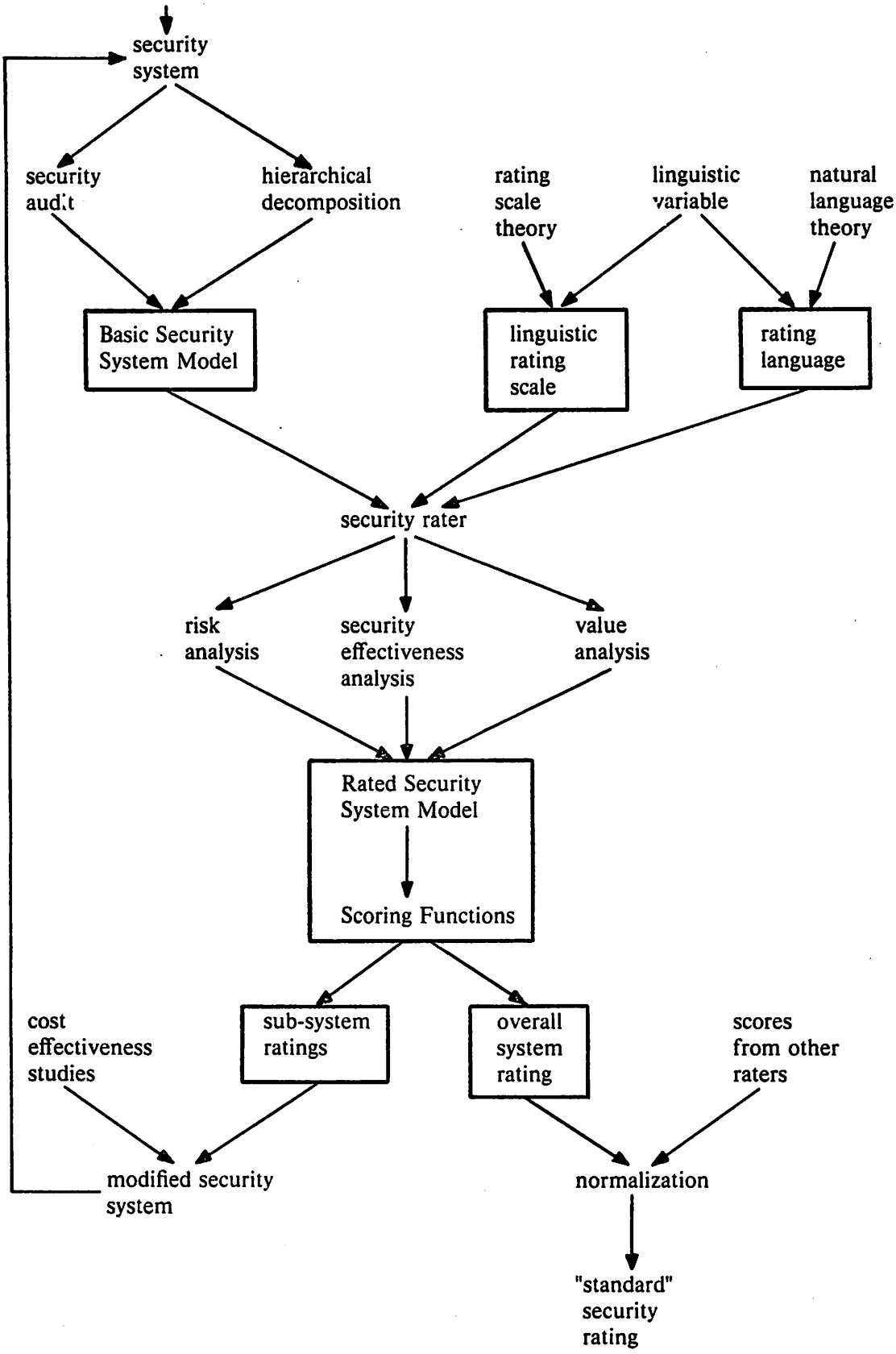

Figure 7.1 The security evaluation process

postulated above. Three separate rating scales and languages would perhaps be more useful but add too much complexity to be within the scope of our present efforts (see, for example, the discussion of Assumption 5 in Appendix A).

One or more of the scoring functions presented in Chapter 4 may then be applied to generate an overall system rating. In the event that standardization techniques are formulated for use with linguistic ratings one final result of all of these efforts will be a "standard" security rating which may be compared against data processing installations of similar type. Of course, alternative scoring functions may also be useful. This is another field for future research. We are currently studying the application of scoring functions to subsystems (defined mainly by function within the installation) as a technique for isolating the weaker security areas [MICHELMAN 1977]. Armed with relative ratings which indicate the areas of security importance and the resistance of these areas, proposed modifications to the existing security system may be studied for cost-effectiveness and an iterative process utilized to raise the security rating of the installation as desired within the limits of cost or other external constraints.

Some of the items in Figure 7.1 represent the on-going efforts of other researchers. We are primarily concerned with the security analysis portions of the methodology, trusting that further advances in linguistics and the psychology of measurement will yield improved rating languages. We also await the development of normalization techniques which will be directly applicable to the linguistic approach to evaluation. The remainder of this chapter will be devoted to the brief exploration of some specific recommendations for extending the present work.

7.2 Improving the Implementation

There are a number of ways in which the implementation described in Chapter 5 and Appendix B might be improved. In this experimental rating system issues of space and time efficiency have been largely ignored. For example, the linguistic approximator concatenates terms from the rating language and then executes the resulting string as a set of *APL* function calls. This results in a large number of re-executions as different combinations of linguistic terms are combined to find a "best-fit" fuzzy set for the final score. A useful space-time trade off may be had by redesigning the approximator to "remember" the fuzzy sets which are the meanings of the various rating phrases as the approximation heuristic proceeds.

There are definite considerations of storage efficiency in connection with the calculation of the fuzzy scoring functions presented in Chapter 4. The arithmetic operators are defined upon the fuzzy Cartesian product of their operands. For example, assume a base scale of the integers 1 to 9. The addition of two fuzzy sets on the base scale will yield a fuzzy set on the interval

2 to 18. For simplicity, the interval 1 to 18 is actually used. Multiplication will map to the interval 1 to 81. Since we implement fuzzy sets as *APL* vectors (see Section 5.1), the application of the scoring functions presents the requirement for large intermediate vectors in the *APL* workspace. The division operation presents no additional demands upon intermediate storage because of the way in which it is implemented. Division maps back to the original base scale but only integer base values are retained by the *APL* function which performs fuzzy division. In general, given a base scale of k elements with n security elements to be rated, the calculation of the fuzzy mean will generate a summation vector of kn values. During the calculation of the fuzzy weighted fuzzy mean, each weighted resistance (recall the weight is also a fuzzy set) is a vector of k^2 values. Summation of these weighted values yields a vector of k^2 elements. For a large number of security elements (as in the analysis of a complex real-world data processing installation) the demands for workspace size become significant.

The *APL* code of the fuzzy various fuzzy arithmetic functions is given in Appendix B. These are loop-driven calculations which are not designed in keeping with the philosophy of *APL* programming [GILMAN 1974]. Alternate methods based upon the *APL* outer product operator could be implemented but it is not clear at this point that any significant gain in efficiency would result. In general, our code has been designed with the goal of maximizing clarity and ease of modification rather than efficiency. This was, we feel, a reasonable design decision given the prototype nature of the Ratings Calculator software.

7.3 Adding Scoring Functions

The scoring functions presented in Chapter 4 are intended to illustrate what might be done and should not be considered the only approach to the generation of a composite security system rating. It may be that a more complex function than the mean or weighted mean may ultimately be more useful in determining the performance of the security system as a whole. Our definition of the weight as the maximum of value and likelihood is probably an oversimplification. It may be that a weighted average of these two parameters would be more satisfactory. It is possible that the makeup of the weight might be varied depending upon the particular security element being rated. Further work in the form of field studies of the use of this method of evaluating security systems is needed to attack these problems.

Recall that a *fuzzy mean* scoring function was introduced in Section 4.2. It would be useful to be able to calculate an analog of classical variance just as the fuzzy mean was defined as an extension of the classical numeric mean. We begin with one formulation of variance:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{7.1}$$

where $\bar{x}$ is the mean. All that is necessary to extend this computation to yield a *fuzzy variance* is to define the operation of fuzzy subtraction. This is easily done by employing the extension principle (see Section 2.7) as was done in Chapter 4 in defining the various fuzzy binary operators used in fuzzy arithmetic. A fuzzy squaring operation is also easily defined. Alternatively, fuzzy multiplication could be used. A minor implementation difficulty results since fuzzy subtraction would necessitate extending the base scale to accommodate zero and negative base values.

A more serious complication arises due to the fact that the result of the variance calculation will not, in general, map back onto the original ratings base scale (e.g. the interval 1 to 9). This implies an increase in the complexity of the linguistic approximator which would have to handle a second language defined over a "variance scale". These problems do not appear to be too difficult but their solution is beyond the scope of the present work. This is an important extension however. The use of the fuzzy mean as a composite rating of a real-world computer security system would require some indication that the variance was **low** to assure the system design was not too unbalanced.

7.4 Better Linguistic Approximation

The existing linguistic approximator employs a rather "brute force" heuristic. While the user is allowed to provide his or her own primary terms, hedges, and relations, the approximator is designed around the syntax described in Section 3.8. A useful extension would be the incorporation of a more general approximation function which would accept the syntax of the rating language as an input. This would facilitate field studies of the rating language which might determine the most "natural" rating language (from the rater's point of view). The linguistic approximator performs essentially two operations in an iterative fashion: 1) - phrase generation and 2)- matching against the composite rating. The proposed extension would make the first operation more general.

Another improvement to the approximator would result from the design of better heuristics for generating phrases which have a high probability of matching the meaning of the composite rating. While the development of such heuristics is beyond the scope of our present efforts, we feel compelled to point out this useful area of research.

7.5 User Interface

There are almost limitless possibilities for extending and improving the interaction of the rating system with the security system rater. As things now stand, the rater has the responsibility for providing correctly written *APL* functions for the generation and manipulation of the fuzzy sets which represent the "meaning" of the linguistic rating phrases. The alternative is to use the default language functions (see Appendix B) in whole or in part. There are no facilities for performing any really useful amount of syntax checking on the linguistic rating phrases which are input by the rater as values of resistance, likelihood, etc. Certain of these syntax errors will be flagged by the *APL* interpreter during execution of the scoring functions. It would be more useful to the rater to be informed of syntax errors when the ratings are entered initially.

It is of course, possible for a rating phrase to be syntactically correct yet be semantically meaningless or, at least, ambiguous. For example, the phrase **not medium** is permissible given the grammar of Section 3.8. The interpretation of this phrase would yield a fuzzy set with high membership values near the base scale end points and low values at the middle of the scale. This is equivalent to saying that the resistance of a security feature (for example) is either **high** or **low** but not **medium**. Of course the tools of Chapter 2 and 3 allow us to formulate such a phrase and there may be other applications where phrases of this sort make sense. However, this kind of linguistic expression is not useful in a *rating system* because it can only be given a contradictory interpretation as an evaluation measure. This is just a restatement of the general problem of maintaining convexity which we have mentioned earlier. A useful extension to the present system would be the testing of each input linguistic value for the convexity property.

7.6 Calibrating the System

Our justification for the form of the fuzzy sets and semantic operations which underlie the rating language (see Chapter 3) relied upon an appeal to the reader's intuition. Such plausibility arguments are necessary in the absence of hard data about the way in which the rater would relate the base scale numbers to the various linguistic values. There are several approaches to the problem of calibration, that is, to the generation and justification of rating semantics.

One possibility would be to expand the user interface of the rating system program to include a software package which would gather information from each rater in some systematic way. This information would relate to the rater's internal conception of concepts such as **high** and **low** as well as ideas such as **very**, **lower**, and so on. In essence, the software builds the fuzzy sets in an empirical fashion by calculating the *linguistic bias* of the rater. A different fuzzy set might be associated with the term **high** for each rater.

Since each rater would have his or her own personal rating language (at least the semantics), it would be useful to investigate methods of *normalizing* ratings so that the scores of a particular installation by different security raters could be compared. A normalization process would aid in comparing different computer security systems which have been rated by different people. This normalization process could also serve to train student raters by comparing their own installation ratings with the standard ratings of security "experts" for the same installation.

A related approach might be the development of a *standard rating scale.* The base scale could be some standard interval such as the nine-point scale used in our implementation, or perhaps an interval on the positive real line. More importantly, the syntax and semantics of the rating language as well as the vocabulary which is used to generate a rating phrase could be standardized. It is our feeling that such a standard would be very difficult to enforce, however. Linguistic bias is a very personal and subjective thing; it would probably take quite a bit of "training" to assure each rater meant the standard fuzzy set when he or she said **high**. A first step in devising a standard for rating semantics would probably be the statistical analysis of data gathered in a manner similar to the operation of the calibration program discussed above. This particular extension would demand much more field research into the way people internalize linguistic values. Since the research would be statistical in nature, this might suggest an avenue for relating the linguistic variable and fuzzy sets to classical probability theory.

7.7 Extending the Model

A major shortcoming of the rated security model of Section 1.5 is that it ignores possible interdependencies of the security elements in an installation. For example, it may be that the password scheme at Installation A has a **very high** resistance to compromise. However, if the system password file is stored in plaintext form (or perhaps using an inadequate encryption algorithm), the actual resistance of the password scheme may be no higher than the resistance of the cryptography process.

One possible method of capturing this element interdependence would be to state the password feature resistance value using a *conditional* rather than simply declarative phrase. Let the password scheme resistance be denoted $\mathbf{R}_1$ and the file cryptography system resistance be denoted $\mathbf{R}_2$. The relationship we wish to model might be stated:

$$\begin{aligned} \text{IF} \quad & \mathbf{R}_2 = \textbf{medium high} \\ \text{THEN} \quad & \mathbf{R}_1 = \textbf{high} \\ \text{ELSE} \quad & \mathbf{R}_1 = \textbf{low} \end{aligned} \tag{7.2}$$

It is possible to model the implication as a binary fuzzy relation [ZADEH 1975a], [ZADEH 1975b] in the following way:

$$\text{IF } A \text{ THEN } B \text{ ELSE } C \;=\; (A \times B) \bigcup (A' \times C) \tag{7.3}$$

where "'" signifies complementation (see Section 2.3), A, B and C are fuzzy sets in a (non-fuzzy) universe of discourse U and " × " denotes the fuzzy Cartesian product:

$$A \times B = \bigcup_{(i,j) \in U \times U} (\mu_A(i) \wedge \mu_B(j))[i,j] \tag{7.4}$$

Thus IF-THEN-ELSE is modeled as a fuzzy *relation* which may then be composed using the methodology discussed in Section 2.5. Let us label this particular relation S. Returning to example (6.2), if

$$S = \textbf{medium to high} \times \textbf{high} \cup \textbf{not (medium to high} \times \textbf{low)} \tag{7.5}$$

then an assignment to $\mathbf{R}_2$ will allow the calculation of $\mathbf{R}_1$ through the compositional rule of inference (see Section 2.6):

$$\mathbf{R}_1 = \mathbf{R}_2 \circ S \tag{7.6}$$

Similarly, interrelationships of threats or other system dependencies could be explicitly modeled in the rating language. The analysis phase of evaluation of the data processing installation would be a good time to generate a set of conditional rating phrases. The rater would at that time make assignments to the element *value, likelihood* and *resistance* variables and the scoring calculation would proceed normally. During the accumulation of the scores by the rating system, the relations which represent the conditional ratings would be evaluated to arrive at the inferred linguistic values.

7.8 Hierarchical Analysis

The model of Section 1.3 treats the computer security system as essentially a single set of triples. Each element of the system is encountered at the same level of analysis as any other. As a result, one would expect the number of triples being enumerated to be quite large in any real-world computer installation. An alternative approach which would reduce the amount of information to be considered at one time has been suggested by Michelman [MICHELMAN 1977]. His approach would model the installation as a *hierarchy* of security subsystems. Each level of the hierarchy would consist of a number of rated security subsystems, where each subsystem was modeled using the rated security system model of Section 1.5.

Thus, at the top level, the installation might be analyzed as three subsystems: hardware, software, and personnel. Each of these subsystems has a security rating which is calculated using the methods suggested in this work. A subsystem rating is inferred by analyzing it using the rated security system model and rating each of its elements. Each element may in turn be a subsystem so that a hierarchy is established.

Such a hierarchical analysis affords the user a great deal of flexibility. The system manager may choose to examine the "important" subsystems in more detail while accepting rough approximations to the security performance of less vital subsystems. The level of detail to be employed at each stage is user-controlled.

Appendix A Major Assumptions

We present here a more detailed discussion of the assumptions listed in Section 1.1 of the thesis. Included are justifications and background material as well as references to existing works which support our plausibility arguments.

Assumption 1

Current methods for evaluating security systems are inadequate.

There does not currently exist a standard methodology which allows one to evaluate the security performance of the entire data processing installation with the goal of producing a "measure" or index of adequacy. An obvious way to approach the development of such an index is to build upon existing techniques. One common contemporary technique is the security audit process. ([FARR 1972], [KRAUSS 1972], [WOOLDRIDGE 1973]). These are usually check list procedures which yield binary results - a particular security mechanism exists at the installation or it doesn't. Sometimes an attempt is made to incorporate knowledge of an estimate of the probability of an intrusion at a particular point in the installations defenses or of the relative effectiveness of a particular security mechanism [FARR 1972].

Our work is concerned with the generalization and formalization of this approach. The ultimate goal is the transformation of the binary audit procedure (Is it present?) into a "measurement" procedure (How adequate is it?). The term "measurement" must be loosely interpreted since we will attempt to attain a very gross qualitative indication of overall system performance. There is no claim that this method is the "best" way to evaluate security systems. The claim is that no best way yet exists although this methodology is at least as adequate as current consultant procedures since it is based upon consultant evaluations. The individual sub-system evaluations are provided by the security "expert" who performs the audit. The resulting rating or index is no better than the inputs.

We are modeling and formalizing the procedures used by a security consultant in producing an evaluation report and we are simplifying the process. We do not claim to be duplicating the consultants thinking process. We are developing a tool to aid him or her by automating much of the bookkeeping required in infering from an examination of the individual system components the overall performance of the security system. Additionally, we are helping the consultant form a more complete analysis since the computerized check list approach based upon the model and developed by Michelman [MICHELMAN 1977] helps avoid oversights in enumeration of security objects, threats and features. As the student responses during our

experiment (see Chapter 7) indicate, items are often overlooked in current risk analysis approaches.

Assumption 2

A rating methodology is a valid way to handle the security evaluation problem.

The question most often asked in real-world computer security is "How secure is XYZ's computer facility really?" It seems to us that a single system score would be a useful way to answer that question. After all, this is the bottom line which the consultant produces and management needs to make security improvement decisions. It is unclear that any current methodology (actually several methodologies are in use) for analyzing complex real-world installations are any more or less adequate than a rating approach. The only reasonable alternative seems to be a case by case examination of the system components without attempting to draw any conclusions about the performance of the facility as a whole. It may be that this is the best that anyone can do but we believe not. At any rate, our methodology also allows one to look at interesting sub-systems once a hierarchy is suitably defined [MICHELMAN 1977]. More importantly, we feel a composite measure, however imprecise, will serve the useful role of drawing the data processing manager's attention to potential weak areas. This simplifies decisions about where to spend money - in security features or elsewhere.

We assume our rating methodology is at least as effective if not superior to existing security analysis methods. This cannot be proven without a substantial amount of empirical testing and therefore must be assumed. Much depends upon the definition of "better". We are not claiming great precision, only that the precision is not there due to the complexity of the system being evaluated and the human elements involved. It is interesting to note that none of our student "experts" (see Chapter 7) found the rating results less justifiable than their personal evaluations of their computer facilities. Some found the ratings to be more reasonable than their original estimates after they had reviewed the results.

We will not attempt to produce a large body of empirical evidence that this evaluation methodology produces completely accurate results. We won't know if the rating approach to security is useful until it is tried out for some time. Our experiment with the student raters (see Chapter 7) cannot give too much support because the sample is so small and they are not true security experts. We must learn more about the way in which raters formulate individual element linguistic values and more about integrating the user into the rating calculation environment. It is not necessarily true that the calculated rating should match the opinion of the security rater about the overall performance of the system. If the calculated rating seems

reasonable or causes the security rater to re-evaluate his or her position on the total system performance, then the methodology and scoring functions are justified.

While some security features may lend themselves to precise quantification, the majority will be evaluated linguistically. The linguistic approach is compatible with and contains the numeric approach as a subset [ZADEH 1965], [ZADEH 1975b]. It is true that the composite rating will be less precise than one in which numbers are used exclusively. We assume the sacrifice of precision for generality is worthwhile. In many cases precise measurements of security effectiveness just do not exist [GLASEMAN 1977].

At any rate, it is not clear that the pure technical features can be evaluated independently of the less objective elements of security systems. For example, the effectiveness of a password against intelligent guessing may depend upon the cleverness with which the password was chosen or its "randomness". Thus an objective criterion such as expected safe time may be inadequate as an estimate of the resistance of the password. In keeping with Assumption 1, we need a measure of overall system security performance. The accuracy implied by objective, physically measurable security features is traded for this ability to form a gross aggregate to allow what boils down to decision (about where to spend the security dollar) in the presence of uncertainty.

At the present time, it is not possible to assert that a particular rating language or scoring function is the only or best. Our goal is to present the basics of rating language design and to study alternative scoring functions. The development of a "standard" or "optimal" rating language is beyond the scope of this work. We make no judgement concerning the scoring functions presented in Chapter 4. "Weakest link" enjoys the advantage of historical use in military security system studies. "Weighted score" enjoys the advantage of allowing the rater to employ information about the relative value of security objects and the likelihood of threats in calculating a system rating. There are probably situations where each would be appropriate depending upon the ultimate use of the rating. For example, weakest link may be the only choice in analyzing a computer or plant security system which relates to nuclear energy. On the other hand, a weighted score may be sufficient in data processing installations where the threats are more in the line of petty embezzlement or misuse of computing resources. In the latter case, a loss may be sustainable if the cost of security is high. Only widespread usage of the rating system in practical situations will determine which language features or scoring functions are most useful or accurate.

The suggestion of the weighted score as an alternative to the purely weakest link approach is an attempt to incorporate the information from the risk analysis aspect of security evaluation into the generation of an "indicator" of the general performance (from a security standpoint) of the entire security system. If industry standards for weights (object value and threat likelihood)

were ever to be established, the weighted score might be useable for comparing systems with bench marks. It is, of course, unknown if such standard weights are obtainable and whether they would be precise numbers, ranges of values with well-known statistical properties, or subjective fuzzy values determined by a panel of industry experts. The weighted score concept has been around a long time outside of the security field. Security people have used it in some cases [FARR 1972]. The weakest link has long been the choice of the Department of Defense and related agencies. Many private security consulting firms use an initial risk analysis followed by weakest link on the "important" elements. Our function for "preselected weakest link" scoring is an attempt to model this hybrid approach. This preselection amounts to risk analysis in the broadest sense of the word.

We don't contend that an individual rater always internalizes one of these scoring functions in estimating the security performance of a system. The rater may or may not, there is too little data to tell. We have no proof that these are valid scoring functions beyond their past usage by security organizations. At this stage our scoring functions are simply suggestions. A scoring function based upon a probabilistic approach may also be useful. It is important to emphasize that this work supports the use of a probabilistic scoring function. We are not excluding that approach but are not advocating it either.

Designing optimal semantics is beyond the scope of this work. We really can't empirically justify any set of semantic functions. This is the province of the social scientist, psychologist, and linguist. We assume that our selections for defaults have a certain intuitive appeal. Some of the student raters (see Chapter 7) have explicitly indicated their intuitive agreement. Where disagreement was encountered, we were able to provide alternative formulations quite easily. Intuition is inadequate for the design of a standard. Accordingly, the user is allowed to provide his or her own compatibility functions. This was one of the reasons we chose to program the rating calculator in *APL*. The English-like syntax of function composition makes it simple to substitute terms and meanings in the rating language. The rating program allows the user to preserve his or her individual compatibility functions (semantics). We allow changes in the basic rating vocabulary although the grammar of the linguistic approximator is fixed (it employs the user's vocabulary with a fixed syntax.)

Although security feature effectiveness may be properly measured with different units depending upon the nature of the feature, it is meaningful to assign relative values so that a score may be calculated. Resistances may have different physical or psychological units. The expected safe time of a password has a different scale than the integrity of a security guard. We assume the security rater has an internal or externally specified "standard of performance" for the ultimate password scheme (for example). He or she compares the existing password mechanism against this standard to arrive at a "normalized" work factor value which is the

resistance of the password. If the rater is able to confidently assign a numerical value (e.g. 7 out of 9) well and good but he or she is not compelled to do so. We suspect he or she will not produce a number in evaluating the work factor due to the presence of the guard. The linguistic rating language with its underlying fuzzy semantics provides the vehicle for specifying less than precise evaluations. Similar mechanisms apply in the formulation of object values and threat likelihoods.

The design of individual language semantic functions for each system rater is beyond the scope of this work. For each security rater, we assume it is possible to determine the compatibility function which matches his or her notion of the meaning of a given rating term with sufficient accuracy for the purpose of "calibrating" the rating calculator for a given rater. A very few studies indicate that methodologies exist and have been employed with limited success. Zadeh has an algorithm based upon the concept of a branching questionnaire but it is unclear that this technique has been proven in practice. The design of a calibration process is beyond the scope of this work (but see Section 7.6). We are more concerned with the results of combining rating terms and the ensuing approximation process. The area of primitive compatibility function construction is a fertile field for future work.

We assume statistical methods for normalizing the computer security system scores from individual raters exist or can be developed. It is not clear that a "standard" rating scale is the best way to attempt to force consistency among raters. Although the training of raters to a single language met with success in the case of our student raters (see Chapter 7) it would be premature to claim that this procedure will work in general. Since individual linguistic bias is likely to be very strong, we may never be able to adequately train individuals to a standard rating scale. This argues against a statistical approach to a certain degree. An alternative use of statistical techniques would be to accept individual languages and normalize scores across languages which have the same numeric base scale (universe of discourse). The development of such normalization procedures is beyond the scope of this work.

The selection of a competent security rater is not considered here. An incompetent rater will produce a score of limited usefulness. Hopefully, the enumeration methodology implied by the model will limit errors of omission. Unrealistic or unsupported scores must still be detected by independent means. A pure numerical rating system would share the same problems in this regard. The fact is that a single subjective evaluation is often used now as the sole measure of the effectiveness of a security system. This is essentially what the security consultant provides. Our rating methodology is intended to provide an *indicator* of system performance. When combined with a suitable "front end" which facilitates the decomposition of the security system under study in a manner consistent with our model, it is also possible to get an indication of the area of the security system which deserves immediate attention to gain the best

improvement in system performance. We assume our tool will be employed by knowledgeable though not necessarily expert security raters.

Assumption 3

It is desirable to evaluate security systems on a component by component basis.

We assume experts can rate system components with more reliability and accuracy than rating the system as a whole. We must not confuse imprecision with incompleteness. Our rating calculator does not attempt to make a set of imprecise values more precise . Nor are we asserting that the final score will be in any way more accurate than any of the individual element scores. We do believe that the enumeration and analysis which our methodology requires of the security rater (especially when Michelman's analysis aid [MICHELMAN 1977] is used as a front end) will make the score which the program calculates more reliable. The rater has less opportunity to overlook elements of the system. There could easily be several hundred elements to consider. It is reasonable to assert that a calculated score is at least as reliable as an overall estimate arrived at in some unknown manner. There is also the possibility that a standard rating language and a standard weighting set would allow a panel of raters to divide the work and assign sub-systems to raters according to individual expertise. However, standardization is beyond the scope of this work.

Assumption 4

Imprecision is a characteristic of complex real-world security systems.

The many sources of imprecision and the high degree of complexity in data processing systems justifies the use of an admittedly imprecise and sometimes subjective technique for gross evaluation of security system performance. We believe an estimate of "degree" of security is called for. There have often been arguments that we must view security as a binary condition and that "secure" means all entities which are used to promote security in the system must be perfectly effective. In other words, each feature's resistance to penetration must be infinite. We don't believe such a goal is reachable. A possible exception is the operating system and such related large scale software sub-systems. We may be able to achieve some form of "total protection" in these sub-systems if correctness proof techniques [LINDEN 1972] are ever made practical. Currently, such techniques are quite a long way from practical application. At any rate, such protection is likely to be quite narrowly defined. Even were we to wait for

these developments, the administrative and other personnel related security problems will remain. In a sense, what we are evaluating is computer system IN-security. We believe a notion of "degrees" of security is needed and that such a degree can be only estimated in many instances. In a less precise vein, conversations with security experts indicate that a "ball park" approach which does not require the rater to justify non-existent precision would be more useful than a "physical reliability" measure since the current state of the art in security cannot produce such measures.

There are at least two sources of imprecision in this problem. First is the absence of reasonable amounts of historical data for the broad range of security violations we know about [BALL 1977], [GLASEMAN 1977]. We know that many existing operating systems are full of trap doors, for example. We know some data processing managers are willing to live with the holes since *in their estimation* the cost of the patches is not justified by the risk of their exploitation. Our goal is a tool which would force the explication (albeit imprecisely) of this estimation process so that management could at least challenge the system rater on the "reasonableness" of his or her loss estimates (object values) and his or her risk estimates (threat likelihood).

Outside of the realm of purely technical features, there is a vast area of security which depends directly or indirectly upon the human element. This administrative environment is the source of a second kind of imprecision which is, we believe, "inherent" in Zadeh's sense [ZADEH 1975b]. Here security feature effectiveness or resistance depends upon human qualities such as integrity, morale, competence, motivation and so on. We are a *very* long way from objective and precise measures of security effectiveness in this area. Yet this is the area which sees a lot of computer crime [PARKER 1976]. We also know that data processing managers and security consultants evaluate the administrative area constantly. We are advocating an evaluation methodology which recognizes the imprecision and subjectivity but demands a degree of consistency and exhaustiveness which doesn't exist now.

We believe the computer scientist cannot ignore the human element in addressing security. This implies a broad notion of what constitutes a computer science problem. We will concern ourselves, then, the environment in which the hardware and software function. The problem is interdisciplinary and the computer scientist must be included since he or she may be the only one with the expertise to analyze the specific security problems which his or her machine generates.

We must assume that people rate in a way which is compatible with the mechanisms of the Fuzzy Set Theory. The establishment of a "fuzzy thinking" hypothesis is beyond the scope of this work. This issue has only recently begun to receive the attention of psychologists and social scientists. One experimenter [KOCHEN 1975] has attempted to empirically verify the existence of fuzzification in humans. He solicited responses concerning agreement with the

statement: "X is much larger than 5" where agreement was expressed by the position of the subject's mark on a continuous scale. Kochen found that about half of his small population were fuzzy estimators and half were more attuned to a threshold approach.

The numeric approach is actually a subset of the linguistic approach since classical set theory is a subset of fuzzy set theory. If one really has a numeric value available it may be used. It is expressible as a fuzzy singleton at that value on the support scale (see Chapter 2). We suspect a pure numeric approach would not be useful for people who are described by Kochen as "estimators" (the half of his sample who measured in a fuzzy way). [KOCHEN 1975] There would be borderline cases where the results of the approximation process would be further off base than the fuzzy case where a multiple valued result is available for finding "best fit". Composite rating terms (such as "**low to medium**", "**not very high**" and "**slightly lower than medium**") might be difficult to define in a straight word-to-number system.

When a rater is asked to assign a value in the presence of imprecision, linguistic values are more useful than numeric values. A 1-1 mapping of linguistic values to numbers is insufficient since this implies the words are as precise as the numbers. We do not wish to hide the numbers but to offer an alternative which does not force reliance upon a precise numeric quantity.

Assumption 5

The classical concept of probability (in particular the random event) is not always appropriate as a model for the mechanism by which security threats occur. Additionally, it is unclear that people formulate concepts such as "**low**" in a probabilistic manner. We are assuming they do not.

In the case of natural disasters or accidents due to human error which lead to security losses, classical probability may be adequate. It is not clear that sufficient historical data exists at this time to apply probabilistic measures however. At any rate, in the case of a malicious intrusion the random variable is clearly inadequate since it cannot capture the effect of the intelligent intruder. A game-playing approach may be more suitable in these cases [TURN 1972], [IVANOV 1975], [GLASEMAN 1977]. The end result of any analysis on the part of the security rater is a notion of how likely the intruder is to select the particular security feature under study as his or her target. In some cases the intruder may be nature. We have used the term "likelihood" to name this estimate in order to minimize confusion with classical probability. Likelihood is often subjective. We assume the rater uses objective historical data when available in formulating his or her likelihood estimate. The rater also draws upon past

experience in analyzing similar systems. For simplicity we have assumed a non-dimensioned base scale of 1 to 9 for likelihood. A scale of 0 to 1 would work as well and yield compatibility with the classical probability scale but might be construed as an unwarranted claim of similarity between these disciplines. This would needlessly complicate our discussion however. The design of more complex scales of measurement is discussed in Chapter 7.

The point is that a threat may be viewed as a random variable in some cases but not all. We expect most cases of interest to be non-random in nature. It might be useful to separate the two main classes of threats into accidental/natural disaster types as opposed to intentional/criminal activities. We could then have different scoring functions for the two types of security sub-systems. It is not clear that different scoring functions are really necessary in this case, however.

In the area of security effectiveness, it is not clear that all security techniques fit the classical probability model. We expect probability would be applicable only in the case of hardware security techniques. It seems inappropriate to attempt to model software bugs or human integrity with a probability distribution. In the area of risk analysis we acknowledge that many threats are probabilistic in nature with the term "likelihood". We have not addressed the area of interacting risks such as the memory bounds register partially failing just as the user program attempts an illegal memory access. These failures might be amenable to a joint probability approach but other security compromises are not probabilistic in the precise sense but are the result of intelligent analysis by an intruder of the point of highest profit (which may not be the point of highest loss to the owner) or of the intruder's point of easiest penetration. The model is general enough to allow these two varieties of security threats to be considered separately. In a hierarchical analysis approach such as Michelman is developing, these two types of threats might naturally partition but it is too early to tell. This does indicate an area for future work.

Linguistic probabilities [ZADEH 1975b] could be chosen as the basis for evaluating threat likelihoods. We have rejected this approach for two reasons. First, the universe of discourse must be the interval [0,1] for linguistic probabilities. Our rating scale of 1 to N (which is dimensionless) allows the use of a single base scale for object value, threat likelihood, and feature resistance. This simplified the design of the prototype rating calculator. An integer valued rating scale (as opposed to the [0,1] interval) allows the simple implementation of fuzzy sets as *APL* vectors. There is no inherent advantage to the [0,1] scale since we are not multiplying in the probabilistic sense. Secondly, linguistic probabilities (as defined by Zadeh) employ a language based upon terms such as "likely", "unlikely", instead of the more neutral adjectives "high", "low" and so on. Thus, at least two rating languages would be required. There is certainly no reason why there couldn't be three rating languages and three base scales. There is no inherent restriction in the model which prevents this. Future work will have to address this

issue since the linguistic values are likely to be dependent upon the concept being measured.

Our rating system is based upon a linguistic rating scale which employs fuzzy sets as the language semantics (see Chapter 3). The translation of the linguistic phrase to a numeric scale allows computation to be performed and composite scores to be generated. These fuzzy set semantics bear a close resemblance to classical probability distributions. This is perhaps unfortunate since it is *not* necessary that statistical or probability methods be employed in the development of the rating language semantics. While there is no inherent restriction in the theory of fuzzy sets or in the rating system design proposed in this work which would preclude the meaning of "**low**" being defined as a probability distribution over the underlying support set, we feel that such an approach may not always reflect the rater's evaluation process.

We are adopting the view of Zadeh [ZADEH 1977] that the *meaning* of the information conveyed by a term such as "**low**" is more properly treated in a possibilistic (to employ Zadeh's terminology) rather than probabilistic framework. The theory of possibility [ZADEH 1977] is analogous to, yet slightly different than, the classical theory of probability. The distinction between a possibility and a probability is a subtle one in practice. Essentially, a possibility statement places some restriction upon the values which some variable may assume. As an example from the realm of computer security, suppose we are attempting to evaluate the resistance of our security guard to errors in identifying authorized users from their security badge photographs. Suppose we are interested in rating our guard on a scale of one to nine. In general, our rating scale is relative and dimensionless (see Chapter 3) but in this case we may think of the scale as representing the number of errors in nine identifications.

Under suitable assumptions about the randomness of identification errors, etc. we might arrive (given sufficient historical data or some other empirical means of probability determination) at a probability distribution such as:

errors	1	2	3	4	5	6	7	8	9
prob.	.1	.8	.1	0	0	0	0	0	0

which indicates we generally expect the guard to make two errors in each nine trys.

Now consider the case in which sufficient historical data is not available to allow a reliable probability distribution calculation. Perhaps our assumption of random errors is not justified. We suspect intuitively that the guards error rate may be much more closely related to job motivation or the amount of sleep he or she had before starting the shift. In the role of security rater, we may observe the guard's performance, study past performance records and gain some estimate of his or her effectiveness. We may then have sufficient information to attempt to *bound* the *possibility* of error in identification of a masquerading intruder. We might decide

that the guard has a low error rate or equivalently (in the context of our model - see Chapter 1) a high resistance to error. The term "**low**" might be represented by the following *possibility distribution:*

errors	1	2	3	4	5	6	7	8	9
poss.	1	1	1	.7	.5	.2	.1	.1	.1

Although our rating of "**low**" conveys much less precision than our probability distribution, there is a certain loose relationship which is intuitively reasonable: an impossible event is certainly improbable but a high possibility does not imply a high probability. Also an improbable event is not necessarily restricted to a low possibility. This relationship is further explored in the references [ZADEH 1977] in the form of the *possibility/probability consistency principle.*

It is important to emphasize that the 0.7 possibility of an error rate of 4 does not mean the guard will mistake 4 out of 9 people 70% of the time. What is indicated is that the rating of "**low**" is not as compatible with an error rate of 4 as with 1,2 or 3. We have restricted somewhat the error rate which we expect to see from this particular security guard. In other words, the value of 0.7 is much less precise than a comparable probability value. It merely indicates a trend.

At this point in time, we cannot prove that humans evaluate via a probabilistic or possibilistic mechanism. A great deal of empirical work must be done to ascertain the internal mechanism by which people operate upon possibilities and probabilities. Such studies are beyond the scope of this work. We use as a given the theory of possibility in this work. Possibility distributions are numerically equivalent to the compatibility functions discussed in Chapter 2. We will adopt the latter terminology.

Why do we choose a rating language instead of just working with numeric ratings? The evaluations by the system rater will in many instances be quite imprecise. In these instances, the rating language allows the rater to express the imprecision succinctly. We are assuming (see Assumption 4) that in at least some of these instances, this imprecision is inherent. In the areas of rating people, physical and administrative security techniques, etc. no one has yet presented a precise metric. We are offering an alternative to the unnatural modeling of human performance as some analog to physical component failure rates.

Assumption 6

A suitable foundation for the construction of a subjective and/or imprecise rating methodology is the Fuzzy Set Theory.

We know of no existing work directly linking Fuzzy Set Theory to the concept of "ratings" or "indices" of adequacy. There is a close parallel in the work of Wenstop however [WENSTOP 1975]. He has studied the analysis of human organizations via simulation. He found the appropriate vehicle for the evaluation of human characteristics such as motivation, satisfaction, etc. was the linguistic variable since it is more natural to speak of "fairly high" motivation or "low" competence rather than a motivation of 6 on a scale of 9. His conclusion was that a number implies too much precision to be useful in such evaluation processes.

In the area of computer security, we feel the human component is ignored too often. Many security features rely upon human integrity or competence for their effectiveness. Some examples are identification via security badges, two-key systems, most administrative controls such as tape sign-out logs, etc. If possible, a rating methodology should allow the use of numbers in situations where they are meaningful. This includes the more purely technical security mechanisms such as passwords, cryptography, and protection features in the operating system. Fuzzy Set theory seems well suited to this goal since it is essentially an extension of classical set theory and thus facilitates the combining of numeric performance measures with non-numeric ratings in a consistent and meaningful way. Mechanisms which are similar in effect to Zadeh's linguistic approach are now in use. The clearest example is the classification of security mechanisms into "high", "low", and "moderate" categories suggested by Farr [FARR 1972].

Only minor extensions to existing Fuzzy Set Theory are needed to allow its application to the security rating problem. We are not primarily concerned with theoretical contributions to the theory of fuzzy sets. We have implemented new linguistic features (e.g "**to**") in Chapter 3 and contributed to the theory of fuzzy arithmetic (see Chapter 4). Our main goal is an *application* of the theory to a real world problem: measurement in the presence of imprecision which is in a sense inherent in the structure due to the human component in data processing installations.

We assume that arithmetic which approximately parallels classical arithmetic is satisfactory for defining the fuzzy mean and fuzzy weighted score. Fuzzy numbers do not have inverses [MIZUMOTO 1976]. Therefore the fuzzy mean is not the precise equivalent of the classical mean. It does reduce to the classical mean when the component values reduce to single numbers. The error seems to be within the range of the linguistic approximator in the practical

situations we have considered. We have seen no bizarre approximations so far.

The specification of any given fuzzy set is quite precise in contrast to the need for imprecision in security ratings. Does "**about** 1" mean .8[0], 1[1], .8[2] or .6[0], 1[1], .7[2] or something else? One can argue that this is quite a constraint upon the concept of "**about**". However, in the manipulation of fuzzy sets (at least in our application) it is the general form of the curve which is important. Once the scoring is completed, the process of linguistic approximation generates the phrase which is the final answer. Small changes in the compatibility function values will not greatly influence the approximator since it works upon the principle of finding a close match between the final fuzzy value and some reasonably simple combination of linguistic phrases in the original rating language. The fuzzy calculation and approximation process is robust and forgiving enough to allow minor variances among users without significantly different end results. We do not have a large body of empirical evidence to back up this claim but close study of the way the scoring functions, linguistic hedges, and relations operate seems to support this position. In other words, there seems to be a high degree of consistency in the linguistic variable approach.

Assumption 7

A simple model is best for dealing with the diversity and complexity of real-world security problems.

Our model (see Chapter 1) is simple and ignores the more complex interactions we would expect to occur in a real-world data processing installation. Security features interact and can not be considered in isolation. Threats are not isolated occurrences but may appear simultaneously and have greater effect in combination than individually. An extreme example is the operating system. There is no mechanism for taking into account the dependence of one security feature upon the effectiveness of another in the system. This is discussed in Chapter 7. There is no explicit mechanism for handling multiple security features on a single object-threat pair. The model would separate the features by generating extra triples. Each would then be on equal footing as far as the scoring was concerned although the weights could be different. This is not really what you want if all of the features for the particular object-threat pair must be overcome to penetrate. In that case the resistances are in some sense additive. We will not investigate this issue.

At any rate, the model is very broad and simple because we wish to include technical, physical, administrative, and legal security techniques. The more complex models in the literature generally deal with protection problems in the classical sense of information flow or access

within the operating system or data base. These models are very important to the development of secure systems but they ignore all of the other areas which studies [PARKER 1976] show contain most of the threats and most of the penetrations in real computer installations. This model is more concerned with all of the various ways in which a computing facility might be attacked. We attempt to encompass the major areas of security by sacrificing depth for breadth. Our goal is not to establish any major theoretical results about the model. We are using it strictly as a vehicle for organizing and explicating the rating methodology.

Appendix B The Rating Calculator Software

Our rating calculator has been written in *APL*PLUS* and runs on the IBM 360/91 at UCLA through the Campus Computing Network. The listings on the following pages are ordered to roughly correspond with the presentation of the fuzzy set primitive operations, the rating language semantics and the scoring functions presented in the main body of this work. These are followed by the *APL* functions which facilitate the generation of individual user language semantics and the gathering of component rating scores. Finally, some utility functions are listed. The functions *AKI*, *AYN* and *NIP* are conversational input functions available in *APL*PLUS* as implemented at UCLA.

Veteran *APL* programmers will note that the functions listed here bear little resemblance to the usual tightly designed functions of *APL*. We have chosen to program in short explicit statements with few operators per statement and few statements per function. There is, of course, a consequent loss of efficiency. We emphasize that the rating calculator is still very much in the prototype stage; we have sacrificed efficiency of expression for clarity whenever we felt it necessary to do so.

```
     ∇ OUT←NORM IN;MAX
[1]   ⍝NORMALIZATION FUNCTION
[2]   ⍝ROW REDUCE TWICE TO HANDLE MATRICES
[3]    OUT←IN ⍝DEFAULT
[4]    MAX←⌈/⌈/IN
[5]    →0 IF MAX=0
[6]    OUT←IN÷MAX
     ∇


     ∇ OUT←CON IN
[1]   ⍝FUNCTION CONCENTRATOR
[2]    OUT←IN*2
     ∇


     ∇ OUT←PWR CONC IN
[1]   ⍝PARAMETERIZED CON FUNCTION
[2]    OUT←IN*PWR
     ∇


     ∇ OUT←DIL IN
[1]   ⍝FUNCTION DILATOR
[2]    OUT←IN*0.5
     ∇


     ∇ OUT←INT IN;LO;HI
[1]   ⍝INTENSIFIER, BROADENS AND STEEPENS
[2]    LO←2×(IN<0.5)×IN*2 ⍝DIMINISH LOWS
[3]    HI←(IN≥0.5)×1-2×(1-IN)*2 ⍝INCREASE HIGHS
[4]    OUT←ROFF LO+HI ⍝COMBINE
     ∇


     ∇ OUT←FUZ IN;LEN;DIFF
[1]   ⍝FUZZIFIER, KERNEL IS GLOBAL
[2]   ⍝IN IS A SCALAR
[3]    LEN←⍴BASE ⍝PAD KERNEL FOR ENDOFF SHIFT
[4]    OUT←(LEN⍴0),KERNEL,LEN⍴0
[5]    DIFF←(KERNEL⍳1)-IN ⍝SHIFT COUNT
[6]    OUT←DIFF⌽OUT ⍝SHIFT ENDOFF
[7]    OUT←LEN↓(-LEN)↓OUT ⍝REMOVE PADDING
     ∇
```

```
     ∇ OUT←REL IN;I
[1]   ⍝PRIMITIVE RELATION
[2]   ⍝SAME IDEA AS HIGHER IN (WENSTOP 1975)
[3]    OUT←IN ⍝FIRST ROW
[4]    I←1
[5]   LOOP:IN←0,¯2↓IN ⍝END OFF RIGHT SHIFT
[6]    IN←IN,1 ⍝RIGHT COL ALL 1'S
[7]    OUT←OUT ATTACH IN ⍝TACK ON NEW ROW
[8]    I←I+1
[9]    →LOOP IF I<⍴IN
     ∇


     ∇ OUT←SFN PARMS;Z;C;P
[1]   ⍝S-FUNCTION (SEE CHAPTER 3)
[2]   ⍝PARMS IS Z,C,P
[3]    Z←1↑PARMS
[4]    P←¯1↑PARMS
[5]    C←1↑1↓PARMS
[6]    OUT←(⍴BASE)⍴0 ⍝INITIALIZE
[7]    OUT←OUT+(BASE≥P)×1 ⍝PEAK
[8]    OUT←OUT+(BASE≤C)×(BASE>Z)×2×((BASE-Z)÷P-Z)*2
[9]    OUT←OUT+(BASE>C)×(BASE<P)×1-2×((BASE-P)÷P-Z)*2
[10]   OUT←ROFF OUT
     ∇


     ∇ OUT←PFN PARMS;B;P
[1]   ⍝PI-FUNCTION
[2]   ⍝PARMS IS B,P
[3]    B←1↑PARMS
[4]    P←1↓PARMS
[5]    OUT←(BASE≤P)×SFN(P-B),(P-B÷2),P
[6]    OUT←OUT+(BASE>P)×1-SFN P,(P+B÷2),P+B
[7]    OUT←ROFF OUT
     ∇


     ∇ OUT←EFN PARMS;PEAK;SPREAD
[1]   ⍝SHAKET'S EXPONENTIAL
[2]    PEAK←1↑PARMS
[3]    SPREAD←1↓PARMS
[4]    OUT←*-((BASE-PEAK)÷SPREAD)*2
     ∇


     ∇ OUT←IN SHF NUM;PK;DIR
[1]   ⍝SHIFTS PEAK MAINTAINING CURVE SHAPE
[2]   ⍝FOR USE WITH EFN
[3]    PK←IN⌷1 ⍝CURRENT PEAK
[4]    DIR←×5-PK ⍝DIRECTION TO SHIFT
[5]    PK←PK+DIR×NUM ⍝NEW PEAK
[6]    OUT←PK EFN BASE
     ∇
```

```
     ∇ OUT←HIGH
[1]    OUT←SFN 5 7 9
     ∇

     ∇ OUT←LOW
[1]    OUT←1-SFN 1 3 5
     ∇

     ∇ OUT←MEDIUM
[1]    OUT←PFN 3 5
     ∇

     ∇ OUT←VERY IN
[1]    OUT←CON IN
     ∇

     ∇ OUT←EXTREMELY IN
[1]    OUT←3 CONC IN
     ∇

     ∇ OUT←MOREORLESS IN
[1]    OUT←DIL IN
     ∇

     ∇ OUT←INDEED IN
[1]    OUT←INT IN
     ∇

     ∇ OUT←ABOUT IN
[1]    OUT←FUZ IN
     ∇

     ∇ OUT←PRETTY IN
[1]    ⍝CLOSEST TO ENDPOINT
[2]    OUT←ROFF NORM DIL(3 CONC IN) AND DIL NOT CON IN
     ∇

     ∇ OUT←FAIRLY IN
[1]    ⍝PEAK SHIFT FURTHER THAN PRETTY
[2]    OUT←ROFF NORM(INT IN) AND NOT INT CON IN
     ∇

     ∇ OUT←SORTOF IN
[1]    ⍝CLOSEST TO MIDDLE
[2]    OUT←ROFF NORM(DIL DIL IN) AND NOT IN
     ∇
```

```
     ∇ OUT←NOT IN
[1]   ⍝COMPLEMENTATION
[2]    OUT←1-IN
[3]    OUT←ROFF OUT
     ∇


     ∇ OUT←LEFT AND RIGHT
[1]   ⍝INTERSECTION FUNCTION
[2]   ⍝SEE (ZADEH 1972)
[3]    OUT←LEFT⌊RIGHT
     ∇


     ∇ OUT←LEFT OR RIGHT
[1]   ⍝UNION
[2]    OUT←LEFT⌈RIGHT
     ∇


     ∇ OUT←X TO Y
[1]   ⍝OR WITH CONVEX FILL
[2]    OUT←X⌈Y
[3]    OUT←1 CONVEX OUT
     ∇


     ∇ OUT←LOWER
[1]    OUT←⌽⍷HIGHER
     ∇


     ∇ OUT←HIGHER
[1]    OUT←REL RELGEN
     ∇


     ∇ OUT←SLIGHTLY IN
[1]    OUT←(INT IN) AND NOT INT VERY IN
[2]    OUT←NORM OUT
     ∇


     ∇ OUT←MUCH IN
[1]    OUT←CON IN
     ∇


     ∇ OUT←LEFT THAN RIGHT
[1]   ⍝RELATION EVALUATOR
[2]    OUT←RIGHT⌈.⌊LEFT
     ∇
```

```
     ∇ OUT←X MIN Y;ROW;I;LIM
[1]   ⍝FUZZY MINIMUM FUNCTION
[2]    OUT←⍳0 ⍝INITIALIZE
[3]    I←0
[4]    LIM←⍴X
[5]   LOOP:I←I+1
[6]    →0 IF I>LIM
[7]    ROW←X[I]⌊Y
[8]    OUT←OUT⌈(I-1)↑ROW ⍝MERGE IN UP TO I
[9]    OUT←OUT,⌈/(I-1)↓ROW ⍝TACK ON MAX OF REST
[10]   →LOOP
     ∇

     ∇ OUT←X MAX Y
[1]   ⍝FUZZY MAXIMUM FUNCTION
[2]   ⍝JUST REVERSE VECTORS
[3]   ⍝AND DO MINIMUM
[4]    X←⌽X
[5]    Y←⌽Y
[6]    OUT←⌽X MIN Y '
     ∇

     ∇ OUT←X PLUS Y;I;LIM
[1]   ⍝FUZZY ADDITION
[2]    OUT←(⍴Y)⍴0
[3]    LIM←⍴X
[4]    I←0
[5]   LOOP:I←I+1
[6]    →SKP IF X[I]=0
[7]    OUT←OUT⌈((I-1)⍴0),X[I]⌊Y
[8]   SKP:→QUIT IF I≥LIM
[9]    OUT←OUT,0 ⍝LENGTHEN FOR NEXT ROW
[10]   →LOOP
[11] QUIT:OUT←0,OUT ⍝INITIAL ZERO TO KEEP SCALE
     ∇

     ∇ OUT←X TIMES Y;I;MSK;LIM
[1]   ⍝FUZZY SET MULTIPLICATION
[2]    OUT←(⍴Y)⍴0 ⍝INITIALIZE
[3]    LIM←⍴X
[4]    I←0
[5]    MSK←1 ⍝SUBMASK
[6]   LOOP:I←I+1
[7]    →SKP IF X[I]=0
[8]    OUT←OUT⌈((⍴OUT)⍴MSK)\X[I]⌊Y
[9]   SKP:→QUIT IF I≥LIM
[10]   OUT←OUT,(⍴Y)⍴0 ⍝LENGTHEN RESULT
[11]   MSK←0,MSK ⍝AND SUBMASK
[12]   →LOOP
[13] QUIT:OUT←1 CONVEX OUT
     ∇
```

```
     ∇ OUT←X DIVBY Y;I;MSK;LEN;L;LIM
[1]   ⍝ FUZZY SET DIVISION
[2]    LEN←⍴X
[3]    L←⍴BASE
[4]    LIM←⍴Y
[5]    OUT←L⍴0 ⍝INITIALIZE
[6]    I←0
[7]    MSK←1 ⍝SUBMASK
[8]   LOOP:I←I+1
[9]    →QUIT IF I>LIM
[10]   →SKP IF Y[I]=0
[11]   OUT←OUT⌈L↑Y[I]⌊(LEN⍴MSK)/X
[12] SKP:MSK←0,MSK ⍝LENGTHEN SUB-MASK
[13]   →LOOP
[14] QUIT:OUT←1 CONVEX ROFF NORM OUT
     ∇


     ∇ OUT←WEAKLINK;LIM;I
[1]    LIM←TIEFILE ⍝SETUP
[2]    OUT←EXEC(FREAD 1,1)[3;] ⍝FIRST RESISTANCE
[3]    I←1
[4]   LOOP:I←I+1
[5]    →QUIT IF I>LIM
[6]    OUT←OUT MIN EXEC(FREAD 1,I)[3;] ⍝ACCUMULATE MINIMUM
[7]    →LOOP
[8]   QUIT:FUNTIE 1
[9]    OUT←DEBLANK SYMTAB LAPPROX OUT
     ∇


     ∇ OUT←SELWEAKLINK;X;THR;LIM;WT;I
[1]    LIM←TIEFILE
[2]    THR←EXEC AKI 'THRESHOLD:'
[3]    I←0
[4]    OUT←⍳0
[5]   LOOP:I←I+1
[6]    →QUIT IF I>LIM
[7]    X←FREAD 1,I ⍝NEXT ELEMENT
[8]    WT←(EXEC X[1;]) MAX EXEC X[2;]
[9]    →LOOP IF WT LT THR ⍝SELECT ELEMENT
[10]   →SKP IF 0<⍴OUT
[11]   OUT←EXEC X[3;] ⍝FIRST RESISTANCE
[12]   →LOOP
[13] SKP:OUT←OUT MIN EXEC X[3;] ⍝ACCUMULATE
[14]   →LOOP
[15] QUIT:FUNTIE 1
[16]   OUT←DEBLANK SYMTAB LAPPROX OUT
     ∇
```

```
    ∇ OUT←MEAN;LIM;MASK;I
[1]   LIM←TIEFILE
[2]   OUT←EXEC(FREAD 1,1)[3;]
[3]   I←1
[4]  LOOP:I←I+1
[5]   →QUIT IF I>LIM
[6]   OUT←(EXEC(FREAD 1,I)[3;]) PLUS OUT ⍝ACCUMULATE
[7]   →LOOP
[8]  QUIT:MASK←(LIM-1)⍴0 ⍝INTERVAL MASK
[9]   MASK←MASK,1
[10]  MASK←(⍴OUT)⍴MASK ⍝REPEAT
[11]  OUT←MASK/OUT ⍝SAMPLE TO 'DIVIDE'
[12]  OUT←DEBLANK SYMTAB LAPPROX OUT
[13]  FUNTIE 1
    ∇


    ∇ OUT←WMEAN;LIM;X;WT;WSUM;I
[1]   LIM←TIEFILE
[2]   I←0
[3]  LOOP:I←I+1
[4]   →QUIT IF I>LIM
[5]   X←FREAD 1,I
[6]   WT←(EXEC X[1;]) MAX EXEC X[2;]
[7]   →SKP IF I>1
[8]   WSUM←WT ⍝FIRST WEIGHT
[9]   OUT←WT TIMES EXEC X[3;] ⍝FIRST RESISTANCE
[10]  →LOOP
[11] SKP:WSUM←WT PLUS WSUM ⍝ACCUMULATE
[12]  OUT←(WT TIMES EXEC X[3;]) PLUS OUT
[13]  →LOOP
[14] QUIT:OUT←OUT DIVBY WSUM
[15]  OUT←DEBLANK SYMTAB LAPPROX OUT
[16]  FUNTIE 1
    ∇


    ∇ OUT←X LT Y
[1]  ⍝RETURN 1 IF X 'LESS THAN' Y
[2]  ⍝LESS IS FUZZY HERE
[3]  ⍝BEST FIT TO MIN IS USED TO MAKE DECISION
[4]   OUT←X MIN Y
[5]   OUT←(X FIT OUT)≤Y FIT OUT
    ∇


    ∇ OUT←EXEC IN
[1]   OUT←ROFF NORM ROFF EXQ IN
[2]  ⍝MAKE A FUNCTION FROM CHARACTER STRING
[3]  ⍝EXECUTE IT VIA EXQ, A MODIFIED
[4]  ⍝VERSION OF XEQ (SEE UCLA APL DOCUMENTATION)
    ∇
```

```
    ∇ OUT←PEAK CONVEX IN;LEFT;RIGHT;MID
[1]  ⍝IN IS FUZZY SET
[2]  ⍝MAKE IT CONVEX VIA INTERPOLATION
[3]   LEFT←IN⍳PEAK ⍝LEFTMOST MAX
[4]   OUT←⌽IN
[5]   RIGHT←OUT⍳PEAK ⍝RIGHTMOST MAX
[6]   LEFT←FILL LEFT↑IN ⍝FILL LEFT SIDE
[7]   RIGHT←⌽FILL RIGHT↑OUT ⍝FILL RIGHT AND REVERSE
[8]   MID←(⍴IN)-(⍴LEFT)+⍴RIGHT ⍝DIST BETWEEN 1'S
[9]   OUT←LEFT,(MID⍴PEAK),RIGHT
    ∇


    ∇ OUT←FILL IN;LEFT;RIGHT;MASK
[1]  ⍝IN IS INCREASING EXCPT AT SOME PTS
[2]  ⍝FILL BY SETTING HOLES TO CURRENT NON-ZERO VALUE
[3]   LEFT←((IN>0)⍳1)-1 ⍝LEADING ZEROS
[4]   →SKP IF LEFT<⍴IN ⍝CHECK FOR NONE
[5]   LEFT←0 ⍝NO LEADING ZEROS
[6]  SKP:OUT←LEFT⍴0 ⍝LEADING ZEROS OUT
[7]  LOOP:IN←LEFT↓IN ⍝START HERE
[8]   LEFT←1↑IN ⍝LOW VALUE
[9]   →QUIT IF 1=⍴IN
[10]  RIGHT←((IN>LEFT)⍳1)-1 ⍝NEXT INCREASE LESS 1
[11]  OUT←OUT,RIGHT⍴LEFT ⍝ADD ON FLAT SEG
[12]  LEFT←RIGHT ⍝NEW LEFT END LESS 1
[13]  →LOOP
[14] QUIT:→0 ⍝SMOOTHED IN LESS RIGHT END 1
[15]  ⍴IN
    ∇


    ∇ GETRATINGS;NEWF;CNO;PART;RATING
[1]  ⍝PROMPTS WITH BARRIER NUMBER
[2]  ⍝WANTS 3 RATINGS IN O-T-F ORDER
[3]  ⍝STORES IN (MAYBE NEW) FILE
[4]   CNO←1
[5]   NEWF←GETFILE 1
[6]  LOOP:→LOOP1 IF NEWF
[7]   CNO←NIP 'C-NUMBER?' ⍝COMPONENT TO REPLACE
[8]   →QUIT IF CNO=0
[9]  LOOP1:'ELEMENT ';CNO
[10]  RATING←AKI 'VALUE?'
[11]  →QUIT IF 0=⍴RATING
[12]  RATING←RATING ATTACH AKI 'LIKELIHOOD?'
[13]  RATING←RATING ATTACH AKI 'RESISTANCE?'
[14]  RATING FUPDATE 1,CNO,NEWF
[15]  ' '
[16]  CNO←CNO+1
[17]  →LOOP
[18] QUIT:FUNTIE 1
    ∇
```

```
    ∇ SEMANTICS;NEWF;CNO;NAME;BODY;FLG;OUT;LINE
[1]  ⍝MAKES CANONICAL FORM (CHAR MAT)
[2]  ⍝OF FUNCTION FROM KEYBOARD OR EXISTING FUNC
[3]  ⍝SAVES ON FILE
[4]   CNO←0
[5]   NEWF←GETFILE 1
[6]   FLG←AYN 'FROM KEYBOARD?'
[7]  LOOP:→LOOP1 IF NEWF
[8]   CNO←NIP 'C-NUMBER?' ⍝COMPONENT TO REPLACE
[9]   →QUIT IF CNO=0
[10] LOOP1:NAME←AKI 'NAME?' ⍝FUNCTION NAME
[11]  →QUIT IF EMPTY NAME
[12]  BODY←NAME ATTACH AKI 'TYPE?'
[13]  →GETIN IF FLG ⍝FUNC FROM KEYBOARD
[14]  BODY←BODY ATTACH 2 ΔFD NAME ⍝USE EXISTING FUNC
[15]  →UPDT
[16] GETIN:'ENTER CODE:'
[17] LOOP2:LINE←AKI '?' ⍝ACCUMULATE LINES
[18]  →UPDT IF EMPTY LINE
[19]  BODY←BODY ATTACH LINE
[20]  →LOOP2
[21] UPDT:BODY FUPDATE 1,CNO,NEWF
[22]  →LOOP
[23] QUIT:FUNTIE 1
    ∇
```

```
    ∇ OUT←LOADSEMANTICS;NAMEF;CNO;FUNC;NAME;ERR;TYPE;TABLE
[1]  ⍝READS FUNCTIONS FROM FILE
[2]   NAMEF←AKI 'FILE NAME?'
[3]   NAMEF FTIE 1
[4]   TTAB←TABLE←⍳0
[5]   CNO←1
[6]  LOOP:FUNC←FREAD 1,CNO
[7]   NAME←,FUNC[1;] ⍝NAME
[8]   NAME←(NAME≠' ')/NAME ⍝BLANKS OUT
[9]   TABLE←TABLE ATTACH NAME ⍝SYMBOL TABLE
[10]  TYPE←,FUNC[2;] ⍝SAME FOR TYPE
[11]  TYPE←(TYPE≠' ')/TYPE
[12]  TTAB←TTAB ATTACH TYPE ⍝SYNTACTIC TYPE TABLE
[13]  FUNC← 2 0 ↓FUNC
[14]  ERR←6 ΔFD NAME ⍝ERASE OLD FUNCTION
[15]  ERR←3 ΔFD FUNC ⍝NEW FUNC
[16] SKIP:NAME,' FORMED'
[17]  CNO←CNO+1
[18]  →LOOP IF CNO< 0 1 0 0 /FSIZE 1 ⍝ENDFILE CHECK
[19] QUIT:FUNTIE 1
[20]  TTAB←TTAB PAD RLEN TABLE ⍝MAKE CONFORMABLE
[21]  TABLE←TABLE PAD RLEN TTAB
[22]  OUT←TABLE,[0.5] TTAB ⍝LAMINATE
    ∇
```

```
    ∇ OUT←TABLE LAPPROX X;HERR;LERR;LIST;BESTLH;LIST1;MERR
[1]  ⍝LINGUISTIC APPROXIMATOR
[2]  ⍝X IS RESULTANT MU VALUE
[3]  ⍝TABLE IS TERM LIST WITH TYPES
[4]  ⍝FIND BEST PRIMARY ABOVE AND BELOW X
[5]  ⍝REPEAT FOR HEDGES
[6]   LERR←⍴BASE ⍝GLOBAL TO FUNC LATRY
[7]   HERR←LERR ⍝LOW AND HIGH ERRORS
[8]   LIST←TABLE GETLIST 'P' ⍝GET PRIMARIES
[9]   BESTLH←0 ⍝INIT
[10]  BESTLH←X LATRY LIST ⍝BEST LOW AND HIGH
[11]  HOLD←BESTLH ⍝TEMP STORAGE
[12]  →QUIT IF EPSILON>HERR⌊LERR
[13]  LIST←TABLE GETLIST 'H' ⍝HEDGE LIST
[14]  LIST1←LIST,HOLD[1;] REPEAT CLEN LIST
[15]  BESTLH←BESTLH PAD RLEN LIST1
[16] ⍝COMPOSITE OF HEDGES AND BEST LOW PRIMARY
[17]  BESTLH←X LATRY LIST1 ⍝HEDGES ON LOW PRIMARY
[18]  LIST1←LIST,HOLD[2;] REPEAT CLEN LIST
[19]  BESTLH←BESTLH PAD RLEN LIST1
[20] ⍝NOW TRY BEST HIGH PRIMARY
[21]  BESTLH←X LATRY LIST1
[22]  MERR←HERR⌊LERR ⍝GLOBAL MINIMUM ERROR
[23]  →QUIT IF EPSILON>MERR
[24] ⍝TRY 'TO' CONNECTIVE
[25]  OUT←'(',BESTLH[1;],') TO (',BESTLH[2;],')'
[26]  →0 IF EPSILON>X FIT ROFF NORM EXQ OUT
[27] ⍝NEXT APPLY RELATIONS
[28]  LIST←,TABLE GETLIST 'L' ⍝LOWER REL
[29]  LIST1←,TABLE GETLIST 'G' ⍝HIGHER
[30]  LIST←LIST,') THAN ',BESTLH[2;]
[31]  LIST1←LIST1,') THAN ',BESTLH[1;]
[32] ⍝TRY 'AND' CONNECTIVE
[33]  OUT←'((',LIST,') AND ((',LIST1,')'
[34]  →0 IF EPSILON>X FIT ROFF NORM EXQ OUT
[35]  OUT←LIST LARHEDG LIST1 ⍝CHECK BEST COMBINATIONS
[36]  →0
[37] QUIT:OUT←BESTLH[1+HERR<LERR;]
    ∇


    ∇ OUT←LEFT FIT RIGHT
[1]  ⍝COMPUTES CLOSENESS OF FIT
[2]  ⍝SUM OF SQUARES OF ERRORS
[3]   OUT←+/(LEFT-RIGHT)*2
    ∇

    ∇ OUT←TABLE GETLIST CODE;MASK
[1]  ⍝RETURNS LIST OF TERMS
[2]  ⍝WHOSE SECOND ENTRY MATCHES CODE
[3]   MASK←CODE=TABLE[2;;1]
[4]   OUT←MASK⌿TABLE[1;;]
    ∇
```

```
    ∇ OUT←X LATRY LIST;LIM;I;NERR;Y
[1]  ⍝MATCH X AGAINST PHRASE LIST
[2]  ⍝CHECK GLOBAL HERR,LERR FOR IMPROVEMENT
[3]  ⍝REPLACE PHRASES AS APPLICABLE
[4]  ⍝THUS BOUNDING ON LOW AND HIGH SIDE
[5]   OUT←BESTLH ⍝GLOBAL CURRENT PAIR
[6]   →PD IF 0≠⍴⍴OUT ⍝IF EMPTY
[7]   OUT←LIST[⍳ 2 ;] ⍝DEFAULT ANY PRIMARIES
[8]  PD:OUT←OUT PAD RLEN LIST
[9]   LIM←CLEN LIST ⍝NUMBER OF PHRASES
[10]  I←0
[11] LOOP:→QUIT IF I≥LIM
[12]  I←I+1
[13]  Y←ROFF NORM EXQ LIST[I;] ⍝EXECUTE RATING
[14]  NERR←X FIT Y ⍝CHECK FIT
[15]  →LOOP IF NERR>HERR⌈LERR ⍝NO HELP
[16]  →SKP IF Y LT X ⍝BRANCH IF RATING LOWER
[17]  →LOOP IF NERR>HERR ⍝NO HELP HIGH SIDE
[18]  HERR←NERR ⍝UPDATE GLOBAL HIGH SIDE ERR
[19]  OUT[2;]←LIST[I;] ⍝NEW TERM
[20]  →LOOP
[21] SKP:→LOOP IF NERR>LERR ⍝NO HELP LOW SIDE
[22]  LERR←NERR
[23]  OUT[1;]←LIST[I;]
[24]  →LOOP
[25] QUIT:→0
    ∇

    ∇ OUT←LWR LARHEDG HGHR;LIST;LIST1
[1]  ⍝FIND BEST COMBO OF HEDGED RELATIONS
[2]   OUT←BESTLH[1+HERR<LERR;] ⍝DEFAULT
[3]   HERR←⍴BASE ⍝NEW GLOBALS FOR LATRY
[4]   LERR←HERR
[5]   LIST←TABLE GETLIST 'R' ⍝RELATION HEDGES
[6]   LIST1←'(',LWR ATTACH LIST,LWR REPEAT CLEN LIST
[7]  ⍝FIND BEST HEDGED LOWER
[8]   LWR←X LATRY LIST1
[9]   LWR←,LWR[1+HERR<LERR;] ⍝BEST ONE OF TWO
[10]  →SKP IF MERR<HERR⌊LERR
[11]  OUT←LWR ⍝CHOOSE HEDGED RELATION
[12]  MERR←HERR⌊LERR
[13] SKP:HERR←⍴BASE ⍝RESET SEMI-GLOBALS
[14]  LERR←HERR
[15]  LIST1←'(',HGHR ATTACH LIST,HGHR REPEAT CLEN LIST
[16] ⍝NOW BEST HEDGED HIGHER
[17]  HGHR←X LATRY LIST1
[18]  HGHR←,HGHR[1+HERR<LERR;]
[19]  →0 IF MERR<HERR⌊LERR
[20]  OUT←HGHR
[21]  MERR←HERR⌊LERR
    ∇
```

```
    ∇ OUT←GETFILE FNO;NAMEF
[1]  ⍝CREATES OR TIES FILE
[2]  ⍝RETURNS 1 IF NEW FILE
[3]   OUT←AYN 'NEW FILE?'
[4]   NAMEF←AKI 'FILE NAME?'
[5]   →TIE IF~OUT
[6]   NAMEF FCREATE FNO
[7]   →0 ⍝EXIT
[8]  TIE:NAMEF FTIE FNO
    ∇
```

```
    ∇ OUT←TIEFILE;NAMEF
[1]   NAMEF←AKI 'RATINGS FILE NAME?'
[2]   NAMEF FTIE 1
[3]   OUT← 0 1 0 0 /FSIZE 1
[4]   OUT←OUT-1 ⍝LAST ELEMENT
    ∇
```

```
    ∇ IN FUPDATE FCFLG;FNO;CNO
[1]  ⍝UPDATES A FILE
[2]  ⍝FCFLG IS FNUM,CNO,NEWF
[3]   FNO← 1 0 0 /FCFLG
[4]   CNO← 0 1 0 /FCFLG
[5]   →APP IF 0 0 1 /FCFLG ⍝NEW FILE, APPEND ONLY
[6]   →APP IF CNO≥ 0 1 0 0 /FSIZE FNO ⍝ADD TO OLD FILE
[7]   IN FREPLACE FNO,CNO
[8]   →0
[9]  APP:IN FAPPEND FNO
    ∇
```

```
    ∇ FDUMP;NAMEF;CNO;LIM
[1]  ⍝FILE DUMPING UTILITY
[2]   NAMEF←AKI 'FILE NAME?'
[3]   NAMEF FTIE 1
[4]   CNO←1
[5]   LIM← 0 1 0 0 /FSIZE 1 ⍝FILE LENGTH+1
[6]  LOOP:→QUIT IF CNO≥LIM
[7]   '     ';CNO ⍝LABEL WITH CNO
[8]   FREAD 1,CNO
[9]   ' '
[10]  CNO←CNO+1
[11]  →LOOP
[12] QUIT:FUNTIE 1
    ∇
```

```
    ∇ OUT←OLD ATTACH NEW
[1]  ⍝ADDS TO TABLE (OLD)
[2]  ⍝IF BOTH OLD AND NEW ARE VECTORS
[3]  ⍝OUT WILL BE A TWO ROW MATRIX
[4]   NEW←MAKMAT NEW ⍝CONVERT VECTOR
[5]   →ADD IF 0<⍴OLD ⍝EMPTY TABLE CHECK
[6]   OUT←NEW ⍝START TABLE
[7]   →0
[8]  ADD:OLD←MAKMAT OLD
[9]   NEW←NEW PAD RLEN OLD ⍝EQUALIZE ROWS
[10]  OLD←OLD PAD RLEN NEW
[11] EQU:OUT←OLD,[1] NEW ⍝CATENATE
    ∇
```

```
    ∇ OUT←MAKMAT IN
[1]  ⍝MAKES MATRIX OF A VECTOR OR SCALAR
[2]   →MATRIX IF 2=⍴⍴IN ⍝LEAVE MATRIX ALONE
[3]   IN←,IN ⍝MAKE A VECTOR
[4]   IN←(1,⍴IN)⍴IN ⍝MAKE A TABLE OF 1 ROW
[5]  MATRIX:OUT←IN
    ∇
```

```
    ∇ OUT←IN PAD LEN;INC;TAIL;SHAPE
[1]  ⍝ADDS BLANK COLUMNS TO IN (IF NEDDED)
[2]   INC←LEN-RLEN IN ⍝CALCULATE WIDTH OF BLANK ARRAY
[3]   →FAIL IF INC≤0 ⍝NO NEGATIVE PADDING ALLOWED
[4]   SHAPE←CLEN IN ⍝COLUMN LENGTH OF INPUT
[5]   →DONE IF SHAPE=0 ⍝INPUT IS VECTOR OR SCALAR
[6]   INC←SHAPE,INC ⍝INPUT IS AN ARRAY
[7]  DONE:TAIL←INC⍴' ' ⍝FORM BLANK ARRAY
[8]   OUT←IN,TAIL ⍝TACK IT ON
[9]   →0 ⍝RETURN
[10] FAIL:OUT←IN ⍝NOOP A NEG. PAD TRY
    ∇
```

```
    ∇ OUT←IN REPEAT NUM
[1]  ⍝MAKE LIST OF NUM ROWS OF IN
[2]  ⍝PREFIX WITH BLANK FOR CONCATENATION
[3]   IN←' ',IN
[4]   OUT←(NUM,⍴IN)⍴IN
    ∇
```

```
    ∇ OUT←CLEN IN
[1]  ⍝MATRIX COLUMN LENGTH
[2]  ⍝OR RETURNS 0 FOR SCALAR OR VECTOR
[3]   OUT←⍴IN ⍝GET SHAPE
[4]   →NOMAT IF 2>⍴OUT ⍝CHECK RANK
[5]   OUT←1↑OUT ⍝GET FIRST COORDINATE
[6]   →0
[7]  NOMAT:OUT←0 ⍝0 IF VECTOR OR SCALAR
    ∇


    ∇ OUT←RLEN IN
[1]  ⍝GETS ROW LENGTH (1 FOR SCALAR)
[2]   OUT←¯1↑,⍴IN ⍝GET LAST COORDINATE OF SHAPE
[3]   OUT←OUT+OUT=0 ⍝RETURN 1 FOR ZERO
    ∇


    ∇ OUT←LABEL IF IN
[1]  ⍝AN APL IF STATEMENT
[2]  ⍝IN SHOULD BE A SCALAR 0 OR 1 (>1 WILL WORK AS 1)
[3]  ⍝IF IN IS A VECTOR, DOMAIN MUST BE {0,1}
[4]  ⍝OUT IS ⍳0 IF FALSE (0)
[5]   OUT←LABEL×⍳0≠∧/IN
    ∇


    ∇ FLAG←EMPTY VEC
[1]  ⍝RETURNS 1 IF VEC IS NULL VECTOR
[2]   FLAG←0=⍴VEC
    ∇


    ∇ OUT←ROFF IN
[1]  ⍝ROUNDS TO PLACES DECIMAL PLACES
[2]  ⍝NEEDS TO HANDLE+NUMBERS ONLY
[3]   OUT←IN×10*PLACES ⍝SHIFT LEFT
[4]   OUT←⌊OUT+0.5 ⍝ROUND UP AND TRUNC
[5]   OUT←OUT×10*-PLACES ⍝SHIFT RIGHT
    ∇


    ∇ OUT←DEBLANK IN;MASK
[1]  ⍝REMOVE EXTRA BLANKS
[2]   IN←,IN ⍝MAKE VECTOR
[3]   MASK←(IN=' ')
[4]   MASK←¯1⌽MASK<1⌽MASK ⍝ONE OF EACH SERIES
[5]   OUT←(IN≠' ') ⍝NON-BLANKS
[6]   OUT←(MASK∨OUT)/IN
    ∇
```

Bibliography

Annotations are included with those references which are especially useful and/or relevant to the present work.

ANDREWS 1975 — Andrews, Gregory R., "Partitions and Principles for Secure Operating Systems," *Proceedings of the 1975 ACM National Conference,* pp. 177-180.

BALL 1977 — Ball, Leslie D. and Hora, Stephen C., "Computer Security and Privacy Threats: A Bayesian Approach to the Estimation of Their Risk and Cost," *Proceedings 1977 Trends and Applications Symposium,* IEEE Computer Society, (May 19, 1977), pp. 14-18.

The authors' approach could perhaps be employed to arrive at our "likelihoods" under the assumption of random threats and our "values" under the assumption that losses can be statistically predicted. One difficulty lies in estimating prior probabilities; the authors suggest subjective estimates (which supports an assumption of this work).

BELL 1973 — Bell, D. and LaPadula, L. J., "Secure Computer Systems: A Mathematical Model," MITRE Corporation, Bedford, MA., MTR-2547, vol. II, Nov. 1973, ESD-TR-73-278.

BELLMAN 1973 — Bellman, R. and Giertz, M., "On the Analytic Formalism of a Theory of Fuzzy Sets," *Information Sciences,* vol. 5 (1973), pp. 149-156.

The authors suggest min and max are the "only reasonable" operators for implementing intersection and union of fuzzy sets. This paper lends intuitive insight to the use of these fundamental operators.

BERKELEY 1974 — Ordinance Number 4732-N.S., "Social Impact Statement for Automated Record Keeping Systems," Berkeley California City Council, 1974.

BROWNE 1973 — Browne, P., "Taxonomy of Security and Integrity," in (HOFFMAN 1973).

A very broad and complete checklist without being voluminous. A good starting point for security analysis.

CBEMA 1975 — Computer and Business Equipment Manufacturers Association, Periodical Lists of State Legislation on Privacy and Security.

CLEMENTS 1974 — Clements, D., and Hoffman, L.J., "Computer Assisted Security System Design," ERL Memo M-468, Electronics Research Laboratory, University of California, Berkeley, Nov. 1974.

A prototype for a computer aid to security design. The central idea is to provide a "menu" of risks and security countermeasures for the designer.

COOMBS 1970 — Coombs, C.H., Dawes, R.M. and Tversky, A., *Mathematical Psychology,* Prentice-Hall, 1970.

DENNING 1976 — Denning, D. E., "A Lattice Model of Secure Information Flow," *Communications of the ACM,* vol. 19, no. 5 (May 1976), pp. 236-243.

Perhaps the most general and complete protection model to date.

FARR 1972 — Farr, M., Chadwick, B., and Wong, K., *Security for Computer Systems,* National Computing Center Ltd., Manchester, England, 1972, 172pp.

A checklist and security audit approach. The authors provide menu matrices of security features classified in terms of "high", "low" and "medium" effectiveness.

GILMAN 1974 — Gilman L. and Rose A.J., *APL - An Interactive Approach,* John Wiley, NY, 1974.

A very detailed introduction designed for use at the terminal. There are many well-explained examples of all standard features as well as a short overview of file systems and formatted output.

GLASEMAN 1977 — Glaseman, S., Turn, R. and Gaines, R.S., "Problem Areas in Computer Security Assessment," *Proceedings of the 1977 National Computer Conference,* pp. 109-112.

GOLDSTEIN 1975 — Goldstein, R.C., *The Cost of Privacy,* Honeywell Information Systems, 40 Guest Street, Brighton, MA., 02135.

GOLDSTEIN 1976 — _____, Seward, H.H., and Nolan, R.L., "A Methodology for Evaluating Alternative Technical and Information Management Approaches to Privacy Requirements," NBS Technical Note 906, National Bureau of Standards, June 1976.

A questionnaire based methodology for evaluating the costs of complying with privacy regulations in computer systems.

GRAHAM 1972 — Graham, G.S. and Denning, P.J., "Protection Principles and Practice," *Proceedings 1972 Spring Joint Computer Conference,* May 1972, pp. 417-429.

GREY 1973 — Grey, L.D., *A Course in APL/360 with Applications,* Addison-Wesley, Reading, Mass., 1973.

A fairly useful presentation of basic APL with many scientific application examples.

HARRISON 1976 — Harrison, M. A., Ruzzo, W. L., and Ullman, J. D., "Protection in Operating Systems," *Communications of the ACM,* vol. 19, no. 8 (August 1976), pp. 461-471.

HARTSON 1976 — Hartson, J. R. and Hsiao, D. K., "A Semantic Model for Database Protection Languages," *Proceedings Second Very Large Data Base Conference,* Brussels, Belgium, 1976.

HOFFMAN 1973 — Hoffman, L.J., *Security and Privacy in Computer Systems,* Melville Publishing Company, Los Angeles, 1973.

A collection of the most important papers on computer security and privacy up to 1973.

HOFFMAN 1974 — _____, "Constructing Security Ratings for Computer Systems," *Proceedings of the 1974 IEEE National Telecommunications Conference,* San Diego, CA.

First attempt to apply linear weight and score technique to the rating of computer security systems.

HOFFMAN 1977 — _____, *Modern Methods for Computer Security and Privacy,* Prentice-Hall, Englewood Cliffs, NJ.

HR 1984 — House Resolution 1984, "Comprehensive Right to Privacy Act," A Bill in the 94th U.S. Congress, First Session, January 23, 1975.

Also appears as Appendix B in [HOFFMAN 1977].

HSIAO 1974 — Hsiao, D.K., Kerr, D.S. and Nee, C.J., "Context Protection and Consistent Control in Data Base Systems (Part I)," Ohio State University, Computer and Information Science Research Center, Report OSU-CISRC-TR-73-9, Columbus, Ohio, 1974.

IVANOV 1975 — Ivanov, K., "Privacy and the Management of (Data) Security," IBM Sweden, 1975.

KAUFMANN 1975 — Kaufmann, A., *Introduction to the Theory of Fuzzy Subsets - Volume 1,* Academic Press, NY.

An excellent text on fuzzy set theory providing a detailed treatment with many intuitive examples.

KOCHEN 1975 — Kochen, M., "Applications of Fuzzy Sets in Psychology," in *Fuzzy Sets and thier Applications to Cognitive and Decision Processes,* Zadeh, L.A., Fu, K.S., Tanaka, K. and Shimura, M. (eds.), Academic Press, Inc (1975), pp. 395-408., New York.

An experiment to determine if there are fuzzy estimators. Unfortunately, the sample was small and the results somewhat inconclusive.

KRAUSS 1972 Krauss, L.I. *SAFE: Security Audit and Field Evaluation for Computer* Facilities and Information Systems, Firebrand, Krauss and Co., P.O. Box 165, East Brunswick, N.J. 08816, 1972.

A **very** *extensive and detailed checklist intended for the professional security audit team. This gives the reader a feel for the magnitude of the security analysis problem.*

LAKOFF 1973 Lakoff, G., "Hedges: A Study in Meaning Criteria and the Logic of Fuzzy Concepts," *Journal of Philosophical Logic,* vol. 2, no. 4 (October 1973), pp. 458-508.

This paper contains very intuitive arguments for the formulation of linguistic hedges using composition of the basic fuzzy set operations.

LAMPSON 1971 Lampson, B.W., "Protection," *Fifth Annual Princeton Symposium on Information Sciences and Systems,* March 25-26, 1971, pp. 437-443. Reprinted in *Operating Systems Review,* vol. 8, no. 1 (January 1974), pp. 18-24.

LINDEN 1972 Linden, T.A., "A Summary of Progress Toward Proving Program Correctness," *1972 Fall Joint Computer Conference,* vol. 41, pp. 201-211.

MICHELMAN 1977 Michelman, E. and Hoffman, L.J., "SECURATE: A Security Evaluation and Analysis System," Memorandum No. UCB/ERL M77/36, Electronics Research Laboratory, University of California, Berkeley, June 1977.

This work extends the checklist audit approach to security analysis through a hierarchical model. The system described serves as a user interface and "front end" for our rating calculator.

MIZUMOTO 1976 Mizumoto, M. and Tanaka, K., "Algebraic Properties of Fuzzy Numbers," *Proceedings of the International Conference on Cybernetics and Society,* Washington, D.C., (1976), pp. 559,564.

Several results concerning fuzzy arithmetic are presented. In particular, our work relies upon the convexity preservation properties proven here.

NASIS 1974 National Association for State Information Systems, Suggested Guidelines for a State Information Practices Act.

NAUR 1963 Naur, P. (ed.), "Revised Report on the Algortihmic Language ALGOL 60," *Communications of the ACM,* vol. 6, no. 1 (January 1963), pp. 1-17.

PARKER 1976 Parker, Donn B., *Crime by Computer,* Charles Scribner's Sons, New York, 1976.

Case histories of various types of computer-related abuse.

PL 93-579 Public Law 93-579, "Privacy Act of 1974," 93rd Congress, December 31, 1974.

Also appears as Appendix A in [HOFFMAN 1977].

POPEK 1974 Popek, G. J., and Kline, C. S., "Verifiable Secure Operating System Software," *Proceedings 1974 National Computer Conference.*

SAFE 1974 "What Every Executive Should Know About Privacy in Information Systems," Project SAFE, State of Illinois, 1974.

SALTZER 1974 Saltzer, J. H., "Ongoing Research and Development on Information Protection," *ACM Operating Systems Review,* vol. 8 no. 3, July 1974.

SHAKET 1975 Shaket, E., "Fuzzy Semantics for a Natural Like Language Defined over a World of Blocks," Master of Science Thesis, University of California, Los Angeles, 1975.

The author proposes the use of exponential curves as the semantics of the linguistic values. This provides a useful alternative to the curves suggested by Zadeh.

STS 1974 *A Users Guide to Enhancements in the APL*PLUS System,* Scientific Time Sharing Corporation, Bethesda, Maryland, December 1974.

*Covers the APL*PLUS file subsystem and many other special system functions which extend the standard APL language.*

THURSTONE 1967 Thurstone, L.L. "Attitudes can be Measured," *Readings in Attitude Theory and Measurement,* M. Fishbein ed., John Wiley, NY, 1967.

TORGERSON 1958 Torgerson, W.S., *Theory and Methods of Scaling,* John Wiley, NY, 1958.

TURN 1972 Turn, R. and Shapiro, N., "Privacy and Security in Databank Systems: Measures of Effectiveness, Costs and Protector-Intruder Interactions," RAND Corporation, Memo P-4871, July 1972.

This is the classic protector-intruder model which offers an alternative to pure probabilistic risk analysis.

TURN 1974 Turn, R., Memo P-5142, Rand Corporation, Santa Monica, California, January 1974.

Qualitative discussion of proposed field of Data Security Engineering. His view of the security environment forms the basis for the Basic Model in Chapter 1 of this work.

US 1974 U.S. Congress, Public Law 93-579.

WALTER 1974 Walter, K.G., Ogden, W.F., Rounds, W.C., et al., "Primitive Models for Computer Security," EDS-TR-74-117, Case Western Reserve University, Cleveland, Ohio, January 23, 1974.

WEISSMAN 1969 Weissman, C., "Security Controls in the Adept-50 Time Sharing System," *Proceedings 1969 Fall Joint Computer Conference,* pp. 119 ff.

This paper is also reprinted in [Hoffman 1973].

WENSTOP 1975 Wenstop, F.E., "Application of Linguistic Variables in the Analysis of Organizations," Ph.D. Thesis, School of Business Administration, U.C. Berkeley, July, 1975.

An application of linguistic variables to simulation of interactions in management organizations. His development of a simulation measurement language has greatly influenced the design of the rating language in this work.

WENSTOP 1976 ——, "Deductive Verbal Models of Organization," *International Journal of Man-Machine Studies,* vol. 8 (1976), pp. 293-311.

A more recent version of [WENSTOP 1975].

WIEDMANN 1974 Wiedmann, C., *Handbook of APL Programming,* Petrocelli Books, NY, 1974.

This is a terse and well organized reference manual. There are many good exercises and demonstrations of the behavior of the primitive functions in limiting cases.

WOOLDRIDGE 1973 Wooldridge, S., Corder, C., and Johnson, C., *Security Standards for Data Processing,* Halsted Press, New York, 1973.

A good basic reference on auditing and effectiveness measures for security systems.

ZADEH 1965 Zadeh, L.A. "Fuzzy Sets," *Information and Control,* vol. 8 (1965), pp. 338-353.

The fundamental paper on fuzzy set theory.

ZADEH 1971 ——, "Similarity Relations and Fuzzy Orderings," *Information Sciences,* vol. 3 (1971), pp. 177-200.

ZADEH 1972 ——, "A Fuzzy Set Theoretic Interpretation of Linguistic Hedges," *Journal Cybernetics,* 2:3 (1972), pp. 4-34.

Discusses the various forms of linguistic modifiers. Much of the development of Chapters 2 and 3 draws upon this work.

ZADEH 1973 ——, "Outline of a new Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 3 (January 1973), pp. 28-44.

ZADEH 1975a ——, "Calculus of Fuzzy Restrictions," in *Fuzzy Sets and Their Application to Cognitive and Decision Processes,* Zadeh, L.A., Fu, K.S., Tanaka, K. and Shimura, M. (eds.), Academic Press, New York (1975), pp. 1-39.

ZADEH 0975b ——, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," Part I, *Information Science,* vol. 8 (1975), pp. 199-249; Part II, *Inf. Sci.*, 8 (1975), pp. 301-357; Part III, *Inf. Sci.*, 9 (1975), pp. 43-80.

The most complete work on the principles and application of the linguistic variable. Includes the extension principle and principle of compositional inference.

ZADEH 1976 ——, "A Fuzzy-Algorithmic Approach to the Definition of Complex or Imprecise Concepts," *International Journal of Man-Machine Studies,* vol. 8 (1976), pp. 249-291.