Incorporating Uncertainty into the Formal Development of the Fusion Operator

Jingsong Li
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, U.S.A.

Mieczyslaw M. Kokar
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, U.S.A.

Jerzy Weyman
Department of Mathematics
Northeastern University
Boston, MA 02115, U.S.A.

Abstract *This paper uses a formal approach to incorporating uncertainty of input information into the fusion process and decision making. Fuzzy set theory (fuzzy numbers, and fuzzy operators) is used to characterize and then manipulate (reason about) uncertainty. A library of specifications of fuzzy set theory is developed using category theory and Specware, a tool that supports category theory based algebraic specification of software. The library is then used to construct specifications of fuzzy information processing systems. The main construction in this process is composition. Category theory operators of limits and colimits are used for composition. As an example, a fuzzy edge detection algorithm is shown, which uses fuzzy operations in its processing. One of the advantages of this approach is that every aspect of the fusion process is specified formally, which allows us to reason about the uncertainty associated with the sensors and the processing.*

Keywords: fuzzy set, category theory, colimit

1 Introduction

In information fusion systems, uncertainty of information comes into the picture for a number of reasons: incompleteness of the coverage of the environment, inaccuracy of the sensors (e.g., limited resolution of sensors), background noise in the environment, and others. There are many ways of dealing with uncertainty. Statistical methods and efficient filtering algorithms have been applied to this area using mathematical tools, such as FFT or wavelets, but none in a completely formal way, i.e., these mathematical formalisms have been used to derive algorithms by humans, but not by computing machines (computers).

Why is a formal method so important? We know that in order to design a fusion system, we need to be able to reason about the impact of the uncertainty of the input information on the outcome of the fusion system, before the system is built. In other words, we need to be able to predict the performance of the fusion system for any given level of uncertainty and guarantee that it will give satisfactory solutions provided that the uncertainty of incoming information is within some prespecified bounds. With conventional methods, reasoning about the performance of the system cannot be done automatically, but even humans might draw different conclusions about a specific system due to the lack of full mathematical specification of the system.

In this paper, we describe the process by which uncertainty is formally incorporated into the fusion system design, so that it allows us to reason about the uncertainty of the decisions of the fusion system while in the design phase. Section 2 describes how a fuzzy set theory library is built using category theory and Specware, and how the library is used to construct specifications of fuzzy information processing systems. This is the main part of the paper. Section 3 describes a simple conventional edge detection algorithm, and then maps this algorithm into a corresponding fuzzy edge detection algorithm in which all the operations are replaced by fuzzy operations. This part serves as an example of the application of our approach to reasoning about the uncertainty in information fusion. Section 4 concludes the paper and gives directions for future research.

2 Fuzzy Information Processing

Before fuzzy set theory was introduced by Zadeh in 1965, uncertainty was solely treated by probability theory. But there are some situations where uncertainty is non-probabilistic. In information processing systems, for instance, we cannot guarantee that the input data are precise numbers; instead they are often referred to as *approximately* x, or *around* x. The reason for this uncertainty is not that we measure the values with some error, but simply because we do not know what it should be. This uncertainty of imprecision can be modeled by using fuzzy set theory. Another example is evident in linguistic expressions, such as *tall, big, hot,* or *likely, unlikely,* etc. This linguistic uncertainty, of *vagueness* or *fuzziness*, can be well described by appropriate fuzzy sets.

In this paper we use fuzzy set theory to handle uncertainty in information processing systems. We show how fuzzy information processing systems can be specified by using category theory and Specware. Category theory is a mathematical technique that is suitable for representing relations between various types of objects [5]. Specifically, we are interested in relations between (algebraic) specifications. Specware is a tool that supports category theory based algebraic specifications of software [10]. This section will talk about the construction of a fuzzy set theory library and fuzzy information processing specifications.

2.1 Construction of Fuzzy Set Theory Library

The fuzzy set theory library is composed of specifications (also called *specs*) of the main concepts of fuzzy set theory: *fuzzy sets, fuzzy numbers,* α*-cuts*, and *fuzzy arithmetic operations.* These specs are useful in composing formal specifications of fuzzy information processing systems.

2.1.1 Fuzzy Sets

There are a number of definitions for fuzzy sets. Two most popularly used definitions are listed here for comparison, out of which we chose the second one.

Definition 1 [4]: Fuzzy set A is a set of ordered pairs

$$A = \{(x, \mu_A(x)) | x \in X\}$$

where X is a collection of objects (called *universe of discourse*), and $\mu_A(x)$ is the *membership function.* This function takes real values between 0 and 1.

Definition 2 [3]: Fuzzy set A is a function

$$A : X \to [0,1],$$

where X is the universe of discourse.

The difference between the two definitions is that the former defines a function that is not necessarily total on X, while the latter requires that the function be total. Since Specware requires that all functions be total, we chose the second definition of fuzzy set for building specifications. The diagram of the specification of fuzzy set is shown in Figure 1.

The spec UNI-INTVL *imports* REAL and introduces a new *sort*: Uni_intvl = $Real$ | $between\_zero\_one?$. FUZZY-SET is a *definitional extension* [5] of the *colimit* of UNI-INTVL and SET; it defines a *function sort*: $Fuzzy\_set = E \to Uni\_intvl$, where E is the

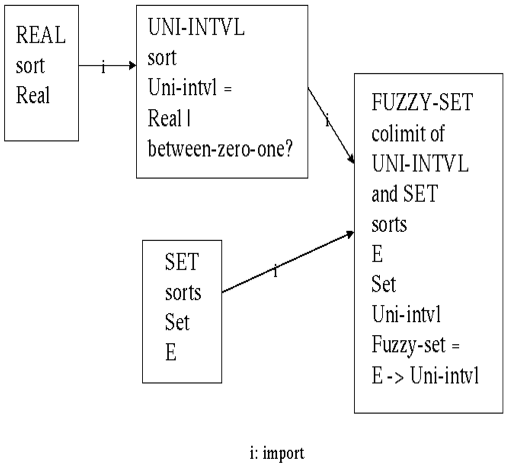

Figure 1: Diagram for Fuzzy-set

type of all elements in Set. In the FUZZY-SET spec, α-cut and height are defined as

$$op\ alpha\_cut : Fuzzy\_set, Uni\_intvl \rightarrow Set$$
$$op\ height : Fuzzy\_set \rightarrow Uni\_intvl$$

The α-cut is a powerful concepts that links fuzzy sets with sets. The application of the α-cut to a fuzzy set results in a set, and thus all operations and relations of sets can be applied to the α-cuts of the fuzzy set, or to α-levels.

2.1.2 Fuzzy Numbers

Fuzzy numbers are one specific type of fuzzy set. The universe od discourse for fuzzy numbers is real numbers. Fuzzy number A has the form: $A : Real \rightarrow [0, 1]$. It has the following properties:

- A must be a normal fuzzy set. That is, the height of the fuzzy set A should be 1:

  $$height(A) = \sup_{x \in X} A(x) = 1$$

- A must be a convex fuzzy set. The property of convexity is captured by the following theorem:

  Theorem: A fuzzy set A on $Real$ is convex iff

  $$A(\lambda x_1 + (1 - \lambda)x_2) \geq min[A(x_1), A(x_2)]$$

  for all $x_1, x_2 \in Real$ and all $\lambda \in [1, 0]$, where min denotes the minimum operator.

- α-cut of the fuzzy set A should be a closed interval for every $\alpha \in (0, 1]$.

These properties are intuitively obvious. A fuzzy number is normal since our concept of a fuzzy number "approximately x" means that it is fully satisfied by x itself. We require that the shape of the fuzzy number be monotonicly increasing on the left and monotonicly decreasing on the right, so α-cuts of any fuzzy number should be closed intervals, which leads to the property that fuzzy numbers are convex.

Fuzzy number is specified in the spec FUZZY-NUMBER, which *imports* FUZZY-SET and adds one *sort axiom*: $E = Real$. It also adds two axioms: *normality* and *convexity*.

2.1.3 Fuzzy Operations

In [3], two methods have been presented for developing fuzzy arithmetic. One method is based on interval arithmetic. Let A, B denote two fuzzy numbers, * denote any of the four basic arithmetic operations, $+, -, \times, and \div$. Then $A * B$ is a fuzzy number, which can be represented by

$$A * B = \bigcup_{\alpha \in [0,1]} ({}^{\alpha}A *^{\alpha} B) \times \alpha$$

This method requires using α-cuts of fuzzy numbers. The second method represents fuzzy number A*B in the following way:

$$(A * B)(z) = \sup_{z = x * y} min[A(x), B(y)]$$

for all $z \in Real$. We chose the latter one because it is more explicitly expressed, thus more convenient to be specified in Specware.

Fuzzy arithmetic operations are specified in the spec FUZZY-ARITHM, which is a *definitional extension* of FUZZY-NUMBER, with fuzzy operations being of the following types.

$$\begin{aligned} op\ f\_add &: Fuzzy\_number, Fuzzy\_number \\ &\rightarrow Fuzzy\_number \\ op\ f\_sub &: Fuzzy\_number, Fuzzy\_number \\ &\rightarrow Fuzzy\_number \\ op\ f\_mult &: Fuzzy\_number, Fuzzy\_number \\ &\rightarrow Fuzzy\_number \\ op\ f\_div &: Fuzzy\_number, Fuzzy\_number \\ &\rightarrow Fuzzy\_number \end{aligned}$$

2.2 Fuzzy Information Processing

There are three stages in fuzzy information processing: fuzzification, fuzzy reasoning, and defuzzification. They are covered in the following three subsections.

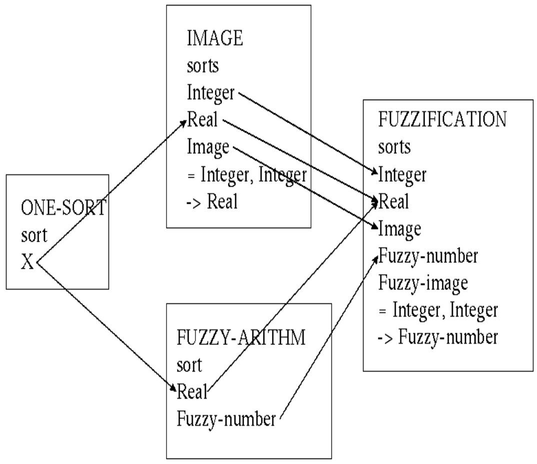

Figure 2: Diagram for Fuzzification

2.2.1 Fuzzification

The first step in fuzzy information processing is to fuzzify input data. There are many ways to do this. We chose the one in which a triangular membership function is involved. For a given value c, we define the triangular fuzzy number A, such that for all $x \in Real$, $A(x)$ satisfies the equation

$$A(x) = \begin{cases} 0 & \text{if } x < c - \delta, \\ & \text{or } x > c + \delta \\ (x - c + \delta)/\delta & \text{if } c - \delta \leq x \leq c \\ (c + \delta - x)/\delta & \text{if } c \leq x \leq c + \delta \end{cases}$$

In this equation, δ represents the uncertainty level. The larger the δ, the more uncertain the input data.

One kind of typical input data for an information fusion system is image, which is generally sampled into a rectangular array of pixels. Each pixel has an x-y coordinate that corresponds to its location within the image, and an intensity value representing brightness. The spec IMAGE *imports* INTEGER and REAL, and defines a *function sort*: $Image = Integer, Integer \rightarrow Real$. The spec FUZZIFICATION is generated by taking the *colimit* of IMAGE and FUZZY-ARITHM, and defining another *function sort*: $Fuzzy\_image = Integer, Integer \rightarrow Fuzzy\_number$. The diagram for this specification is shown in Figure 2. FUZZIFICATION maps $Image$ to $Fuzzy\_image$, so that each pixel has a corresponding fuzzy triangular number instead of a crisp number. Also in this spec, two operations are defined:

$$\begin{aligned} op\ fuzzify &: Real, Nonzero \rightarrow Fuzzy\_number \\ op\ fuzzify\_2 &: Real \rightarrow Fuzzy\_number \end{aligned}$$

where $fuzzify$ takes a crisp number and some uncertainty level, and generates a fuzzy triangular number. The operation $fuzzify\_2$ deals with the situation when the uncertainty level is zero, which means there is no fuzziness about the result. The latter operation is specified so that a crisp number can also be regarded as a fuzzy number.

2.2.2 Fuzzy reasoning

Fuzzy reasoning takes fuzzified inputs and applies fuzzy arithmetic operations on them. For instance, as we discussed above, the input can be a fuzzy image in which each pixel corresponds to a fuzzy triangular number. While for crisp numbers we apply some arithmetic operations, like $+, -, \times, and \div$, for fuzzy numbers we will apply f_add, f_sub, f_mult, and f_div, as specified in FUZZY-ARITHM. Some additional fuzzy operations are specified there too, which will be useful in our applications. One is fuzzy minimum($fmin$), another is fuzzy maximum($fmax$). Let A, B denote two fuzzy numbers, then

$$fmin(A,B)(z) = \sup_{z=min(x,y)} min[A(x), B(y)]$$
$$fmax(A,B)(z) = \sup_{z=max(x,y)} min[A(x), B(y)]$$

for all $z \in Real$. The results of these two operations are fuzzy numbers. These two operations introduce partial ordering of fuzzy numbers.

Corresponding logic operations such as fuzzy equal($fequal$) and fuzzy less than(flt) are also specified here. There are many ways to define such operations. Here we have chosen the following:

$$op\ fequal : Fuzzy\_number, Fuzzy\_number \rightarrow Fuzzy\_number$$
$$op\ flt : Fuzzy\_number, Fuzzy\_number \rightarrow Fuzzy\_number$$

The operation $fequal$ takes two fuzzy numbers, defuzzifies them and compares the difference of the result. If the difference is less than a threshold, $fequal$ will return a $fone$, which is generated by $fuzzify(one, \alpha)$. α is the value where the two membership functions intersect and α will be zero if there is no intersection. If the difference is larger than the threshold, $fequal$ will return a $fzero$, which is generated by $fuzzify(zero, \alpha)$. The intersection of the two membership functions are taken to generate α, the same way as in $fuzzify(one, \alpha)$. The result of $fequal$ and flt is either $fone$ or $fzero$. This is the fuzzy equivalent of boolean values true and false. They are not limited to stating whether something is a fact or not, but in addition to this, they give the value of the uncertainty associated with such a statement.

2.2.3 Defuzzification

The input to the defuzzification process is a fuzzy number, and the output is a crisp number. There are several defuzzification methods - *centroid calculation* that returns the center of the area under the curve of the fuzzy number, *middle of maximum* that returns the average of the maximum value of the fuzzy number, *largest of maximum*, and *smallest of maximum*. We chose the *largest of maximum* method to implement the defuzzification process.

Defuzzification is implemented in DEFUZZIFICATION, which is a *definitional extension* of FUZZY-NUMBER. This spec defines the *defuzzify* operation as: $op\ defuzzify : Fuzzy\_number \rightarrow Real$. It takes a fuzzy number, finds the largest of maximum of its membership function, and returns the real number as defuzzification result. In our situation we fuzzify the input data using triangular membership function, so after fuzzy operations are applied to these fuzzy triangular numbers, the result will always have only one peak value. Therefore the largest of maximum of its membership function will always return only one value. There are situations where other types of fuzzification are used, and then the defuzzification spec should be more complex.

3 An Example: Fuzzy Edge Detection

In this section, we will show how to use fuzzy information processing specifications to translate a standard detection algorithm into a fuzzy detection algorithm, and see how uncertainty of input data propagates during the process and influences the final decision.

3.1 Edge Detection Algorithm

An edge in an image could be considered as a boundary at which a significant change of *intensity*, I, occurs. Detecting an edge is very useful in object identification, because edges represent shapes of objects. There are many algorithms for edge detection. The objective of an edge detection algorithm is to locate the regions where the *intensity* is changing rapidly. So we can decompose the whole process into two steps, the first is to derive *edge points* in an image, the second is to apply edge detection method only to these points.

We use the *Laplacian-based* method to derive *edge points*. *Edge points* are where the second-order derivatives of the points are zero, *zero crossing*. So *edge points* can be searched by looking for *zero crossing points* of $\nabla^2 I(x,y)$, which can be calculated by the equation

$$\nabla^2 I(x,y) = I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)$$

In order to avoid false edge points, *local variance* is estimated and compared with a threshold. The local variance can be estimated by

$$\sigma^2(x,y) = \frac{1}{(2M+1)^2} \sum_{k_1=x-M}^{x+M} \sum_{k_2=y-M}^{y+M} [I(k_1,k_2) - m(k_1,k_2)]^2$$

where

$$m(x,y) = \frac{1}{(2M+1)^2} \sum_{k_1=x-M}^{x+M} \sum_{k_2=y-M}^{y+M} I(k_1,k_2)$$

with M typically chosen around 2. Since $\sigma^2(x,y)$ is compared with a threshold, the scaling factor $\frac{1}{(2M+1)^2}$ can be eliminated.

The spec EDGE-POINT *imports* IMAGE and defines a *sort* and some *ops*:

$$\begin{aligned} &sort\_axiom\ Edge\_point = \\ &(Integer, Integer) | edge\_point? \\ &op\ edge\_point? : Integer, Integer \\ &\qquad \to Boolean \\ &op\ grad : Integer, Integer \to Real \\ &op\ var : Integer, Integer \to Real \end{aligned}$$

where *grad* and *var* represent gradient and local variance respectively, and for all Integers x, y:

$$edge\_point?(x,y) \Longleftrightarrow grad(x,y) = 0 \wedge var(x,y) < thrd$$

Therefore a pixel at (x,y) satisfies an edge point if and only if the gradient equals zero and the local variance is less than the threshold. Otherwise the pixel is not an edge point.

3.2 Fuzzy Edge Detection

Now we will use fuzzy information processing specifications and translate the above edge detection algorithm into a fuzzy edge detection algorithm.

Fuzzy edge detection is specified in FUZZY-EDGE-POINT, which *imports* FUZZIFICATION, and defines a *function sort*:

$$Fuzzy\_edge\_point = Integer, Integer \to Fuzzy\_number$$

which maps each pixel to a fuzzy number representing the level at which the pixel satisfies an edge point. This fuzzy number represents *fuzzy boolean*. Instead of making the decision that a pixel *is* an edge point or *is not* an edge point, a $fone$ or a $fzero$ is given. A $fone$ states that the pixel satisfies an edge point with uncertainty as described by the fuzziness of this $fone$. A $fzero$, on the other hand, states that the pixel does not satisfy an edge point with uncertainty that is described by the fuzziness of this $fzero$. The following constants and operations are specified:

$$\begin{aligned} &const\ delta : Nonzero \\ &const\ thrd : Real \\ &op\ fgrad : Integer, Integer \to Fuzzy\_number \\ &op\ fvar : Integer, Integer \to Fuzzy\_number \end{aligned}$$

where $fgrad$ and $fvar$ represent fuzzy gradient and fuzzy local variance respectively. Calculation of $fgrad$ and $fvar$ requires fuzzy arithmetic operations that have been specified before. The operations $fequal$, flt and $fmin$ are also needed here to realize fuzzy edge detection. The operation $fequal$ takes two fuzzy numbers and returns a $fzero$ or a $fone$, representing how similar these two fuzzy numbers are. The operation flt takes two fuzzy numbers and returns a $fzero$ or a $fone$, representing how much the first one is less than the second one. For all Integers x, y:

$$\begin{aligned} Fuzzy\_edge\_point(x, y) = \\ fuzzy\_min[fequal(fgrad(x, y), \\ fuzzify(one, delta), \\ flt(fvar(x, y), fuzzify(thrd, delta))] \end{aligned}$$

Thus the likelihood that one pixel satisfies an edge point depends on both the likelihood that the fuzzy gradient is close to zero and the likelihood that the fuzzy local variance is less than a threshold. The more the fuzzy gradient is near zero and the fuzzy local variance is far less than the threshold, the more likely this pixel is an edge point. Then $fequal(fgrad(x, y), fuzzify(zero, delta))$ should return a $fone$ with less uncertainty, and $flt(fvar(x, y), fuzzify(thrd, delta))$ should also return a $fone$ with less uncertainty. Therefore $Fuzzy\_edge\_point(x, y)$ corresponds to a $fone$ with less uncertainty.

If $fequal(fgrad(x, y), fuzzify(zero, delta))$ returns a $fzero$, which means fuzzy gradient of the pixel (x, y) is not close to zero with some uncertainty, and if $flt(fvar(x, y), fuzzify(thrd, delta))$ also returns a $fzero$, which means fuzzy local variance of the pixel (x, y) is not less than a fuzzy threshold, then $Fuzzy\_edge\_point(x, y)$ should return a $fzero$, which is the fuzzy minimum of the two results and which shows that the pixel is not an edge point with some uncertainty.

If one of these two operations($fequal$ and flt) returns a $fzero$, and the other returns a $fone$, then $Fuzzy\_edge\_point(x, y)$ should return a $fzero$ which is the fuzzy minimum of the two results. It shows that the pixel is not an edge point with some uncertainty.

3.3 Results and Analysis

In order to show that with this approach we can reason about the influence of uncertainty of input information on the final decision before the system is built, we specify a GOAL spec, which *imports* FUZZY-EDGE-POINT and introduces a theorem:

$$\begin{aligned} \forall \delta_1, \delta_2 \in Real, \delta_1 \leq \delta_2 \\ \Longrightarrow \alpha_1 \leq \alpha_2 \end{aligned}$$

where δ_1 and δ_2 are two different values chosen to fuzzify the input data and represent the uncertainty levels of the input information, and α_1 and α_2 are the generated uncertainty values for deriving the results of $Fuzzy\_edge\_point(x, y)$ for the two different fuzzified images. These α_1 and α_2 represent the uncertainty levels in decision making. They are influenced by the result of the *fuzzy gradient* and the *fuzzy local variance*. It is natural that the more uncertain the input data the more uncertain the decision. Depending on the values δ_1, δ_2, α_1 and α_2, the theorem prover [10] returns either a "yes" or a "no".

In the above example we have applied fuzzy information processing specifications on a standard edge point derivation algorithm and the results show that the uncertainty of input data propagates through the whole process and influences the uncertainty level of the decision. The uncertainty of input data influences the fuzzy gradient and the fuzzy local variance results, which in turn influence the uncertainty of the decision. So instead of giving a crisp decision (true or false), a fuzzy decision is given: true with some uncertainty or false with some uncertainty. The relation between the uncertainty levels in the final decision and in the input information can be proved in this specification stage.

4 Conclusions and Future work

In this paper we have introduced a formal approach to characterize and manipulate uncertainty in information processing systems. We chose fuzzy set theory to represent uncertainty. We have shown how to specify basic elements of fuzzy set theory in Specware. As an example, fuzzy information processing specifications were applied to an edge detection algorithm. We showed how the uncertainty of input information propagates and influences the final decision.

In our future work, we will enrich the fuzzy set theory library by putting in more specifications for fuzzy set theory. α-cut is a powerful link between fuzzy set and crisp set, so more specs for α-cut will be built. We will also put more specs in the fuzzy information processing system. For instance, various fuzzification methods other than triangular will be specified. Trapezoidal, Gaussian, and bell fuzzification methods are three most popularly used. They can represent different levels and kinds of uncertainty among input data or decision making. Fuzzy reasoning will be enriched by defining different versions of fuzzy equal and fuzzy less than. Other defuzzification methods will also be specified.

Also in our future work, we will generalize this uncertainty topic by using *random set* instead of *fuzzy set* to characterize and manipulate uncertainty. We will also specify random processing and formally introduce *randomness* to some typical information processing problems.

References

[1] G. J. Klir. *On the Alleged Superiority of Probabilistic Representation of Uncertainty.* IEEE Transactions on Fuzzy Systems, Vol.2, No.1, 1994.

[2] G. J. Klir, U. S. Clair and B. Yuan. *Fuzzy Set Theory Foundations and Applications.* Prentice Hall PTR, 1997.

[3] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic Theory and Applications.* Prentice Hall PTR, 1995.

[4] J. R. Jang and C. Sun. *Neuro-Fuzzy Modeling and Control.* IEEE Transactions, 1995.

[5] S. A. DeLoach and M. M. Kokar *Category Theory Approach to Fusion of Wavelet-Based Features.* To be published on Fusion'99, 1999.

[6] J. S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice Hall, Inc, 1990.

[7] G. A. Baxes. *Digital Image Processing.* John Wiley & Sons, Inc, 1994.

[8] B. C. Pierce. *Basic Category Theory for Computer Scientists.* MIT Press, 1991.

[9] Y. V. Srinivas *Category Theory Definitions and Examples.* Technical report, Kestrel Institute, Palo Alto, California, 1990.

[10] *Specware Language Manual.* Kestrel Institute, Palo Alto, California, 1998.

[11] *Specware User's Guide.* Kestrel Institute, Palo Alto, California, 1998.