# Formalizing Classes of Information Fusion Systems

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

Jerzy A.Tomasik
Universite d'Auvergne
LLAIC1, BP86 63172 AUBIERE
France

Jerzy Weyman
Department of Mathematics
Northeastern University
Boston, MA 02115, USA

**Abstract**

This paper provides an outline of a formalization of classes of information fusion systems in terms of category theory and formal languages. The formalization captures both the inputs/outputs of a fusion system and the fusion processing algorithms. The paper also introduces a notion of *subclass*, which is used to compare classes of fusion systems, whether they are different or one is a special case of another. Two examples of classes of fusion systems formalized in the paper are data fusion and decision fusion; decision fusion is shown to be a subclass of data fusion. A number of other classes of fusion systems are defined. The formalization is extended by adding the notion of *measure of effectiveness*, which is then used to prove that one of the classes (so called *overlapping system*) is at least as efficient as a single-source system. And finally it is shown how data association can be formalized in this framework. While at first the formalization could be used by information fusion scientists to formally define various types of fusion systems and then to prove theorems about properties of such systems, it is expected that it should lead to the development of tools that could be used by software engineers to formally derive designs of fusion systems.

Keywords: information fusion, formal methods, category theory, classes, subclass relation, fusion systems

## 1 Introduction

Over the past two decades *information fusion* has established itself as an independent research area. However, in spite of a significant progress in research on information fusion, there is still a lack of a formal theoretical framework for defining various types of information fusion systems, defining and analyzing relations among such types, and finally designing information fusion systems using the formal method approach [1, 2, 3, 4]. In particular, although various classifications of fusion systems exist (e.g., the JDL classification [5]), the classifications are based on the input/output data, rather than on algorithms of fusion, and moreover, they are expressed mainly in natural language rather than fully in mathematics and logic. The consequence of this situation is that it is not possible to formally prove that one type of fusion system is superior to another. Instead, algorithms are compared as "black boxes", i.e., they are first implemented and then their performance is compared based on the simulated (or sometimes real) experiments. The main goal of this paper is to show how fusion systems can be formalized (in a logical and mathematical notation) to enable their theoretical analysis.

It is our belief that the main difficulty of formalizing information fusion lies in the fact that the real issues of information fusion are resolved at the time of designing an information fusion system, rather than at run time of such a system [6]. At run time, a fusion operation (algorithm) is executed. But the real challenge of information fusion is to derive such an algorithm, rather than to execute it. In our view then, the problem of information fusion lies in the search through a space of various algorithms for one that satisfies some prespecified criteria. The term "search" does not mean a search through a file of algorithms, since such a file does not exist; it is used here in a sense similar like in "searching for a solution to a problem". It may involve, for instance, synthesizing an algorithm using some primitive algorithms and algorithm composition operations.

The current situation in information fusion is that a designer proposes an algorithm for fusion and then tests it on either real or simulated data. So the real fusion process is done in the designer's head rather than in the computer. Our ultimate goal is to develop a framework in which various design solutions can be searched for and formally analyzed by a computer rather than solely by the designer. Towards this goal, we are interested in methods for analyzing fusion systems before they are built rather than testing their performance after the system is implemented (cf. discussion on the selection of a fusion algorithm based upon knowledge of the environment in [7]). In other words, we are interested in a *formal theory of information fusion.*

To address these issues we need to ask the question of what defines our search space (design space). The first step in this direction is answering the question of what is given to us in the formulation of a specific fusion problem that would define the primitives from which synthesis could start. It seems that we could consider the following three kinds of *knowledge*:

- knowledge of the sensors used,
- knowledge of the goal for a fusion system, and
- background knowledge (e.g., physics, geometry)

There are various notions of "knowledge" in various research communities. Since we follow the formal methods paradigm (cf. [1, 2, 3, 4]), to us knowledge means formal (logical) *theories* and their *classes of models* [8]. Theories may be given as collections of signatures of functions and relations and collections of axioms over the signatures. In particular, the knowledge of sensors means theories through which we interpret sensory data. The central element of such a theory is a *measurement function.* In our approach we conceptualize it as a function that assigns sensory values to specific coordinates. Knowledge of the goal may mean, for instance, theories describing targets to be detected by an Automatic Target Recognition system (ATRS) and a target recognition function that assigns targets to world locations. In the process of development of a fusion system, at first only the signature of such a function is known. The function is then realized by a specific target recognition algorithm. The algorithm is synthesized using some knowledge from the three types - theories of sensors, signatures of the goal function and background knowledge theories.

The next question is what should be the composition operators and the formalism in which such a search problem can be specified and an algorithm synthesized. The constraint here is that the formalism must be able to capture various theories, for instance theories of sensors and targets, models of the theories, as well as relations among them. And moreover, the formalism must include combination operators for this kind of components. The set theory operators, like union, shared union, intersection, Cartesian product, cannot be used here since the components (theories and models) are not just set elements, they are structured elements. To be able to manipulate structured elements, the operators of set theory must be extended.

We came to the conclusion that the formalism that best satisfies these requirements is category theory (cf. [9]). Category theory is a mathematically sound representation technique used to capture the commonalities and relationships between structured objects, in particular theories and their models. This feature makes category theory a very elegant language for describing information fusion systems and the information fusion process itself. This formalism is very convenient for combining (fusing) such structures. In particular, it provides the operators of *colimit* and *limit* that allow us to combine such structures in a sound and rigorous way. The colimit operator is a generalization of the shared union operator and the limit operator is a generalization of the Cartesian product operator.

Finally, we would like to have some tool support for manipulating various theories in the process of specifying a fusion problem and searching for solutions. In other words, we need a language in which we could specify all of the knowledge and a tool that would support (semi)automatic analysis of such specifications. To satisfy this requirement we used the Specware specification tool (Kestrel Institute) and Slang, its specification language. Specware is based on category theory. In Specware [10], category theory objects are called specs (short for "specifications"). Specware supports the colimit operation. It also supports progressive modular development of specifications. Additionally, it supports the process of *refinement* - the process of progressive translation of specifications into code. The refinement process is guaranteed to be correct, i.e., the code satisfies its specification.

In this paper we show some results of our investigations into an information fusion theory within the category theory based framework. In the next section we describe a simple example of one class of information fusion, i.e., the *data fusion*. Then we give a formal definition of this class. Then we define a class called *decision fusion* (cf. [11, 12]). The results in this paper include the definition of a *subclass* relation between classes of information fusion systems in Section 4, proof that decision fusion is a subclass of data fusion in Section 4.1, identification of necessary and sufficient conditions for decision fusion to be equivalent to data fusion in Section 4.2 and identification of a number of additional classes of information fusion systems in Section 5. Section 6 extends the formalization by adding a *measure of effectiveness* of a fusion system. In Section 7 we briefly discuss the way *data association* is dealt with within our framework. In Section 8 we review other approaches to the formalization of fusion. And finally in Section 9 we present conclusions.

## 2 Example of Data Fusion

In order to explain our ideas presented in this paper we use a simple example of an information fusion scenario. We consider two vision sensors $Sens_1$ and $Sens_2$ observing an object in the world. The first sensor, $Sens_1$, returns the image denoted as $I_1(x_1, y_1)$ and the second sensor, $Sens_2$, returns the image $I_2(x_2, y_2)$. The signatures of sensory output functions can be represented as $I_1 : X_1 \times X_2 \rightarrow V_1$, $I_2 : X_2 \times Y_2 \rightarrow V_2$. These functions assign intensity values to world coordinates. We assume that $I_1$ and $I_2$ consist of two sub-functions. For $Sens_1$, there is a function $g_1(x_1, y_1)$ which returns pixel values, which are then filtered by $h_1(z_1)$. The composition of these two functions returns the values of $I_1(x_1, y_1)$. Similarly, $Sens_2$ consists of two functions $g_2$ and $h_2$.

The goal of the fusion system is to utilize the information from both sensors in order to detect edges of the observed object. This goal can be represented by a goal function $\Delta : X \times Y \rightarrow E$, where $X, Y$ represent the world coordinates and $E$ represents edge points. This goal can be achieved in two ways:

1. *Data Fusion:* Two images $I_1(x_1, y_1)$ and $I_2(x_2, y_2)$ are fused into one combined image $I(x, y)$ and then edge detection is performed on this image. The resulting edges (or more precisely, *edge points*) are denoted by $E(x, y)$.

2. *Decision Fusion:* The two images $I_1(x_1, y_1)$ and $I_2(x_2, y_2)$ are analyzed separately by edge detection algorithms. This results in edges $E_1(x_1, y_1)$ and $E_2(x_2, y_2)$. Then the

detection information (edges) is fused into one $E(x, y)$.

As we can see, in the end both systems derive the same kind of global information about edges represented by $E(x, y)$. For simplicity we assume that edge detection is based on the magnitude of the gradient, for the image of $Sens_1$, for the image of $Sens_2$ and for the fused image.

## 3 Formal Definition of Fusion

### 3.1 The Formalization Approach

Formalizations of a specific problem are usually called *formal specifications.* There are two kinds of specifications: *declarative* and *operational.* In this paper we use the declarative approach. In particular, we formalize fusion systems and subsystems in terms of *algebraic theories.* Since a fusion system is to be composed of a number of subsystems, we also need some structuring mechanism in which more complex specifications are composed from simpler specifications. For this purpose we use the structuring operations of *category theory* (cf. [9]). A *category* is a mathematical structure consisting of *category objects* and *category arrows* (or *morphisms*). Category objects are the objects in the category of interest. For instance, in category `Set` the objects are all sets. Category arrows are the mappings that define the relationships between pairs of (possibly structured) objects, with one object called its *domain* and the other called *codomain.* One of the arrows must be an *identity* arrow. In the category `Set` the arrows are total functions between sets. A category has a *composition* operation which assigns an arrow to a pair of arrows (the pair must satisfy the domain/codomain compatibility condition). The composition operation must be associative. A *diagram* in a category is a collection of objects and a collection of arrows between these objects. In the category `Set` the composition is just the composition of functions and the identity arrow is the identity function. An arrow identifies parts that are common to the two objects. The *colimit* operation creates an object for a given diagram so that the common (*shared*) parts are unified and the rest of the parts are inserted into the new object while preserving the structures of all the objects.

To formalize specifications we use the category `Spec` in which objects are algebraic specifications and the category arrows are *morphisms* among the specifications. Algebraic specifications are pairs $(\Sigma, T)$, where $\Sigma$ are *signatures* and $T$ - *theories* over the signatures. Signatures have the following form: $\Sigma = (\sigma, F)$, where $\sigma$ are *sorts* and $F$ are functions over the sorts. Theories associated with the signatures are collections of axioms over the signatures. Signatures and associated theories are called *specifications*, or for short *specs*. In the rest of this paper we will represent specs in the following form:

$$S = ((\sigma, F), T). \tag{1}$$

First we will show sorts and signatures of operations delimited by parentheses. Then we will show the axioms of a specific theory, if any.

Specs are considered as *objects* in the category `Spec` related through *morphisms*. More specifically, morphisms map sorts and operations of the *source* spec to sorts and functions of the *target* spec. One special kind of morphism is called *definitional extension*. A morphism $S \to T$ is called a (strict) definitional extension if it is injective and if every element of $T$ which is outside of the image of the morphism is either a defined sort or a defined operation. The target spec $T$ is also called *definitional extension of S*. Specs and morphisms are represented as diagrams. The colimit operation creates a new specification from a diagram of existing specifications. This new specification has all the sorts and operations of the original set of specifications without duplicating the shared sorts and operators. We always assume that our theories are *consistent*, i.e., that they have models, formally denoted as $M \models T$. We implement and verify our specifications in the Slang language [13, 14]. In the paper, however, we use mainly the mathematical notation.

The above introduction to category theory and algebraic specification is semi-formal and incomplete. The purpose of this section was to give an overview of the concepts used in this paper. For a more complete treatment of category theory the reader can refer to e.g., [9]. Algebraic specifications are described in [15]. Examples related to information fusion can be found in [16, 17, 18].

## 3.2 Data Fusion

The two kinds of information fusion systems known in the subject literature (cf. [19, 12]) are *data fusion* and *decision fusion*. Since, as we later prove in this paper, decision fusion is a subclass of data fusion, we first define this more general class of information fusion systems. We first introduce the diagram of a data fusion system (see Figure 1) in the category `Spec`. This diagram consists of five nodes (specs) and six arrows (morphisms). The nodes represent the following specifications: $S_c$ - world coordinates; $S_1$, $S_2$ - sensors, $S_w$ - world theory, $S_f$ - the fused system. We describe each of the nodes in the sections that follow. The arrows specify which sorts and operations are unified as described in Section 3.1. First, we give a general description of each spec, and then explain their meaning in relation to the example of Section 2.

$$\begin{tikzcd}
 & S_f & \\
S_1 \arrow[ur] & S_w \arrow[u] & S_2 \arrow[ul] \\
 & S_c \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 1: Data Fusion

### 3.2.1 The World Specification, $S_w$

In the diagram of Figure 1 we assume that

$$S_w = ((L, E, \Delta : L \to E), T_w) \tag{2}$$

specifies the world that both sensors observe. The specification includes signatures $(L, E, \Delta)$ and axioms $(T_w)$. $L$ represents the sort of world coordinates (it can be, for instance a 2D space where $L = X \times Y$), $E$ is the objects in the world; they "occupy" locations in the world. The function $\Delta$ assigns these objects to particular locations. We do not assume that we always know this function, but we assume that we may know it for a number of cases. This function is given as part of each of the models from the set of models $M$. This is also referred to as *ground truth*. These known models are used for testing the resulting fusion system. Additionally, the specification of the world can contain theories $T_w$ (axioms) that capture known dependencies and constraints that the world is known to obey.

Implicit in this formulation is the fact that the goal of the fusion system is to recognize objects in the world, or more precisely, assign object names to all locations in the world. For other goals, the fusion problem could have a different signature, but the idea should remain the same.

Referring to the example of Section 2, the coordinates of the world are $X, Y$, i.e., $L = X \times Y$. The objects are $E = [0, 1]$ - a subset of real numbers representing the confidence of an edge point being at a particular world location. We may define the function $\Delta$ that assigns objects to each location in the world in a number of ways. For instance, we can map each location to a subset of $[0, 1]$:

$$\Delta : X \times Y \to 2^{[0,1]} \tag{3}$$

In such a case the subset represents the possible values of "edgeness" that can be associated with a given location. Alternatively, we can define $\Delta$ as a map to an element of the interval $[0, 1]$, i.e.,

$$\Delta : X \times Y \to [0, 1] \tag{4}$$

In this case the function $\Delta$ returns just one edge point (one value of edgeness) for each world location.

### 3.2.2 The Sensor Specifications, $S_1, S_2$

The specifications $S_1$, $S_2$ represent specifications of two sensors.

$$S_1 = ((L_1, V_1, f_1 : L_1 \to V_1), T_1) \tag{5}$$

$$S_2 = ((L_2, V_2, f_2 : L_2 \to V_2), T_2) \tag{6}$$

Each sensor has its own coordinate sort: $L_1$ is the coordinate sort of the sensor specified by $S_1$ and $L_2$ is the coordinate sort of the sensor specified by $S_2$. $V_1$ and $V_2$ are the sorts of values returned by the sensors. The functions $f_1$ and $f_2$ are the *measurement functions* of $Sens_1$ and $Sens_2$, respectively. $T_1$ and $T_2$ specify theories of sensor operation.

In our example, both sensors have the coordinates denoted as $L_1 = X_1 \times Y_1$ and $L_2 = X_2 \times Y_2$, respectively. Their measurement functions are $f_1 = I_1$ for $Sens_1$ and $f_2 = I_2$ for $Sens_2$. The measurement functions return the values from $V_1$ and $V_2$, respectively. Since $I_1$ and $I_2$ are compositions of two functions, the theories of $S_1$ and $S_2$ must have appropriate axioms to this effect.

$$I_1 = h_1 \circ g_1 \tag{7}$$

$$I_2 = h_2 \circ g_2 \tag{8}$$

where $g_1$ and $g_2$ return values from $V_{11}$ and $V_{21}$, respectively. The specification of the first sensor, $Sens_1$, is shown below. We do not show the specification for the second sensor since it is similar to the specification of the first sensor.

$$S_1 = ((X_1, Y_1, V_1, V_{11}, g_1 : X_1 \times Y_1 \to V_{11}, h_1 : V_{11} \to V_1, I_1 : X_1 \times Y_1 \to V_1), I_1 = h_1 \circ g_1) \tag{9}$$

The sensor specification includes all the sorts and the signatures. Then, in its theory part, it includes the axiom stating that the function $I_1$ is computed as a composition of the measurement function $g_1$ and the filtering function $h_1$ (see Eq. 7).

### 3.2.3 The Coordinate Sort Specification, $S_c$

So far we have shown three systems of coordinates: $L$, $L_1$ and $L_2$. Each of them can have a number of components. In our example we showed two components for each. In the final specification we need to show which coordinates are treated as the same coordinates. From the notation in the example one might suspect that $X$, $X_1$ and $X_2$ refer to the same world coordinate. But this fact would have to be made explicit in the specification. To achieve this goal using the category theory formalism we need to define a diagram that captures this fact. Towards this goal we introduce a spec $S_c$ that contains only a number of *unifying coordinate sorts* (note that the axiom set is shown as empty)

$$S_c = ((C_1, \ldots, C_n), \emptyset) \tag{10}$$

and three morphisms: $S_c \to S_w$, $S_c \to S_1$, $S_c \to S_2$, as shown in Figure 2.

For our example the unifying sorts in the $S_c$ spec are:

$$S_c = ((C_1, C_2), \emptyset). \tag{11}$$

$$\begin{tikzcd}
S_1 & S_w & S_2 \\
 & S_c \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 2: Unification of Coordinate Sorts

We assume that we want to associate $X_1$ and $X_2$ with $X$, $Y_1$ and $Y_2$ with $Y$. The unification of sorts is achieved by specifying the three morphisms:

$$S_c \rightarrow S_1 = \{C_1 \rightarrow X_1, C_2 \rightarrow Y_1\} \tag{12}$$

$$S_c \rightarrow S_2 = \{C_1 \rightarrow X_2, C_2 \rightarrow Y_2\} \tag{13}$$

$$S_c \rightarrow S_w = \{C_1 \rightarrow X, C_2 \rightarrow Y\} \tag{14}$$

### 3.2.4 The Goal Specification, $S_f$

Now we are ready to construct the *goal specification*, $S_f$. We call this so since it contains the goal function of the ultimate fusion system. The specification $S_f$ is obtained in two steps. First, the colimit of the diagram in Figure 2 is constructed. At this point some of the sorts in the specs $S_w, S_1, S_2$ are identified (or "glued" together). For instance, the six sorts in our example $(X, Y, X_1, Y_1, X_2, Y_2)$ would form two equivalence classes $\{X, X_1, X_2\}$ and $\{Y, Y_1, Y_2\}$. Note that this does not mean that in the final spec we would not distinguish between the variables defined in the original specs. We would still have the variables representing the values coming from the two sensors separately. Only after *data association* (discussed in Section 7) is done could we use the same variables for the two sensors. In this paper we assume, for simplicity, that the coordinates of the two sensors are perfectly associated and thus we will use the symbols $X$ and $Y$ to represent the coordinates of the two sensors in the final specification of the system.

In the second step the resulting specification is extended by adding the goal function $D_f$. Its signature is constructed out of the signatures of the two sensors and of the world. This function takes two measurement functions $f_1 \in (L_1 \rightarrow V_1)$, $f_2 \in (L_2 \rightarrow V_2)$ as inputs and returns a decision function that assigns subsets of objects to the world coordinates (note that we write $(L_1 \to V_1)$ to represent the space of functions from $L_1$ to $V_1$).

$$D_f : (L_1 \to V_1) \times (L_2 \to V_2) \to (L \to 2^E) \tag{15}$$

So the final spec $D_f$ is of the following form:

$$S_f = ((L, E, \Delta : X \to E, L_1, V_1, L_2, V_2, f_1 : L_1 \to V_1, f_2 : L_2 \to V_2,\\ D_f : (L_1 \to V_1) \times (L_2 \to V_2) \to (X \to 2^E)), T_f) \tag{16}$$

For our example, the morphisms $S_1 \to S_f$, $S_2 \to S_f$ and $S_w \to S_f$ would be specified first (similarly as the morphisms shown above) and then the colimit operation would be specified next. The resulting specification would include the sorts $X, Y, E$, the operations $I_1, I_2, g_1, g_2, h_1, h_2$ and all the axioms from $S_w, S_1, S_2$. The colimit operation would guarantee that sorts are unified appropriately, and the operations are applied to the appropriate sorts. Additionally, it would insure that the axioms from the source specifications are preserved, i.e., they are theorems of the colimit specification. This kind of mechanisms for formally checking the colimit operation are part of the Specware tool [14].

The signature of the fusion function for our example would take the form as shown in Eq. 17 below.

$$D_f : (X_1 \times Y_1 \to V_1) \times (X_2 \times Y_2 \to V_2) \to (X \times Y \to E) \tag{17}$$

Note that the mapping is to the set $E$ rather than to $2^E$. This means that we expect a concrete value for each of the objects (in this case, edges) rather than a distribution of confidence as a result of the fusion process. This differs from our general specification where the mapping is to $2^E$. The rationale behind the mapping specified in Definition 1 is to show that the decision is not always unique, in some cases it may return a number of possibilities rather than just one specific object.

### 3.2.5 Definition: Data Fusion

Now we summarize our discussion from the previous subsections in the following definition of the class of data fusion systems.

**Definition 1** *A data fusion system consists of the following specs related through morphisms as shown in Figure 1:*

$$S_w = ((L, E, \Delta : X \to E), T_w) \tag{18}$$

$$S_1 = ((L_1, V_1, f_1 : L_1 \to V_1), T_1) \tag{19}$$

$$S_2 = ((L_2, V_2, f_2 : L_2 \to V_2), T_2) \tag{20}$$

$$S_c = ((C_1, \ldots, C_n), \emptyset) \tag{21}$$

$$\begin{aligned} S_f = ((L, E, \Delta : L \to E, L_1, V_1, L_2, V_2, f_1 : L_1 \to V_1, f_2 : L_2 \to V_2, \\ D_f : (L_1 \to V_1) \times (L_2 \to V_2) \to (L \to 2^E)), T_f) \end{aligned} \tag{22}$$

Note that the spec of Eq. 22 includes all of the other specs. So one might think that this spec should be sufficient for a definition of a data fusion system. However, in such a case the relations between the sources of information, the world and the final spec would not be specified. In other words, any specs $S_1, S_2, S_w, S_c$ could be used. By adding the morphisms $S_c \to S_w,\ S_c \to S_1,\ S_c \to S_2$ and the requirement that the resulting spec $S_f$ must be the colimit of the diagram shown in Figure 2 we significantly constrained the set of specs that can satisfy this definition. In particular, note that all of the axioms (theories) associated with the specs of sensors $S_1,\ S_2$ and of the world $S_w$ must be theorems in $S_f$. This wouldn't be required if only Eq. 22 was used to define a data fusion system.

Ideally, we would like to have a perfect data fusion system, i.e., such that satisfies the following requirement:

$$T_f \vdash \forall_{x \in X}\ D_f(f_1, f_2)(x) = \Delta(x), \tag{23}$$

This requirement states that the resulting decision function $D_f$ is compatible with the world (ground truth) specified through function $\Delta$. This would mean that the fusion system can always find a unique solution and that the decision would always be correct. Such a strong requirement is difficult to achieve in practice. Thus instead, we can have a somewhat weaker requirement by replacing the equality in this equation by the inclusion ($\in$); we call such a system a *correct data fusion system.*

**Definition 2** *A data fusion system of Definition 1 will be called a* correct data fusion system *if it satisfies the following condition:*

$$T_f \vdash \forall_{\Delta} \forall_{f_1, f_2} \forall_{x \in L} \; \Delta(x) \in D_f(f_1, f_2)(x) \tag{24}$$

This definition states that the axioms of the goal specification constrain the sets of decisions so that the true value is always included in the system's decision. This issue will be discussed in more detail in Section 6 after we introduce the notion of measure of effectiveness of a fusion system.

Another way of relating a data fusion system to the ground truth is to require that it satisfy the known ground truth given in the form of *models*. Note that the term "model" is used here in the sense of logic and model theory, where a model is a relational structure (cf. [8]). For a decision fusion system a model would include explicit specifications of measurement functions $f_1$, $f_2$ and the values of the function $\Delta$ for these input functions. We will call a system that agrees with a given set of models a *model-satisfying data fusion system.*

**Definition 3** *Consider a data fusion system as in Definition 1 and a collection of models $M = \{M_1, \ldots, M_n\}$. A data fusion system will be called a* model-satisfying data fusion system *if the following condition holds:*

$$M_i \models T_f, \forall M_i \in M \tag{25}$$

In practice, these models would serve as test cases that are used to check the system we are designing. It is expected that the system will perform correctly at least on the given test cases.

Returning to the example used in this paper, the models would provide images representing the measurement functions $I_1^i, I_2^i$ as well as the decision functions $\Delta^i(x, y)$ that would assign the value of $E$ for each location $x, y$. To check that the designed system performs correctly we would need to show that for each $M_i \in M$ we have $M_i \models T_f$. This would be achieved by showing that

$$\forall_i \forall_{x,y} D_f(I_1^i(x, y), I_2^i(x, y)) = \Delta^i(x, y) \tag{26}$$

## 3.3 Decision Fusion

The class we have described so far is termed in the literature (cf. [19]) *data fusion* systems. Another class is known as *decision fusion* systems.

**Definition 4** *A decision fusion system consists of the following seven specs related according to the diagram of Figure 3.*

$$S_w = ((L, E, \Delta : L \to E), T_w) \tag{27}$$

$$S_1 = ((L_1, V_1, f_1 : L_1 \to V_1), T_1) \tag{28}$$

$$S_2 = ((L_2, V_2, f_2 : L_2 \to V_2), T_2) \tag{29}$$

$$S_c = ((C_1, \ldots, C_n), \emptyset) \tag{30}$$

$$S_{d1} = ((L_1, V_1, \Delta : L \to E, f_1 : L_1 \to V_1, D_1 : (L_1 \to V_1) \to (L \to 2^E)), T_{d1}) \tag{31}$$

$$S_{d2} = ((L_2, V_2, \Delta : L \to E, f_2 : L_2 \to V_2, D_2 : (L_2 \to V_2) \to (L \to 2^E)), T_{d2}) \tag{32}$$

$$\begin{aligned} S_d = ((L_1, V_1, L_2, V_2, \Delta : L \to E, f_1 : L_1 \to V_1, D_1 : (L_1 \to V_1) \to (L \to 2^E), \\ f_2 : L_2 \to V_2, D_2 : (L_2 \to V_2) \to (L \to 2^E), \\ D_d : (L \to 2^E) \times (L \to 2^E) \to (L \to 2^E)), T_d) \end{aligned} \tag{33}$$

Similarly as for data fusion systems, we can define a *correct decision fusion system* by requiring that it satisfy the condition of Eq. 24 and a *model-satisfying decision fusion system*, when it satisfies Eq. 25.

The first four specs are the same as in the definition of data fusion. The functions $D_1, D_2$ are the decision functions for the two sensors. They could have been used for making decisions when only one of the sensors is available. In the process of decision fusion these two decision functions are used instead of raw data. The spec $S_d$ represents the decision fusion block. Note that in this spec $D_d$ takes the assignments that are the results of application of functions $D_1$ and $D_2$ and combines these two assignments into one (fused) assignment.

Returning back to our example, we take the decision function $D_1$ to have the signature

$$D_1 : (X_1 \times Y_1 \to V_1) \to (X \times Y \to E) \tag{34}$$

$$\begin{tikzcd}
 & S_d & \\
S_{d1} \arrow[ur] & & S_{d2} \arrow[ul] \\
S_1 \arrow[u] & S_w \arrow[uu] \arrow[ul] \arrow[ur] & S_2 \arrow[u] \\
 & S_c \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 3: Decision Fusion

In other words, the decision function $D_1$ takes the function $I_1$ and returns another function (the decision function) which maps the world coordinates to the values of edges. An edge in an image is manifested through a discontinuity (for continuous images) or a significant jump in the intensity value (in a digital image). There are various edge detection techniques (cf. [20, 21]). The simplest method is to take the gradient magnitude as the value of the edgeness at a specific pixel. Denoting the (normalized) gradient magnitude by $G(I)(x, y)$ we would have

$$D_1 \equiv G(I_1) \tag{35}$$

This information would be incorporated into the theory $T_{d1}$ shown in the spec $S_{d1}$. $T_{d1}$ would then incorporate the axioms about the gradient magnitude operator and the thresholds used for detection. Although $D_2$ could use a different edge detection algorithm, in this paper we assume, for simplicity, that $D_2$ also uses the same kind of "edgeness" operator. The $D_d$ operator can be defined in many different ways, for instance:

$$D_d(G_1, G_2)(x, y) \equiv \bar{G}(x, y) = \frac{1}{2}(G(I_1)(x, y) + G(I_2)(x, y))$$

## 4 The *subclass* Relation

In order to be able to compare various kinds of fusion systems we introduce the relation of *subclass*, which is a relation between classes of fusion systems. The notion of *subclass* is present in most object-oriented modeling languages, e.g., UML [22], DAML [23, 24, 25], OWL [26], or Java. Informally, the meaning of this relation is that one class is a subclass of another if each instance of the former is also an instance of the latter. This definition is sufficient only if the classes are interpreted as simply sets of elements. For instance, this is the case for such languages as DAML and OWL. But this would not be sufficient to specify the subclass relation among programs. For instance, the semantics of UML specifies the subclass relation in terms of another relation called *substitutability*. Each object-oriented programming language has its own interpretation of subclass, not necessarily compatible with the subclass notion of UML.

For fusion systems, we could say that the class of systems defined by $S_{f1}$ is a subclass of systems defined by $S_{f2}$ if each of the instances of $S_{f1}$ is also an instance of $S_{f2}$. But how would one decide whether a given fusion system is an instance of a given class of systems? Clearly, any two systems differ in some respect. For instance, consider two systems that differ in only one line of code, i.e., one of them has the line `a := b + c`, while the other has `a := c + b`. Are they equivalent or not? We propose a formal operational definition of the notion of *subclass* such that will allow us to decide when the relationship of subclass holds, given the formal specifications of two classes of fusion systems. Roughly, the interpretation of the subclass relation for fusion systems is that each system that satisfies the spec $S_{f1}$ must also satisfy the spec $S_{f2}$. The following definition makes this notion precise for the class of data fusion.

**Definition 5** *Let $S_{f1}$ and $S_{f2}$ be two classes of data fusion systems like in Figure 1, where all nodes except $S_{f1}$ and $S_{f2}$ are the same. We say that $S_{f1}$ is a subclass of $S_{f2}$ (meaning that $S_{f1}$ and $S_{f2}$ are related by* subclass*) if there is a morphism of specifications $\mu : S_{f2} \rightarrow \bar{S}_{f1}$, where $\bar{S}_{f1}$ is a definitional extension of $S_{f1}$, such that the diagrams shown in Figure 4 commute.*

$$\begin{tikzcd}
\bar{S}_{f1} & S_{f2} \arrow[l] & \bar{S}_{f1} & & S_{f2} \arrow[ll] & \bar{S}_{f1} & S_{f2} \arrow[l] \\
S_1 \arrow[u] \arrow[ur] & & & S_w \arrow[ul] \arrow[ur] & & & S_2 \arrow[u] \arrow[ul]
\end{tikzcd}$$

Figure 4: Commutativity Requirements for *subclass* Relations

Figure 5 explains the meaning of the diagrams of Figure 4. First, this definition captures the fact that the input information in both classes of fusion systems is the same, i.e., they both deal with the same world and use the same sensors. Second, since the morphism arrow points from $S_{f2}$ to $S_{f1}$ therefore all the constraints (sorts, operations and axioms) of $S_{f2}$ are mapped to $S_{f1}$ and thus $S_{f1}$ must obey all of them, and possibly more. And finally, the three diagrams of Figure 4 ensure that the mappings provided by the morphism $\mu$ agree with the mappings from $S_1, S_w, S_2$ to $S_{f1}$ and $S_{f2}$, respectively.

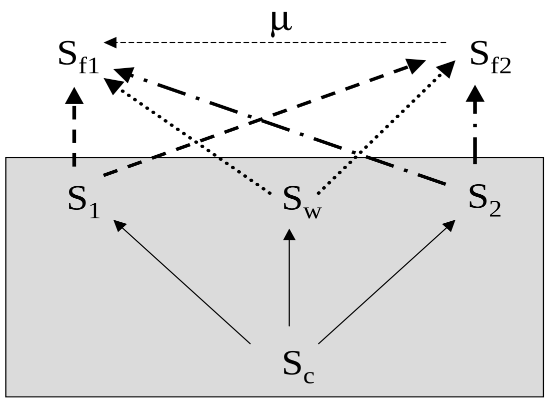

Figure 5: The *subclass* Relation Among Two Classes of Data Fusion Systems

**Definition 6** *Two classes of data fusion systems $S_{f1}$ and $S_{f2}$ are equivalent if both $S_{f1}$ is a subclass of $S_{f2}$ and $S_{f2}$ is a subclass of $S_{f1}$.*

## 4.1 Decision Fusion as a Subclass of Data Fusion

Now we apply the idea of subclass to any two classes of fusion systems, not necessarily two classes of data fusion systems. In particular, we show that decision fusion is a special case of data fusion (see Figure 6).

**Theorem 1** *The class of decision fusion systems, as defined in Figure 3, is a subclass of data fusion systems, as defined in Figure 1.*

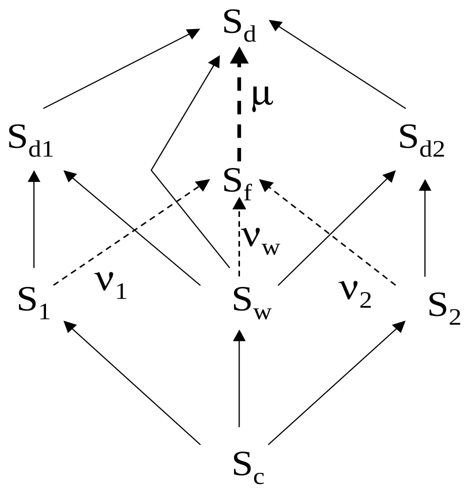

Figure 6: Decision Fusion is a Subclass of Data Fusion

**Proof:**

Assume that we have a decision fusion diagram as in Figure 3. In order to prove that this class is a *subclass* of data fusion we need to show (according to Definition 5), that there is a morphism $\mu = S_f \rightarrow S_d$, such that the three diagrams shown in Figure 4 commute. In other words, we need to produce a data fusion diagram as in Figure 1 such that there is a morphism from $S_f$ of the diagram of Figure 1 to $S_d$ of the diagram of Figure 3. Towards this aim, we define $S_f$ as a definitional extension of $S_d$ by defining a new function $\bar{D}_f : (L_1 \rightarrow V_1, L_2 \rightarrow V_2) \rightarrow (L \rightarrow 2^E)$:

$$\bar{D}_f \equiv D_d \circ (D_1 \times D_2). \tag{36}$$

The meaning of this equation is that an element from $D_1 \times D_2$ is a pair of functions whose values are in $(L \rightarrow 2^E)$, i.e., in the domain of $D_d$. Consequently, $D_d$ can be composed with the function defined by $D_1 \times D_2$ giving a data fusion function as a result.

The definitional extension [13] $S_f$ is equipped with an embedding $S_f \rightarrow S_d$ which is the identity on all sorts, operations and axioms from $S_d$. We define the arrows $S_1 \rightarrow S_f$, $S_2 \rightarrow S_f$ as compositions

$$S_1 \rightarrow S_f \equiv S_1 \rightarrow S_{d1} \rightarrow S_d \rightarrow S_f \tag{37}$$

$$S_2 \to S_f \equiv S_2 \to S_{d2} \to S_d \to S_f \tag{38}$$

Then we define the arrow $S_w \to S_f$ as a composition

$$S_w \to S_f \equiv S_w \to S_{d1} \to S_d \to S_f \tag{39}$$

We can easily check that the new diagram we constructed is a data fusion diagram as in Figure 1. The identity morphism $S_f \to S_d$ makes $S_d$ a subclass of $S_f$ according to Definition 5. Therefore the class of decision fusion systems is a subclass of data fusion systems. This concludes the proof. ∎

For our example, the fusion function $D_f$ for this system is

$$D_f \equiv \bar{G} \circ (G_1 \times G_2) \tag{40}$$

## 4.2 Can Data Fusion be a Subclass of Decision Fusion?

We have already proved that decision fusion is a subclass of data fusion. To prove this we were able to construct a diagram of data fusion from a diagram of decision fusion. The main point of the proof of this theorem was to construct the function $D_f$. We can restate the result of this theorem in terms of this function:

$$\begin{aligned}&\forall_{D_1:(L_1\to V_1)\to(L\to 2^E),D_2:(L_2\to V_2)\to(L\to 2^E),D_d:((L_1\to V_1)\to(L\to 2^E)\times(L_2\to V_2)\to(L\to 2^E))\to(L\to 2^E)}\\ &\exists_{D_f:(L_1\to V_1)\times(L_2\to V_2)\to(L\to 2^E)} \bullet \forall_{f_1:L_1\to V_1,f_2:L_2\to V_2} D_f(f_1,f_2) = D_d(D_1(f_1),D_2(f_2))\end{aligned} \tag{41}$$

Now comes the question of when a class of data fusion systems is a subclass of decision fusion systems. If we were able to prove that any data fusion system is also a decision fusion system, then according to Definition 6, the two classes (decision and data fusion) would be equivalent, and thus there would not be any good reason for introducing such a distinction. Below we show that this is not the case and that actually it is very difficult to satisfy such a requirement.

Note that in our framework, the statement that a class of data fusion systems is a subclass of decision fusion systems would be expressed by the diagram of Figure 7, which shows what it would take to construct a decision fusion diagram out of a data fusion diagram. While the condition of Eq. 41 is a necessary condition for a class of data fusion systems to be a subclass of decision fusion, the following corollary from Theorem 1 gives a sufficient condition.

**Corollary 1** *A class of data fusion systems $S_f$ is a subclass of decision fusion if and only if for a given function $D_f$ of the data fusion class $S_f$ the condition of Eq. 42 is satisfied.*

$$\exists_{D_1:(L_1\to V_1)\to(L\to 2^E),D_2:(L_2\to V_2)\to(L\to 2^E),D_d:((L_1\to V_1)\to(L\to 2^E)\times(L_2\to V_2)\to(L\to 2^E))\to(L\to 2^E)} \bullet$$
$$\forall_{f_1:L_1\to V_1,\ f_2:L_2\to V_2} D_f(f_1, f_2) = D_d(D_1(f_1), D_2(f_2)) \tag{42}$$

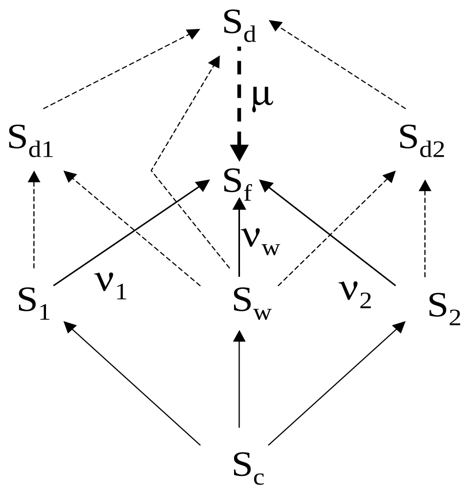

Figure 7: Data Fusion as a subclass of Decision Fusion

The following example shows the difficulty of constructing a decision fusion diagram out of a data fusion diagram. Consider a data fusion system in which the measurement functions $f_1, f_2$ have domains and ranges equal to the closed interval $[0, 1]$ of real numbers and the fusion function is defined as:

$$D_f(f_1, f_2)(x) = \begin{cases} 1 & : \quad 0.5(f_1(x) + f_2(x)) \geq 0.5 \\ 0 & : \quad 0.5(f_1(x) + f_2(x)) < 0.5 \end{cases} \tag{43}$$

Suppose the sensory inputs for a particular situation are given as two functions $f_1(x) = x$ and $f_2(x) = 0.6x$. The data fusion system will generate a decision $D_f(f_1, f_2)(x) = 1$ for $x \geq 0.625$ and $D_f(f_1, f_2)(x) = 0$ for $x < 0.625$. For this situation, the selection of the following decision functions $D_1, D_2$ and the decision fusion function $D_d$ would satisfy the requirement of Corollary 1:

$$D_1(f_1)(x) = \begin{cases} 1 & : \quad f_1(x) \geq 0.5 \\ 0 & : \quad f_1(x) < 0.5 \end{cases} \tag{44}$$

$D_2$ can have the same form. The following decision fusion function $D_d$, in conjunction with the functions $D_1$ and $D_2$, will give the same result as the data fusion function $D_f$.

$$D_d(f_1, f_2)(x) = \begin{cases} 0 & : \quad D_1(f_1)(x) = 0 \ \& \ D_2(f_2)(x) = 0 \\ 1 & : \quad D_1(f_1)(x) = 1 \ \& \ D_2(f_2)(x) = 1, \text{ otherwise} \\ 0 & : \quad x < 0.78125 \cdot (min\{x|D_1(f_1)(x) = 1\} + min\{x|D_2(f_2)(x) = 1\}) \\ 1 & : \quad x \geq 0.78125 \cdot (min\{x|D_1(f_1)(x) = 1\} + min\{x|D_2(f_2)(x) = 1\}) \end{cases} \tag{45}$$

While this works for these two specific functions $f_1$ and $f_2$, it will not work for other functions, say $f_1(x) = 0.3x$ and $f_2(x) = 0.7x$. Note, however, that the condition for a decision fusion system to be a data fusion system involves quantification over all functions $f_1,\ f_2$, i.e., it requires that the selection of $D_1,\ D_2,\ D_d$ should give the same result as $D_f$ for all such input functions $f_1,\ f_2$.

## 5 Multi-Source vs. Single-Source

A classification of fusion systems can also be obtained according to their relation to single-source systems. To introduce this classification we need to talk about functions in terms of the sets of ordered tuples (argument-value pairs). More specifically, we will use the following notation:

$$f = \{(x, f(x))|x \in X\} \tag{46}$$

Consequently, $\Delta$ will represent the set $\{(x, \Delta(x))|x \in X\}$, $D_1(f_1)$ will represent the set $\{(x, D_1(f_1)(x))|x \in X\}$, and so on.

We will explain our classification using an example shown in Figure 8. The rectangle in this figure represents the Cartesian product of the coordinate space (X) and the decision space (E). The bold line annotated with $\Delta$ represents a ground truth function for one measurement. The decision function $D_1(f_1) \in 2^E$ is represented by two lines: $D_{1l}$ - the lower bound and $D_{1u}$ - the upper bound. The intent here is to show that the values of $D_1(f_1)$ for a given $x$ are between these two lines. Note that since values of $D_1(f_1)$ are subsets of $E$, in this example the values are intervals delimited by the points on the upper bound and the lower bound lines respectively, i.e., by $D_{1l}(x)$ and $D_{1u}(x)$. The functions $D_2(f_2)$ and $D_f(f_1, f_2)$ are represented similarly by two lines (lower and upper).

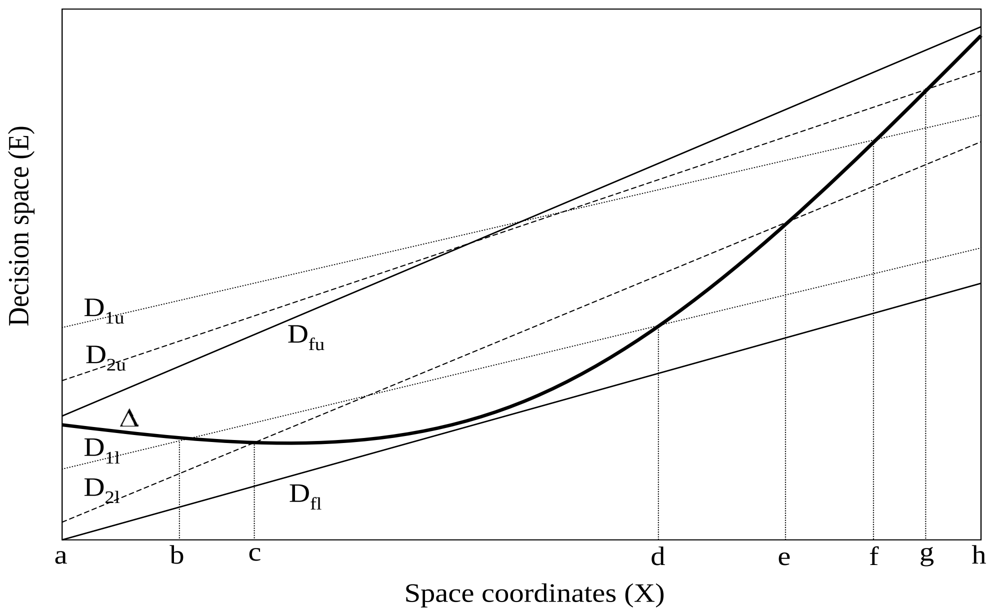

Figure 8: Comparison of Single Decision Systems vs. a Data Fusion System

Note that the single-source systems are not *correct* (see Definition 1), i.e., it is not guaranteed that the ground truth is within the bounds of the decisions of these systems. In mathematical terms, it is not guaranteed that the following relation holds: $\Delta(x) \in D_1(f_1)(x)$. In particular, for $D_1(f_1)$ this condition is satisfied in the intervals $[a, b]$ and $[d, f]$. For $D_2(f_2)$ this condition is satisfied for intervals $[a, c]$ and $[e, g]$. The fusion system, on the other hand, satisfies this condition for all $x$.

In the following we analyze special kinds of information fusion which occur when we add some natural constraints. In this discussion we will not necessarily assume the correctness of either a single-source system or the fusion system.

**Definition 7** *Given two single-source decision systems $S_1,\ S_2$ and a data fusion system $S_f$, the fusion system is called:*

1. *overlapping, if $D_f(f_1, f_2) \subset D_1(f_1) \cap D_2(f_2)$*
2. *inclusive, if $D_f(f_1, f_2) \supset D_1(f_1) \cap D_2(f_2)$*
3. *alternative, if $D_f(f_1, f_2) \subset D_1(f_1) \cup D_2(f_2)$*
4. *preferential, if $D_f(f_1, f_2) \subset D_1(f_1)$*
5. *covering, if $D_f(f_1, f_2) \supset D_1(f_1) \cup D_2(f_2)$*

*for all $f_1,\ f_2$.*

The overlapping fusion system gives the smaller sets as values than any of its components. For instance, if the set $E$ consists of 10 objects and if $S_1$ gives 3 objects for a given $f_1$ and a given $x$, and $S_2$ gives 2 objects, the overlapping system will give at most 2 objects as output. In other words, it will always give a decision that is in agreement with both single-source systems and will never contradict any of the single-source systems when they agree. However, such a system cannot be guaranteed to be complete, i.e., the fusion function $D_f$ is not guaranteed to be global.

The inclusive fusion system will always give at least as large sets as the overlapping system. Again it is not guaranteed to be complete. It will include those decisions on which both systems agree, and possibly more.

The alternative system will give only such decisions that agree with at least one of the single-source systems. Again, this system is not guaranteed to cover the whole coordinate space.

The preferential system will give decisions that are always in agreement with the preferred system. If for the decisions of $S_1$ and $S_2$ do not overlap, the resulting decision set will be overlapped with the set $D_1(f_1)(x)$ and will have no common part with the set $D_2(f_2)(x)$. This does not mean that $S_2$'s input will be ignored. It may be taken into account, but only to the extent that the result does not violate the preference condition.

And finally the covering system will include all of the decisions of the single-source systems and possibly more. This system gives less sharp decisions since the output sets are larger than the output sets of the components. This system is guaranteed to be complete, although still not necessarily correct.

## 6 Measure of Effectiveness

While correctness seems like a natural property to require, it is not sufficient for guiding designers of fusion systems. Note, for instance, that a system that gives the output $E$ (i.e., all possible values) for all $f$ and for all $x$ would be correct, but not very useful. This kind of an answer would mean "I don't know" in every case. To be able to assess various fusion systems we need to have a quantitative measure of performance of such systems. While various measures can be proposed, we give just one example to make our presentation of formalization of fusion relatively complete. The measure captures the closeness to the ground truth and thus can be thought of as a quantitative measure of correctness.

Consider two single-source systems $S_1$, $S_2$ and a fusions system $S_f$. Assume existence of a measure

$$\mu : L \times E \to [0, 1] \tag{47}$$

We can extend specifications of $S_1$, $S_2$ and a fused system $S_f$ by adding the measure $\mu$ to these specifications. We then can define the following measure of effectiveness of a decision $D_1$:

$$\epsilon(D_1(f_1)) = \frac{\mu(\Delta \cap D_1(f_1))}{\mu(D_1(f_1))} \tag{48}$$

The effectiveness measure for $D_2$ takes the same form. For the fusion system the effectiveness measure is:

$$\epsilon(D_f(f_1, f_2)) = \frac{\mu(\Delta \cap D_f(f_1, f_2))}{\mu(D_f(f_1, f_2))} \tag{49}$$

It is a measure of the intersection of the decision function with the ground truth function, relative to the decision function. By definition, the measure of effectiveness would be a number between 0 and 1, which equals 1 when $D_f(f_1, f_2) \subset \Delta$, provided that the set $D_f(f_1, f_2))$ is not of measure zero.

Having defined a measure of effectiveness, we can prove various theorems about the performance of various classes of systems in terms of the measure of effectiveness. Below is a theorem about overlapping fusion systems.

**Theorem 2** *Let $S_1, S_2$ be correct single-source decision systems, and $S_f$ a correct overlapping fusion system that includes $S_1$ and $S_2$. Then the fusion system is at least as effective as any of its parts.*

**Proof:**

We need to show that $\epsilon(D_1(f_1)) \leq \epsilon(D_f(f_1, f_2))$ for all $f_1,\ f_2$. More specifically, we need to show that

$$\frac{\mu(\Delta \cap D_1(f_1))}{\mu(D_1(f_1))} \leq \frac{\mu(\Delta \cap D_f(f_1, f_2))}{\mu(D_f(f_1, f_2))} \tag{50}$$

Recall that (Definition 7) for an overlapping system $D_f(f_1, f_2) \subset D_1(f_1) \cap D_2(f_2)$. This means that the left-hand denominator in Eq. 50 is greater than the right-hand. Since the numerators are the same (because both $S_1$ and $S_f$ are correct systems) then the conclusion follows. ∎

Notice that the effectiveness of this kind of fusion system increases with the set $D_1(f_1) \cap D_2(f_2)$ becoming smaller. This means that such a fusion strategy has the "narrowing" effect on the decision set. However, to prove a similar result for systems that are not necessarily in the class of correct systems we would need to make additional independence assumptions. Many more theorems could be proved for the classes of systems in Definition 7, but this is beyond the scope of this paper. Since the goal of this paper is to provide a formalization of fusion, our intention was simply to show examples of uses of the formalization.

The measure of effectiveness $\epsilon$ gives an assessment of the quality of a given decision. This measure then could be used for defining a measure of effectiveness of the fusion system. Consequently, the signature of the fusion functions $D_f$ and $D_d$ could be changed by adding the measure of effectiveness of either each decision, or of the system. Below we show a case when each decision carries a value of the measure of effectiveness:

$$D_f : (X_1 \to V_1) \times (X_2 \to V_2) \to (X \to E \times [0,1]) \tag{51}$$

In this case the value of the measure of effectiveness of each decision is within a unit interval, i.e., it could be a probability of correct decision.

## 7 Data Association

Finally, we need to discuss the issue of *data association*, which is at the core of any fusion problem. We discuss this problem using the diagram of Figure 9. This diagram, similarly as all previous diagrams, shows how to unify the distinguished sorts. It also shows how to unify other sorts and operations. Once sorts are mapped by appropriate morphisms (similarly as in Eq. 12), the mapping of values of particular sorts is uniquely defined. I.e., if we unify two sorts both of which are isomorphic with say natural numbers, then the number "5" in one sort must be mapped to "5" in the other sort. However, "5" in one coordinate system may correspond to "36" in the world coordinate system. Consequently, data association must map "5" to "36". To achieve this goal, we add an additional specification $S_h$, which is a definitional extension of $S_f$. The association is then done through the transformation of coordinates:

$$x \rightsquigarrow x_1 \equiv x_1(x) \tag{52}$$

$$x \rightsquigarrow x_2 \equiv x_2(x) \tag{53}$$

The specification $S_h$ imports $S_f$ and adds four additional functions: $x_1$ and $x_2$, as described in the above equation, and $f_1'$ and $f_2'$, defined by the following equations:

$$f_1'(x) = f_1(x_1(x)) \tag{54}$$

$$f_2'(x) = f_2(x_2(x)) \tag{55}$$

The functions $f_1'$, $f_2'$ play the role of $f_1$ and $f_2$ from the initial specification.

For an example of data association, consider the two sensors from Section 2. The points of origin of the coordinates $(x_{10}, y_{10}) = (0,0)$ and $(x_{20}, y_{20}) = (0,0)$ for the two sensors must

$$\begin{tikzcd}
 & S_h & \\
 & S_f \arrow[u] & \\
S_1 \arrow[ur] & S_w \arrow[u] & S_2 \arrow[ul] \\
 & S_c \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 9: Data Association

be aligned with the world coordinate system, say

$$(x_{10}, y_{10}) = (a_1, b_1) \tag{56}$$

$$(x_{20}, y_{20}) = (a_2, b_2) \tag{57}$$

Assuming that the coordinates are simply shifted, the coordinate mapping functions then are:

$$(x_1(x), y_1(y)) = (x - a_1, y - b_1) \tag{58}$$

$$(x_2(x), y_2(y)) = (x - a_2, y - b_2) \tag{59}$$

The new measurement functions then are:

$$f_1'(x, y) = f_1(x - a_1, y - b_1) \tag{60}$$

$$f_2'(x, y) = f_2(x - a_2, y - b_2) \tag{61}$$

For instance, if the measurement function for the first sensor is $f_1(x_1, y_1) = 2x_1 + 3y_1$, then the new function is $f_1'(x, y) = 2(x-a_1)+3(y-b_1)$. A similar transformation would take place for the second sensor. Consequently, after the association, the measurements of both sensors are aligned with the world coordinates. Thus the formalism of specifications presented in this paper provides a natural framework for data association.

## 8 Other Conceptualizations of Information Fusion

Various aspects of fusion have been addressed in the literature under the names of *data fusion*, *sensor fusion*, *sensor integration* and *information fusion*. It seems rather clear that the framework presented in this paper is more abstract than other frameworks for fusion known in the literature. Consequently, it should be possible to capture the aspects of other conceptualizations of fusion within our framework. A mapping of each of the approaches is beyond the scope of this paper. Below we just outline the relation of our approach to some of the conceptualizations known in the fusion community, without trying to be complete.

The most influential conceptualization and classification of fusion systems was provided by the JDL Model [5]. The JDL model classifies fusion systems with respect to the data that they take as input and the outputs they generate. For instance, Level 1 fusion inputs raw sensory data and generates either object IDs or object states. Level 2, on the other hand takes object information from Level 1 and derives relations among the objects. This classification is especially tuned to the military domain, although it is also used in the commercial applications as well. Although particular classes of algorithms are suggested for particular levels (see for instance a classification provided in [27]), the assignment is rather loose. It is often stated (cf. [28]) that the algorithms are standard algorithms for computing uncertainty associated with particular decisions and not specific to fusion.

Our formalization presented in this paper differs from this approach in many respects. For one, our formalization focuses on the processing rather than on data, although data are included in the formalization as well. Note that our formalization views fusion as a function together with its inputs and outputs. It also captures the relationships among the functions, for instance it captures the process of constructing a fusion function out of component functions. Finally, our formalization does not contradict the JDL model in any way. Using our approach, one can formalize the fusion function of each of the levels in the JDL model. However, since such a formalization would have to specify the classes of functions used for fusion on particular levels of the JDL model, it would be an extension to the JDL model. While this would be a very interesting and challenging work, it is beyond the scope of this paper.

Another classification of fusion was proposed by Dasarathy (cf. [19]). First of all, he proposed to view fusion systems in terms of what in software engineering terms can be viewed as *data flows*. Data flows can be characterized by inputs, outputs, and processes (functions). In this paper we made this fact more explicit by specifying the sensors and the processes as functions. Similarly as in [19] we distinguished data fusion and decision fusion. The difference is that we showed decision fusion within the context of a complete sensor information processing module, rather than showing it by itself. In our formalization of data fusion we showed the output to be in the set of decisions, $E$. But nothing in our formalization prevents one from interpreting $E$ to be a (fused) data space. Our Definition 7 was also influenced by [19]. For instance, our overlapping fusion is a case of what in [19] is called "AND" fusion, our preferential fusion is similar to the serial sensor suite.

Most of the literature on fusion deals with various approaches to deriving the uncertainty of decisions. However the algorithms are standard algorithms of probability and statistics, fuzzy logic, Dempster-Shafer possibilistic reasoning or neural nets. The aspects specific to fusion are in the *schemes* of processing rather than in the algorithms themselves. In other words, it is the arrangement of algorithms into a *processing architecture* that distinguishes fusion from other data analysis systems. Examples of such schemes are presented in [29, 30, 31, 32, 33, 34, 35] and many others. All of the architectures presented in these papers and books could be formally specified using our formal approach. The benefit of such an exercise would be an ability to perform reasoning about the algorithms using automatic computer-based tools (theorem provers).

A more general conceptualization of fusion, based on the notion of *random set*, was presented in [36]. The distinguishing feature of this approach is that the uncertainty of decisions is combined with the decisions themselves by the means of set-valued functions. This conceptualization then allows one to view various kinds of uncertainty as special cases of the generic scheme. While we did not go into any of the details of algorithms that use this kind of approach, the influence of this conceptualization on our approach can be seen in the fact that our general model starts with the assumption that the result of fusion is a set-valued function (see, for instance, Eq. 22). Our examples, on the other hand, use only object-valued functions. The framework of category theory used in our approach if flexible enough to capture the formalism proposed in [36].

## 9 Conclusions

This paper presents an approach to the formalization of information fusion. The approach is general enough to capture all kinds of fusion, including data fusion, feature fusion, decision fusion and fusion of relational information. The paper gives only a taste of what formalization would look like rather than giving a complete formalization of any kind of a fusion system.

We envision two kinds of advantage of such a formalization. First of all, the formalization of information fusion presented in this paper can be viewed as a first step in developing a formal theory of fusion. Using this approach, the scientist working in the fusion domain can specify various fusion concepts in a clear and unambiguous language so that other scientists can interpret the concept in a unique way. Moreover, the scientist can provide formal proofs of the features of the proposed new fusion related concepts so that other scientists can verify the proofs.

It is our hope that the development of a theory of fusion would be followed by the development of tools that the developer of fusion systems could use. A formal framework for developing fusion systems would allow the designer to first formally specify algorithms in a formal language and then follow the formal method approach to synthesizing and analyzing a fusion system. This expectation strongly depends on the popularity of formal methods in software development.

Everybody in the fusion community seems to agree that fusion is a *process* that accepts some data (from multiple sources) as input and produces some outputs (decisions). However, in the fusion literature, only the input/output data are used to specify various fusion systems, while the processing part is treated as a second-class concept. The main contribution of this paper is that all aspects of multi-source information processing, i.e., both data and processing, are captured in this formalization. And even more, the processing elements (algorithms) can be combined in a consistent way. The concept of *subclass* introduced in this paper allows for comparison of various fusion systems and for proving that one system is a special case of another.

In conclusion, we would like to stress that we do not view this formalization as the only one and correct formalization of information fusion. Others may have different views of information fusion and thus their formalizations might be different than this one. The contribution of this paper is that it gives a precise formal statement of one view of information fusion. Everybody can analyze this conceptualization and present findings in a precise mathematical language.

The directions for future work have been mentioned in the paper in various places. For one, as mentioned above, one could use this framework to formalize and extend the JDL model of data fusion. In particular, this framework would allow one to formalize processes of fusion, in addition to data. Various classifications of information fusion processes could be formally specified and then used in the process of formal development of information fusion systems. Relations among particular classes of processes could be formulated as theorems, which then would have to be formally proved.

## Acknowledgments

This research was partially supported by a grant from the Air Force Office of Scientific Research under contract No: F49620-98-1-0043. The authors wish to express their thanks to all those who provided feedback on their formalization of fusion, especially to the members of the research team: Hongge Gao, Jingsong Li and Marek Malczewski. We would also like to acknowledge the comments of the anonymous reviewers who made suggestions on how to improve the readability of this paper.

## References

[1] R. A. Kemmerer. Integrating formal methods into the development process. *IEEE Software*, 9:37–50, 1990.

[2] Formal methods specification and verification guidebook for software and computer systems. Technical Report NASA-GB-002-95, National Aeronautics and Space Administration, 1995.

[3] Martin D. Fraser, Kuldeep Kumar, and Vijay K. Vaishnavi. Strategies for incorporating formal specifications. *Communications of the ACM*, 37, No.10:74–85, October 1994.

[4] J. M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 9:8–24, 1990.

[5] A. N. Steinberg, C. L. Bowman, and F. E. White. Revisions to the JDL data fusion model. In *The Joint NATO/IRIS Conference*, 1998.

[6] M. M. Kokar and Z. Korona. A formal approach to the design of feature-based multi-sensor recognition systems. *International Journal of Information Fusion*, 2 (2):77–89, 2001.

[7] T. M. Schuck, M. Friesel, and J. B. Hunter. Information properties as a means to define decision fusion methodologies in non-benign environments. In *Proceedings of Fusion'2003, 6-th International Conference on Information Fusion*, pages 479–484, 2003.

[8] C. C. Chang and H. J. Keisler. *Model Theory*. North Holland, Amsterdam, New York, Oxford, Tokyo, 1992.

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

[10] Specware: Language manual, version 2.0.3. Technical report, Kestrel Institute, 1998.

[11] B. V. Dasarathy. Sensor fusion potential exploitation - innovative architectures and illustrative applications. *Proceedings of IEEE*, 85, No.1:24–38, 1997.

[12] P. K. Varshney. *Distributed Detection and Data Fusion*. Springer-Verlag, 1996.

[13] Y. V. Srinivas. Category theory: Definitions and examples. Technical Report TR-90-14, University of California at Irvine, 1990.

[14] Specware: User guide, version 2.0.3. Technical report, Kestrel Institute, 1998.

[15] D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing*, 9:229–269, 1997.

[16] S. A. DeLoach and M. M. Kokar. Category theory approach to fusion of wavelet-based features. In *Proceedings of the Second International Conference on Information Fusion, Vol. 1*, pages 117–124, 1999.

[17] M. M. Kokar, J. A. Tomasik, and J. Weyman. A formal approach to information fusion. In *Proceedings of the Second International Conference on Information Fusion, Vol. 1*, pages 133–140, 1999.

[18] M. M. Kokar, J. A. Tomasik, and J. Weyman. Data vs. decision fusion in the category theory framework. In *Proceedings of FUSION 2001 - 4th International Conference on Information Fusion, Vol. 1*, pages TuA3–15 – TuA3–20, 2001.

[19] B. V. Dasarathy. *Decision Fusion.* IEEE Computer Society Press, 1994.

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

[21] E. R. Dougherty and C. R. Giardina. *Matrix Structured Image Processing.* Prentice-Hall, 1987.

[22] G. Booch, I. Jacobsen, and J. Rumbaugh. *OMG Unified Modeling Language Specification*, March 2000. Available at `www.omg.org/technology/documents/formal/-unified_modeling_language.htm`.

[23] DAML: DARPA Agent Markup Language Web Site, 2001. `www.daml.org`.

[24] J. Hendler and D. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15, No. 6:67–73, 2000.

[25] DAML+OIL, March 2001. `www.daml.org/2001/03/daml+oil-index.html`.

[26] OWL Web Ontology Language Reference, 2003. `www.w3.org/TR/owl-ref`.

[27] D. L. Hall. *Mathematical Techniques in Multisensor Data Fusion.* Artech House, Boston - London, 1992.

[28] D. L. Hall and J. Llinas. An introduction to multisensor data fusion. *IEEE Transactions*, 85, No. 1:6–23, 1997.

[29] J. K. Aggarwal. *Multisensor Fusion for Computer Vision.* Springer-Verlag, 1993.

[30] J. J. Clark and A. L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publisher, Boston, 1990.

[31] L. A. Klein. *Sensor and Data Fusion Concepts and Applications.* SPIE, Bellingham, WA, 1993.

[32] R. Krzysztofowicz and D. Long. Fusion of detection probabilities and comparison of multisensor systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 20 No.3:665–677, 1990.

[33] S. C. A. Thomopoulos. Sensor integration and data fusion. In *Sensor Fusion II: Human and Machine Strategies*, pages 178–191. SPIE, 1989.

[34] S. C. A. Thomopoulos. Sensor integration and data fusion. *Journal of Robotic Systems*, 7(3):337–372, 1989.

[35] E. Waltz and J. Llinas. *Multisensor Data Fusion.* Artech House, Norwood, MA, 1990.

[36] I. R. Goodman, P. S. Mahler, and H. T. Nguyen. *Mathematics of Data Fusion.* Kluwer Academic Publishers, 1997.