An Approach to Automation of Fusion Using Specware

Hongge Gao
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, U.S.A.
hgao@ece.neu.edu

Mieczyslaw M. Kokar
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, U.S.A.
kokar@coe.neu.edu

Jerzy Weyman
Department of Mathematics
Northeastern University
Boston, MA 02115, U.S.A.
weyman@neu.edu

Abstract *This paper introduces an approach to the automation of fusion using category theory based formal method. Category theory has a rich and rigorous mathematical language to manipulate complex systems via the relations between various kinds of objects. Specware, a category theory based formal development system, is used as a platform. This approach has the following advantages. First of all, fusion systems designed using this approach are easy to reuse, extend and maintain under evolution. Secondly, it provides a formal support to represent and state human knowledge explicitly. Finally, with the support of Specware, we can refine the formal specification into final executable code by stepwise refinement. Specware guarantees that the final executable code is provably correct.*

Keywords: information fusion, formal method, category theory.

1 Introduction

A number of information fusion architectures, models and techniques have been proposed, but there are few systematic approaches to representing, implementing and maintaining fusion systems. For instance, it is not possible to guarantee that a system designed using specific architecture actually implements the architecture and its requirements. It is also hard to reuse, extend or evolve such systems.

To deal with these kinds of issues, we use a formal method approach to the development of fusion systems. In our approach, we follow a software engineering paradigm, i.e., we first specify requirements for a fusion system, and then we develop code through progressive refinement of specifications. Our approach has the following advantages. First of all, fusion systems designed using this approach are easy to reuse, expand and manage under changes. Secondly, it enables us to represent and state human knowledge explicitly in the specification. Finally, with the support of Specware, we can refine the formal specification into final executable code by stepwise refinement. Specware guarantee that the final executable code is provably correct.

The main problem that we are addressing in this paper is how to guide the process of fusion of specifications into a final specification of the system. In our approach, we use Specware, a formal method tool that is based on category theory. Since category theory provides us with the rigorous mathematical language and rich operations to represent and manipulate complex information structures, we can assemble fusion system specifications modularly and incrementally by using category theory operators, such as colimit and interpretation, to the particular basic specifications.

While Specware provides a formal specification language, the specification developer has to decide which specifications to combine and how. Our goal is to automate this process. Towards this goal, we investigated the Planware approach [2] to developing specifications. Planware is a process developed at Kestrel Institute for the domain of scheduling. We are investigating a similar approach to developing fusion systems. Basically, our process is as follows.

First, we develop a library of formal specifications of various goals, sensor theories, background theories and fusion theories. The relations among these theories are represented by specification morphisms.

Second, we assemble an abstract specification of a fusion system from the library developed in the first step. Fusion is then considered as an operation of combining those various specifications into a specification of a fusion system. In other words, fusion is an operation on these specifications. This differs from other views of fusion, where it is considered as an operation on data or decisions.

Third, we refine the abstract specification into a concrete specification using the information provided by the user. For any individual specification, we refine it to a more concrete specification via sequential composition of interpretations. For structured specification, we use parallel composition operator to automatically construct the refinement.

Finally, we generate code for the concrete specification.

The rest of the paper will explain how to implement the above procedure. Section 2 provides a brief introduction to category theory and Specware. In Section 3, we describe our approach to automation of fusion using Specware. A specific multisensor fusion example will be given in Section 4 followed by summary in Section 5.

2 Background

Category theory was originally invented as an abstract mathematical language to describe the passage from one type of mathematical structure to another. Specware supports the modular construction of formal specifications. It also supports stepwise and componentwise refinement of structured specification into executable code.

2.1 Category Theory

Category theory is an abstract language for describing external properties of objects. In category theory, an object is described by its interaction with all other objects via *morphisms*. This unique feature of abstract, high-level description makes category theory an ideal mathematical tool for the information fusion problem. In information fusion, we need to know the *relations* or *interactions* between disparate sources (*information*) in order to combine them together (*fusion*).

A good review of category theory related to fusion can be found in [3]. Interested reader can find more information about category theory in [8, 5, 1].

2.2 Specware

In this section, we will introduce Specware concepts which we used to automate the fusion process.

Specware is a system which aims to provide a formal support for specification and development of software [9]. The foundations of Specware are category theory, sheaf theory, algebraic specification and general logics. Using Specware, one can construct formal specifications modularly and refine such specifications into executable code through progressive refinement. The underlying basic concepts of Specware are described below.

A specification (*spec* or *theory*) is a collection of sorts, operations and axioms that define a theory via higher-order logic. An example of specification of image is shown in Figure 1.

```
spec IMAGE is
    sorts Image, E
    op make-image : Nat, Nat, E → Image
    op xsize : Image → Nat
    definition of xsize is
        axiom xsize(make-image(m,n,e)) = m
    end-definition

    other operations and axioms ...

end-spec
```

Figure 1: Image specification

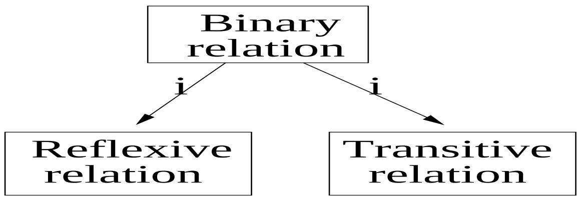

Figure 2: A specification diagram

Specifications can be developed from scratch or can be constructed from other specs via the specification-constructing operations – *import*, *translate* and *colimit*. Spec A has a copy of spec B if A imports B. Translate is similar to import except some elements of the copy of spec B are renamed according to the given renaming rules. The colimit operation takes a *specification diagram* as input and produces a specification called the colimit of the diagram.

A *specification morphism* is a mapping from *source specification* S to *target specification* T such that the signatures of the operations are translated compatibly and theorems are preserved.

A *specification diagram* (or simply *diagram*) is a directed multigraph whose nodes are labeled with specs and whose arcs are labeled with morphisms. So a diagram shows the relations between specifications. A diagram example is shown in Figure 2. In this diagram, both reflexive relation spec and transitive relation spec import binary relation spec. Therefore the morphisms are import-morphisms.

The definition of *interpretation* is as follows [10]: An *interpretation* $\rho : A \Rightarrow B$ from a specification A (called *domain* or *source*) to a specification B (called *codomain* or *target*) is a pair of morphisms $A \to A - as - B \leftarrow B$ with common codomain $A - as - B$ (called *mediating* specification or simply *mediator*), such that the morphism from B to $A - as - B$ is a definitional extension. Interpretation is also called *refinement.*

A morphism $S \to T$ is a *strict definitional extension* if it is injective and if every element of T which is outside the image of the morphism is either a defined sort or a defined operation. A *definitional extension* is a strict definitional extension optionally composed with a specification isomorphism.

Sequential (Vertical) composition of interpretations allows us to connect interpretations together so that we can refine a specification progressively. If ρ_1 and ρ_2 are two interpretations such that $\rho_1 : S \Rightarrow R$ and $\rho_2 : R \Rightarrow T$ then their sequential composition $\rho_1;\rho_2$ is an interpretation from S to T. That is, $\rho_1;\rho_2 : S \Rightarrow T$.

Parallel composition allows us to put interpretations together like specification-constructing operations allow us to put specifications together. Suppose we have interpretations for each of the specifications in a given diagram, we can compose them to obtain an interpretation whose domain is their colimit. The codomain of the composed interpretation will be the colimit of a diagram whose nodes are codomain of the component specification interpretations.

All the above concepts are expressed and implemented in *Slang*, Specware language. The specification example in Figure 1 is written in Slang.

3 Automation of Fusion

In the last section, we reviewed the basic concepts of Specware. Next, we are going to show our approach to automation of information fusion using Specware.

3.1 Information Fusion Problem

Basically, *information fusion* or *fusion* is defined as *the process of acquisition, filtering, correlation and integration of relevant information from various sources, like sensors, databases, knowledge bases and humans, into one representational format that is appropriate for deriving decisions regarding the interpretation of the information, system goals (like recognition, tracking or situation assessment), sensor management, or system control* [6]. A typical multisensor fusion scenario is depicted in Figure 3. In the figure, N sensors observe the region of interest, *World*, and send information to the fusion system. *Human* sends *queries* or *goals* to the fusion system. Based on the information received from sensors and queries or goals from the human, the fusion system computes the *solution* and returns an answer to the human (in the situations such as *detection*, *automatic target recognition*) or sends instructions to sensors (in the situation of *sensor management*).

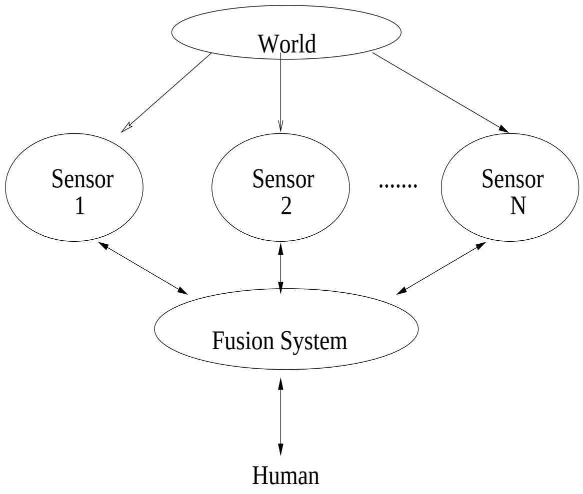

Figure 3: A multisensor fusion scenario

3.2 Abstract Fusion Specification

Based on the analysis of the fusion problem, we represent the abstract fusion problem as a structured specification as shown in Figure 4. Without loss of generality, we use two image sensors as an example. Fusion systems with more than two sensors or different types of sensors have similar structures.

Basically, there are two subdiagrams in this structure. The first subdiagram, COMB-SPECS-DIAGRAM, consists of specs IMAGE, SENSOR, SENSOR1, SENSOR2, ABS-PROB-THY 1 and ABS-PROB-THY 2 where ABS-PROB-THY 1 and ABS-PROB-THY 2 represent fusion problems, such as *detection*, expressed in terms of respective sensor theories. Specification COMB-THY is the colimit of this diagram. The second subdiagram, FUSION-DIAGRAM, consists of specs ABS-PROB, ABS-FUSION-THY and COMB-THY. ABS-REQ-THY, the abstract requirement specification of the multisensor fusion system, is the colimit of FUSION-DIAGRAM. Here we use ABS-PROB to glue ABS-FUSION-THY and COMB-THY together to get the final ABS-REQ-THY.

The above structure provides us with following advantages:

- Represent and state human knowledge explicitly in the specification. For instance, sensor theories, fusion theories are represented by specs SENSOR and ABS-FUSION-THY. Other human knowledge, from geometry to statistics, can also be represented as specifications.

- Reuse, extend and maintain fusion systems relatively easily. The structured specification gives us a clear roadmap of the whole fusion system. The relations between different parts are clearly represented by specification morphisms. This fusion system can be used repeatedly for a class of problems. Also building a larger

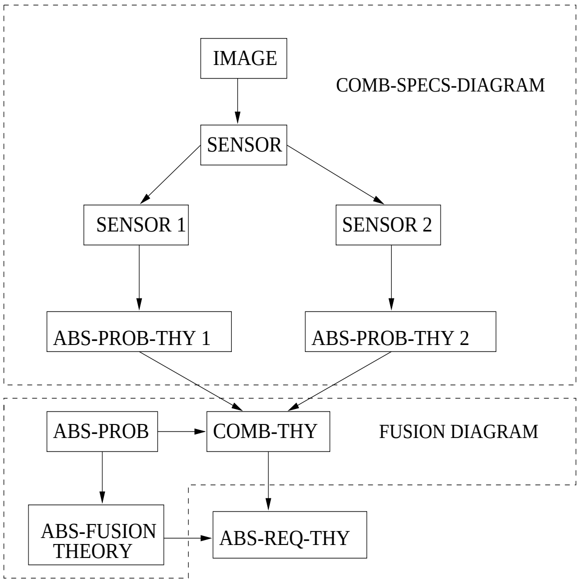

Figure 4: Abstract Fusion Specification Structure

system from this simple one is fairly easy.

- Refine the formal specification into final executable code by stepwise refinement with the support of Specware. As we discussed in Section 2, Specware supports both sequential and parallel compositions of refinement. Once we have such a structured specification, we can refine it incrementally into a *sufficiently refined* specification using sequential and parallel composition. The sufficiently refined specification is such a specification that every sort and operation of it are represented by the built-in *abstract target language* (ATL) [7]. ATL describes the constructs of the target language. Currently, Specware supports two kinds of target language, *C++* and *Lisp*.

Some of the component specifications are shown in Figure 5. Here we modeled the abstract sensor as a function. Similarly, both problem theories(ABS-PROB-THY1 and ABS-PROB-THY2) are also represented as functions. These theories will be refined into the concrete problem theories later on via user's selection. Finally, the fusion theory is represented as a function from outputs of two problem theories (*Q1* and *Q2*) to the final output *Q*. Notice we didn't specify Q1, Q2 and Q at this moment because they could be *Nat*, *Boolean* or any other sort in the refinement.

```
spec SENSOR is
    import IMAGE
    sort Sensor
    op sense : Image, Sensor → Image
end-spec

spec ABS-PROB-THY1 is
    import SENSOR1
    sort Q1
    op p1: Image1, Sensor1 → Q1
end-spec

spec ABS-PROB-THY2 is
    import SENSOR2
    sort Q2
    op p2: Image2, Sensor2 → Q2
end-spec

spec ABS-PROB is
    sorts Image1, Image2, Sensor1,
        Sensor2, Q1, Q2
    op p1 : Image1, Sensor1 → Q1
    op p2 : Image2, Sensor2 → Q2
end-spec

spec ABS-FUSION-THY is
    import ABS-PROB
    sort Q
    op fuse : Q1, Q2 → Q
end-spec
```

Figure 5: Fusion Specifications

3.3 Refining to a Domain-specific Specification

The user can refine the abstract fusion specification into a domain-specific specification by choosing concrete domain theories from the knowledge base. The basic procedure for refinement is as follows:

- First, choose two image sensors as SENSOR1 and SENSOR2 from a library of sensor theories.
- Second, choose a concrete fusion problem from a hierarchy of fusion problems. We will discuss details of this step in the next section.
- Then, choose a corresponding fusion theory.
- Finally, after refining each component theory of abstract specification to its corresponding concrete theory, compute the final domain-specific requirement theory via parallel composition of interpretations. The final domain-specific requirement theory is the refinement of ABS-REQ-THY.

we have analyzed the information fusion problem and introduced a Specware based approach to constructing fusion specification and refining it into final code. We showed that a formal system can be represented as a structured specification and one can develop such a fusion system formally through sequential and parallel composition of refinement.

4 A Fusion Example

In this section, we will show how to apply the refinement procedure described in last section to a particular fusion problem.

4.1 Subdomains of Fusion Problem

Goodman [4] described subdomains of data fusion as follows:

- *Sensor fusion.* In this kind of fusion, evidence from two or more sensors of similar type is combined in order to get more precise information which can not be deduced from each piece of evidence alone.
- *Multisource integration.* This type of fusion includes *Detection*, *Classification*(or *Automatic target recognition*), *Tracking* and *Correlation*.
- *Sensor management.* This refers to the process of adaptively allocating the dwells of each re-allocatable member of a suite of sensors.
- *Situation/threat assessment.* This is to provide an overall picture of the military significance of the data collected by the previous two kinds of fusion.
- *Response management.* This is the process of deciding upon courses of action which are appropriate response to current and evolving military situations.

Based on the above information, we can draw a hierarchy of fusion problems as below (Figure 6).

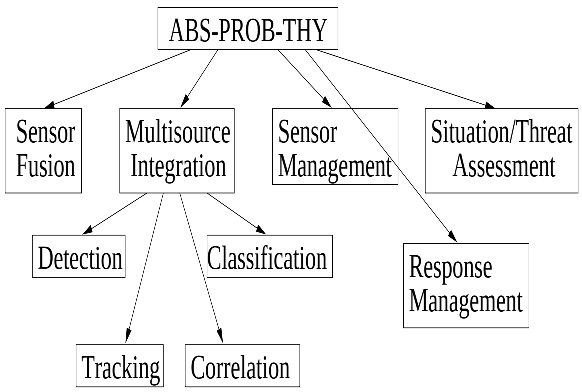

Figure 6: Hierarchy of fusion problems

Next, we will show how to refine an abstract fusion specification to a concrete specification based on this hierarchy.

```
spec DETECTION-THY is
    import LABELING
    op target? : Image, Sensor → Boolean
    definition of target? is
        axiom target?(img,s) ⇔
              gt(max-lab(img,s), zero)
    end-definition
end-spec
```

Figure 7: Detection Theory

4.2 Refining to a Detection Subdomain

To particularize the abstract fusion specification, the user has to select a concrete fusion subproblem from the hierarchy of fusion problem. The (composed) arrow from ABS-PROB-THY to the selected problem theory is the arrow used for refinement.

Suppose the user has chosen a detection theory (Figure 7). It imports LABELING theory which contains operation *max-lab*. The max-lab returns the maximum number of labels of the image being detected.

Then part of the refinement arrow is:

$$Q1 \mapsto Boolean$$

$$p1 \mapsto target?$$

$$Sensor1 \mapsto Sensor$$

Next, the user has to choose a corresponding fusion theory. In this case, the user should select DETECTION-FUSION-THY (see Figure 8).

So the refinement arrow is:

$$p1 \mapsto d1$$

$$p2 \mapsto d2$$

$$Q1 \mapsto Boolean$$

$$Q2 \mapsto Boolean$$

$$Q \mapsto Boolean$$

$$fuse \mapsto final-decision$$

```
spec DETECTION-FUSION-THY is
    sorts Image1, Image2, Sensor1, Sensor2
    const confidence1 : Nat
    const confidence2 : Nat
    op d1 : Image1, Sensor1 → Boolean
    op d2 : Image2, Sensor2 → Boolean
    op final-decision :
        Image1, Image2, Sensor1, Sensor2 → Boolean
    definition of final-decision is
        axiom d1(i1,s1) = d2(i2,s2) ⇒
            final-decision(i1,i2,s1,s2) = d1(i1,s1)
        axiom not(d1(i1,s1) = d2(i2,s2)) ∧
            gt(confidence1, confidence2) ⇒
            final-decision(i1,i2,s1,s2) = d1(i1,s1)
        ...
    end-definition
end-spec
```

Figure 8: Detection Fusion Theory

After having refined each component of the abstract fusion specification, the multisensor detection specification can be computed as described in the last section.

This section has shown how to refine the abstract fusion specification to a particular fusion problem theory. We have chosen a simple example and artificial theories to make the process clear. It needs careful, hard work to develop specifications for real applications.

5 Summary

We have shown in this paper a first step towards automation of information fusion using category theory based formal method. Specifically, we discussed the construction and refinement of fusion specifications using Specware. This approach enables us to represent human knowledge explicitly so that we can utilize this knowledge repeatedly and expand and manage it with ease in a changing environment. This formal approach also provides a way to produce provably-correct code through stepwise refinement.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice-Hall, 1990.

[2] L. Blaine, et. al. *Planware - Domain-Specific Synthesis of High-Performance Schedulers.* Proceedings of the Thirteenth IEEE International Automated Software Engineering Conference, pp. 270-279, Honolulu, Hawaii, October 13-16, 1998.

[3] S. A. DeLoach and M. M. Kokar. *Category Theory Approach to Fusion of Wavelet-Based Features.* To appear in this conference.

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

[5] D. Rydeheard and R. M. Burstall. *Computational Category Theory.* Prentice-Hall, 1988.

[6] W. A. III. Sander. *Information Fusion.* In T. N. Dupuy, F. D. Margiotta, C. Johnson, J. Motley, and D. L. Bongard, editors, *International Military and Defense Encyclopedia*, Vol. 3, G-L, pp. 1259-1265. Brassey's, Inc., 1993.

[7] Kestrel Institute. *SPECWARE LANGUAGE MANUAL.* Specware 2.0.3, March 1998.

[8] Y. V. Srinivas. *Category Theory Definitions and Examples.* Technical Report, Department of Information and Computer Science, University of California, Irvine, February 1990. TR-90-14.

[9] Y. V. Srinivas and J. L. McDonald. *The Architecture of SPECWARE, a Formal Software Development System.* Technical report, Kestrel Institute, 1996.

[10] Y. V. Srinivas and R. Jüllig. *SPECWARE: Formal Support for Composing Software.* Proc. of Conference on Mathematics of Program Construction, Kloster Irsee, Germany, July 1995.