Developing Open Systems Using Theories and Models of the World

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

Zbigniew Korona
Department of Mechanical, Industrial
and Manufacturing Engineering
Northeastern University
Boston, MA 02115

Abstract

This paper considers an *open system* as such that can deal with inputs that were not anticipated by the designer. Using an ATR system as an example, we show how the combination of logic with software engineering techniques allowed us to improve the performance of the system.

1 Introduction

The term *open system* is interpreted in many different ways. We consider here the following three interpretations.

1. A system that has been developed without full specifications, i.e., its functionality includes some mechanisms that should allow it to deal with the unexpected. This definition stresses the unknown (at the design time) inputs that the system may receive during its operation. This definition reflects the understanding of an *intelligent control system* [1].

2. A system that incorporates some adaptation/learning mechanism which allows it to extend its *knowledge*, i.e., to be able to derive the truthfulness of particular statements as a result of its operations; these statements could not be derived initially even if the system was provided with the same input. Also, the truthfullness depends on the context in which the information is interpreted. This characteristic is related to a logical understanding of an intelligent system (e.g., [11]).

3. A system whose interfaces are fully specified and therefore is easily extendible and portable. Both the extendibility and the portability are due to these explicit definitions. This definition stresses the software engineering aspects, i.e., the effort related to extending the system's functionality and to porting it to another computer platform [8].

If we understand an *open system* as one having all three characteristics, then such a system exhibits main characteristic of a semiotic system [7]. The characteristics of a semiotic system include, among others, the following.

1. It must represent information with *signs* and must be able to process information based upon this representation.

2. It must be able to generate new signs and incorporate them into its information processing mechanisms.

3. The meaning of signs is dependent on the context in which the system operates.

The fact that signs are part of a system representation is compatible with the logical approach to intelligent systems and thus to the second characteristic of an open system, where the basic representational element of knowledge is that of a *symbol*, which is just a special case of a sign. Adding new signs to the syntactic structure of a formal logical system is not difficult. But associating meaning with new signs in a consistent way and transforming syntactic representations while preserving the meaning is a difficult problem. The issues of context dependent meaning is addressed in model theory [3]. The transformations that preserve meaning are investigated in formal software specification (e.g., [10]).

The requirement to generate new signs and incorporate the new signs into the system's structure is related to both the intelligent control view of an intelligent system, where the new signs are generated in response to new unexpected situations, and to the logical view of an open system. The idea of a learning agent described in [11] addresses both the issue of new knowledge and the dependence of meaning on context.

The software engineering aspect of open systems is related to the *pragmatics* part of semiotics. If a semiotic system is to be an intelligent system, it must be implemented as a computer system. For this, the engineering aspect must be addressed in addition to philosophical and logical aspects. Although logic can be considered as a universal mechanism for developing intelligent systems, the problem of undecidability calls for the use of other non-logical methods in order to make the approach technically feasible. In this paper we address the issue of building intelligent systems using a mix of methods from logic and software engineering. We propose an approach where logic constitutes a framework that guides the development of an intelligent system. In addition to logic, a number of techniques that can be utilized in this approach are included.

One of the questions in building intelligent systems is what should be the starting point for building up intelligence. One extreme view could be that the system can be equipped with two reasoning engines, one deductive and one abductive, without any prior knowledge; the knowledge is built up from observations (learning). This direction is not followed in this paper due to the fact that such a process, even if it were possible, would require too much time to achieve any reasonable level of performance. In machine learning this fact is expressed as "you cannot learn anything unless you know almost everything." The other extreme is a system that is fully programmed by software engineers and does not change its structure over its life time. This direction is not followed due to a lack of confidence in the existence of such omniscient programmers.

Instead, we presume that there is prior knowledge about the world in which an intelligent system will operate. In our case, since we are dealing with a logical based system, this knowledge is represented as *theories* and *models* of the world. This knowledge constitutes *components* of a software system. As it has been recognized in software engineering, the knowledge should be *domain specific*. Therefore, we pursue the *domain specific software architectures* paradigm of software engineering (cf. [5]). Additionally, we presume that even with the prior knowledge, humans must be involved in the design of the structure of an intelligent system for a specific domain. Consequently, we show the process of developing an intelligent system as consisting of some software engineering and logical reasoning steps performed off-line, some training steps performed off-line (using a *neural network*), and finally some steps performed by the system on line.

Our view of an intelligent system is that it is a tool that supports humans in achieving their goals. As such, an intelligent system must interact with humans. Since one of the aspects of intelligent systems is the change of their knowledge, and possibly structure, over their life time, humans can be involved in the processes of knowledge acquisition and restructuring of the processing schemes. One of the possible human involvements is in design. Our main goals in this paper is to (1) show that logic can play the role of a framework for building intelligent systems, and (2) show that some of the problems of the applicability of logic can be overcome by combining it with software engineering.

To support our claims we present one of our experiments with such an approach to developing an intelligent system. The system's goal is automatic target recognition (ATR). Our main goal in this experiment was to find out where and how symbolic knowledge about sensors and targets can be incorporated in the system, which typically is built using quantitative statistical methods. Moreover, the goal was to be able to improve the performance of an ATR system due to the incorporation of symbolic knowledge with respect to a purely quantitative system. Towards this goal, we implemented two systems, one without using the symbolic knowledge and one with symbolic knowledge about targets and sensors. We showed how the symbolic knowledge, represented as theories and models, can be combined into one coherent symbolic representation. We showed how this symbolic representation can be implemented efficiently using some of the concepts of model checking (cf. [4]) implemented as a neural network. The results of the comparison are presented in the following sections. The full description of our experiments can be found in [6].

2 A Logic-Based ATR System

In this section we describe an example of a system, Automatic Multisensor Feature-based Recognition System (AMFRS) [6], developed according to the principles described above. We show that this approach, due to the incorporation of symbolic knowledge, gave us the advantage in the accuracy of Automatic Target Recognition (ATR).

2.1 Automatic Target Recognition Scenario

As an example of the target recognition problem, we consider an extended version of the waveform recognition problem, originally examined and applied to a ship recognition project in [2] and then analyzed in [9]. The goal of this problem is to recognize the class to which a given target signature belongs, knowing two reference databases (one for each sensor) of target signatures. The reference databases are synthesized using different combinations of three basic waveforms to form three classes of target signatures. Fig. 1 shows five examples of target signatures for each of the four classes of targets in the first signature database DT_r.

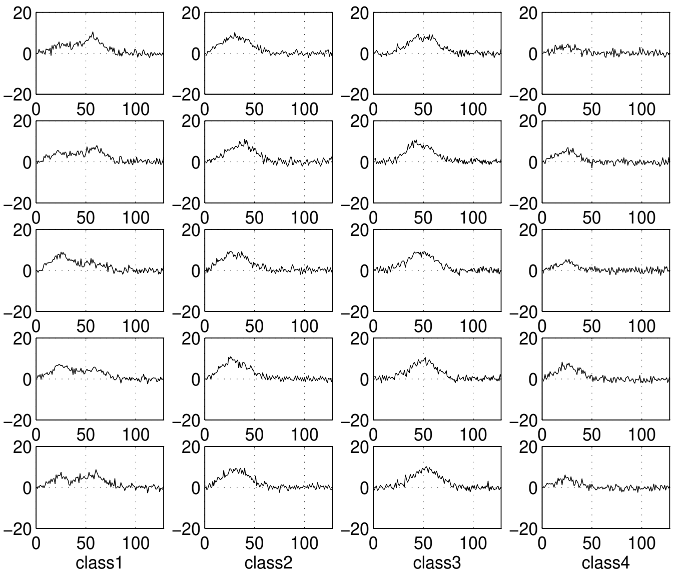

Figure 1: Examples of Target Signatures

2.2 Automatic Multisensor Feature-based Recognition System (AMFRS)

Fig. 2 shows the Automatic Multisensor Feature-based Recognition System (AMFRS) for two sensors.

In this figure, *The World* is a recognition domain with *Sensor1* and *Sensor2* providing information about the domain to the AMFRS. *DWPD* transforms measurements into the wavelet domain. *Feature Selection* selects interpretable features (from the DWPD of sensor data from both sensors) which are the elements of the models M_1 and M_2. The models M_1 and M_2 are the models of the domain theories T_1 and T_2 respectively. *Feature Fusion* fuses features from both sensors into one set of fused features which are the elements of the fused model M_f of the fused theory T_f. *Backpropagation Neural Network* implements target classification based on the fused features.

2.3 Design Methodology for the AMFRS

Fig. 3 shows a block-diagram of the steps involved in the design of an Automatic Multisensor Feature-based Recognition System (AMFRS).

Reference Database 1 and *Reference Database 2* contain target signatures. *DWPD & BDBA* select most discriminant bases for each reference database by using Discrete Wavelet Packet Decomposition (DWPD) and Best Discrimination Basis Algorithm (BDBA). *Most Discriminant Bases (MDB 1 and MDB 2)* are bases with maximum discriminant power with regard

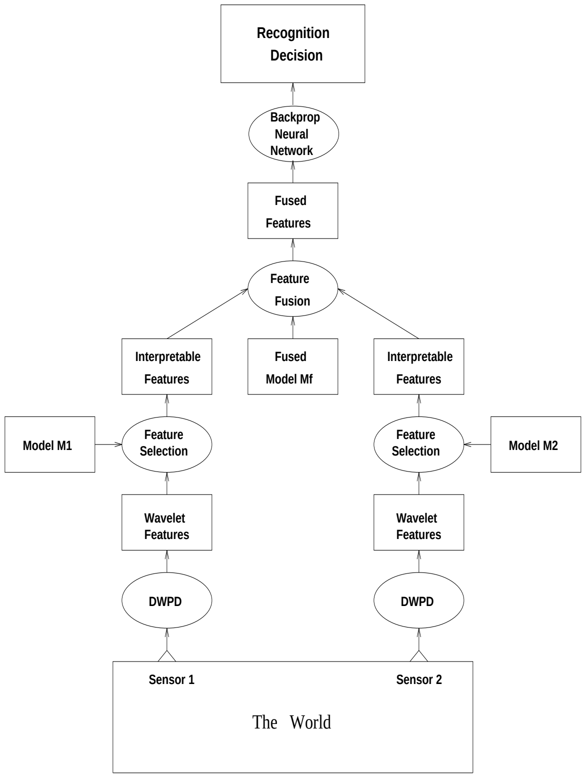

Figure 2: Automatic Multisensor Feature-based Recognition System (AMFRS)

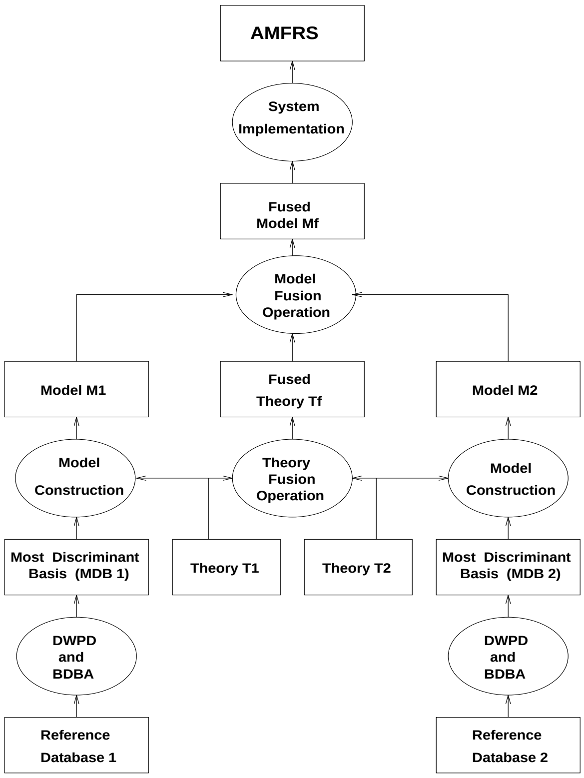

Figure 3: The Design Framework for the AMFRS Using the Model-Theory Based Feature Fusion Methodology

to each of the two reference databases as determined by the relative entropy discriminant measure. *Theory* T_1 *and Theory* T_2 contain domain knowledge about interpretable features in the MDB's (interpretable in terms of symbolic target/sound features), about target/sound classes, and about relations among them. *Model Construction* creates a model for a given domain theory. *Model* M_1 *and Model* M_2 represent features selected from both sensor data, operations on these features, and relations among these features. *Theory Fusion Operation* fuses the domain theory T_1 and the domain theory T_2 into one fused theory T_f. *Model Fusion Operation* fuses the model M_1 and the model M_2 into one fused model M_f. *Fused Model* M_f represents the fused features, operations on these features, and relations among these features. *System Implementation* implements Automatic Multisensor Feature-based Recognition System (AMFRS). The fused model M_f and the reference databases are used to train the neural network that classifies targets. *AMFRS* is the Automatic Multisensor Feature-based Recognition System that implements the proposed model-theory based wavelet-features selection (fusion) methodology.

The diagram in Fig. 3 shows that the designer of the AMFRS is given two reference databases of target signatures corresponding to two sensors. The Discrete Wavelet Packet Decomposition (DWPD) is used to transform these signatures from the time domain into the time/frequency (wavelet) domain and the Best Discrimination Basis Algorithm (BDBA) is used to select the most discriminant basis for each reference database. The designer also knows the domain theories T_1 and T_2 that contain the knowledge about interpretable features, target classes and relations between these features and classes. Knowing the most discriminant bases and the domain theories for each sensor, the designer constructs the models M_1 and M_2 for the domain theories T_1 and T_2 respectively. The knowledge of the fused theory allows the designer to select the right combination of features. The fused theory T_f is derived by using a theory fusion operator to fuse the domain theories T_1 and T_2. Since there is no physical law known for fusing data from these two sensors, the designer manually derives a theory fusion operator based on symbolic knowledge about the domain and the relation between the symbolic knowledge and the interpretable features. Using the fused theory T_f, the designer manually derives a model fusion operator. By applying the model fusion operator to the model M_1 and the model M_2, the designer derives the fused model M_f. The fused model M_f is used in the AMFRS to select interpretable features from the measurement data. The AMFRS implementation also involves off-line training of the backpropagation neural network based on the reference database of target signatures.

3 Models and Theories

The AMFRS has been constructed according to the methodology described above. To derive the fused model M_f and the fused theory T_f we used the theories T_i, T_r and the models M_i, M_r for the two sensors. Below we present just the resulting fused language, theory and model. All the details of the construction and implementation of this system can be found in [6].

Fused Language L_f

$L_f =$
$\{class1, class2, class3, class4, f_f, 0, 1, 2, C_{f_0}, C_{f_1}, \ldots, C_{f_7}, C_{f_8}\}$,

where $class1, class2, class3, class4$ are 9-placed relation symbols (classes of targets to be recognized using fused data), f_f is an 1-placed function symbol (a function that maps feature indices into feature values), $0, 1, 2$ are constant symbols, $C_{f_0}, C_{f_1}, \ldots, C_{f_7}, C_{f_8}$ are constant symbols (feature indices).

Fused Theory T_f

$class1(y_0 \ \ldots \ y_8) \iff (\forall_{y_0,\ldots,y_8}) \ \ ((y_0 = f_f(C_{f_0}) \wedge \ldots \wedge y_8 = f_f(C_{f_8})) \Rightarrow (1 \leq y_0 \leq y_2 \leq y_1 \ \wedge \ 1 \leq y_3 \leq y_5 \leq y_4 \ \wedge \ 2 \leq y_6 \leq y_7 \ \wedge \ 2 \leq y_8 \leq y_7))$,

$class2(y_0 \ \ldots \ y_8) \iff (\forall_{y_0,\ldots,y_8}) \ \ ((y_0 = f_f(C_{f_0}) \wedge \ldots \wedge y_8 = f_f(C_{f_8})) \Rightarrow (1 \leq y_0 \leq y_1 \ \wedge \ y_0 \leq y_2 \ \wedge \ 1 \leq y_5 \leq y_4 \leq y_3 \ \wedge \ y_7 \leq y_6 \leq 1 \ \wedge \ y_8 = 0))$,

$class3(y_0 \ \ldots \ y_8) \iff (\forall_{y_0,\ldots,y_8}) \ \ ((y_0 = f_f(C_{f_0}) \wedge \ldots \wedge y_8 = f_f(C_{f_8})) \Rightarrow (0 = y_0 \leq y_1 \leq y_2 \ \wedge \ 1 \leq y_5 \leq y_4 \leq y_3 \ \wedge \ 2 \leq y_6 \leq y_7 \ \wedge \ 2 \leq y_8 \leq y_7))$,

$class4(y_0 \ \ldots \ y_8) \iff (\forall_{y_0,\ldots,y_8}) \ \ ((y_0 = f_f(C_{f_0}) \wedge \ldots \wedge y_8 = f_f(C_{f_8})) \Rightarrow (1 \leq y_0 \leq y_2 \leq y_1 \ \wedge y_3 = y_4 = y_5 = y_6 = y_7 = y_8 = 0))$,

$C_{f_0} \leq C_{f_1} \leq C_{f_2} \leq C_{f_3} \leq C_{f_4} \leq C_{f_5} \leq C_{f_6} \leq C_{f_7} \leq C_{f_8}$,

where

$C_{f_0} = C_{r_0} \wedge C_{f_1} = C_{r_1} \wedge C_{f_2} = C_{r_2} \wedge C_{f_3} = C_{r_3} \wedge C_{f_4} = C_{r_4} \wedge C_{f_5} = C_{r_5} \wedge C_{f_6} = C_{i_6} \wedge C_{f_7} = C_{i_7} \wedge C_{f_8} = C_{i_8}$,

$f_f(C_{f_0}) = f_r(C_{r_0}) \wedge f_f(C_{f_1}) = f_r(C_{r_1}) \wedge f_f(C_{f_2}) = f_r(C_{r_2}) \wedge f_f(C_{f_3}) = (f_r(C_{r_3}) \wedge f_f(C_{f_4}) = f_r(C_{r_4}) \wedge f_f(C_{f_5}) = f_r(C_{r_5}) \wedge f_f(C_{f_6}) = f_i(C_{i_6}) \wedge f_f(C_{f_7}) = f_i(C_{i_7}) \wedge f_f(C_{f_8}) = f_i(C_{i_8})$.

Fused Model M_f

$$M_f =< A, I_f >,$$

or equivalently

$M_f =$
$< A; class1, class2, class3, class4, f_f, 0, 1, \ldots, 8; I_f >$,

where

$A = \{0, \ldots, 8\}$ is a universe of the model M_f,

$class1 \subset A^9$ is a relation, where $class1 = class1'_i \times class1_r$ and $class1'_i = class1_i \cap (A \cap \{6, 7, 8\})^3$,

$class2 \subset A^9$ is a relation, where $class2 = class2_r \times class2'_i$ and $class2'_i = class2_i \cap (A \cap \{6, 7, 8\})^3$,

$class3 \subset A^9$ is a relation, where $class3 = class3_r \times class3'_i$ and $class3'_i = class3_i \cap (A \cap \{6, 7, 8\})^3$,

$class4 \subset A^9$ is a relation, where $class4 = class4_r \times class4'_i$ and $class4'_i = class4_i \cap (A \cap \{6, 7, 8\})^3$,

$f_f : A \rightarrow A$ is a function, where $f_f = f_r \cup f'_i$ and $f'_i = f_i \mid_{A \cap \{6,7,8\}}$,

I_f is an interpretation function that maps symbols of the language L_f to appropriate relations, functions and constants in the universe A. It assigns relations $class1 \subset A^9$, $class2 \subset A^9$, $class3 \subset A^9$, and $class4 \subset A^9$ in the model M_f to symbols $class1$, $class2$, $class3$, and $class4$ in the language L_f respectively. We use $class1$, $class2$, $class3$, $class4$ to denote both the relation symbols in L_f and the relations in M_f. I_f also assigns a function $f_f : A \rightarrow A$ in the model M_f to a symbol f_f in the language L_f. We use f_f to denote both the function symbol in L_f and the function in M_f. And finally, I_f assigns constants $0, \ldots, 8$ in the model M_f to the constant symbols $C_{f_0}, \ldots, C_{f_8}$ in the language L_f respectively and special constants $0, 1, 2$ in the model M_f to the constant symbols $0, 1, 2$ in the language L_f.

3.1 Evaluation of the AFBRS

We evaluated the AMFRS feature selection methodology by applying it to three recognition applications: triangle/rectangle recognition, Automatic Target Recognition (ATR), and Automatic Lung Sound Recognition (ALSR) [6] by comparing the misclassification rates with the misclassification rate for the MDWC-based recognition system [9]. The misclassification results are presented for different levels of the Gaussian noise (represented by the standard deviation). Statistics for each level of noise were computed based on 100 runs. In this paper we show just the results for the ATR scenario. Fig. 4 shows the resulting AMFRS misclassification rates for different levels of noise for the multisensor ATR problem. This figure also shows the misclassification rate of a recognition system using Most Discriminant Wavelet Coefficients

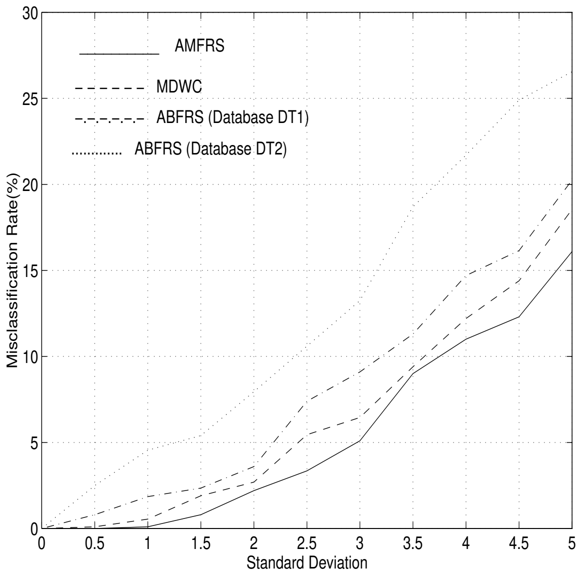

Figure 4: Misclassification Rates for the Multisensor ATR Using Model Theory (AMFRS) and Most Discriminant Wavelet Coefficients (MDWC)

(MDWC) as features. The MDWC are selected from among the features in the most discriminant bases for both test databases using the relative entropy measure. The misclassification results show that for our ATR problem the AMFRS has a better recognition accuracy than the MDWC-based recognition system.

4 Conclusions

Intelligent systems must be able to deal with unexpected inputs, i.e., with such inputs that were not anticipated by the system designer. The unexpected can include not only noise in the sensor measurements, but also qualitatively different kinds of data. To be able to handle this kind of situations, the system may need to generate new symbols and incorporate them into its processing scheme. It is natural to require that there should be some performance criteria that the system should obey even when dealing with such unexpected situations. However, as the structure of the system evolves, the requirements that were imposed on the system at the design time must be traced to the new structure. This is possible only if both the initial system and the transformations are defined in strict mathematical/logical terms.

While the use of logic and mathematics is advantageous from the point of view of measuring and predicting the system performance, there is a problem with the computational complexity of such a system. In this paper we briefly described our efforts to address both the application of logic and mathematics to developing intelligent systems and a solution to the problem of computational complexity. The main theme of this paper is how to combine logic/mathematics with software engineering to make the use of these formalisms practical. We showed an example of both a system that incorporates logical representations and various non-logical representations and techniques. The knowledge of the world is represented using structures from logic: theories and models. The non-logical methods include: component-based libraries of domain-specific knowledge, human-in-the-loop in the design of the system and in the creation of fused models and theories representing knowledge of the world, domain-specific system architecture, and off-line learning and noise handling using neural nets.

An automatic target recognition system is usually implemented using quantitative algorithms that involve signal processing and statistics. In our work we were able to demonstrate that the incorporation of symbolic knowledge improves the performance of an automatic target recognition system without compromising its computational efficiency.

Currently, we are pursuing research to expand the described methodology towards giving more autonomy to the system. As a first step, the methodology includes a formal specification of an intelligent system. The presumption is that the system would be goal oriented, i.e., it would not be an all-purpose system. Under these conditions it is possible to formally specify such a system. This high-level specification can be performed by the designer. The specification would include performance measures for the system, without specifying how the performance would be accomplished. The methodology also includes methods for refining specifications so that the performance requirements are satisfied. The outcome of the refinement process is dependent on the context in which the system operates, which in turn is defined by the inputs the system receives through its sensors. Depending on the range of operation of the system, the refinement process can be accomplished either automatically (for smaller ranges) or with human interaction (for larger ranges). Our goal is to investigate methods that will allow us to build intelligent system with higher levels of autonomy.

References

[1] P. Antsaklis. Defining intelligent control: Report of the task force on intelligent control. *IEEE Control Systems Magazine*, 14, No.3:4–5, 58–66, 1994.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International Group, Belmont, CA, 1984.

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

[4] J. Y. Halpern and M. Y. Vardi. Model checking vs. theorem proving: A manifesto. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 325–334, 1991.

[5] F. Hayes-Roth. Architecture-based acquisition and development of software: Guidelines and recommendations from the arpa domain-specific software architecture (dssa) program. Technical report, Teknowledge Federal Systems, 1994.

[6] Z. Korona. *Model-Theory Based Feature Selection for Multisensor Recognition*. PhD thesis, Northeastern University, Computer Systems Engineering, Boston, MA 02115, 1996.

[7] A. Meystel. *Semiotic Modeling and Situation Analysis: An Introduction*. AdRem, Inc., Bala Cynwyd, PA, 1995.

[8] NIST. Application portability profile (app) the u.s. government's open system environment profile, ose/1 version 1.0. Technical Report 500-187, National Institute of Standards, Gaithersburg, MD, 1991.

[9] N. Saito. *Local Feature Extraction and Its Applications Using a Library of Bases*. PhD thesis, Yale University, 1994.

[10] D. R. Smith. Constructing specification morphisms. *Journal of Symbolic Computation*, 15, 5–6:571–606, 1993.

[11] W. Zadrozny and M. M. Kokar. A logical model of machine learning: A study of vague predicates. In D. P. Benjamin, editor, *Change of Representation and Inductive Bias*, pages 247–266. Kluwer Academic Publishers, Boston-Dodrecht-London, 1989.