A Multimodel Approach to Reasoning and Simulation

Paul A. Fishwick N. Hari Narayanan Jon Sticklen Andrea Bonarini

Abstract— **Models that are constructed within the bounds of a single paradigm are not sufficient for modeling all aspects of complex systems. Therefore, even though reasoning and simulation systems that utilize a single modeling paradigm are the current norm, we explore a *multimodel* approach in this paper. A multimodel approach is defined as one in which more than one model — each derived from a different perspective, and utilizing correspondingly distinct reasoning and simulation strategies — are employed. By describing four models which illustrate the use of different modeling techniques, we show how a multimodel approach can enrich the modeling environment and make it correspond better with real world information. Our models come from many sources — *Systems and Simulation Theory* for the modeling of natural phenomena and artificial devices, and *Artificial Intelligence* and *Cognitive Science* for the modeling of human intuition and expertise in reasoning. Generalizing from these four models, we suggest that modeling complex systems may best be approached from an integrated architectural viewpoint which combines multiple modeling paradigms.**

Keywords— **Multimodeling,Visuo-Spatial Reasoning,Abstraction Levels,Qualitative System,Functional Modeling.**

I. Introduction

THE advent of computers provided a powerful tool for deriving the behavior of complex systems: *simulation*. The corresponding research discipline, Systems and Simulation Theory, has grown to provide theoretical foundations and practical tools for computer-based simulation. However, systems that are thus studied have become increasingly complex. These now include systems with components for which complete analytical models may not be available (i.e., ecological systems) and systems with intelligent agents as components (i.e., social systems). For such systems, the traditional approach to simulation, in which a user programs a single simulation model with initial (boundary) conditions, runs the simulation program, and interprets the typically voluminous data that the program outputs, is proving to be inadequate. There is a need for incorporating "intelligence" into simulation programs to provide the user with support in setting up simulation models, to have the simulation guided intelligently at run time, and to provide support for interpreting the simulation results. This need is being met by reasoning tools and techniques from Simulation, Artificial Intelligence, and models of human cognition from Cognitive Science. Recent *AI and Simulation Workshops* that have been held in conjunction with the *National Conferences on Artificial Intelligence (AAAI)* and a conference series entitled *AI, Simulation and Planning in High Autonomy Systems* [1] provide ample testimony to increasing interest in this confluence of disciplines.

A fortuitous result of these developments has been the realization that *reasoning* (as investigated in AI and Cognitive Science) and *simulation* (as investigated in Systems and Simulation Theory) have much to contribute to each other. In simulation literature we see many papers discussing how reasoning techniques have been profitably incorporated into traditional simulation models [2], [3], [4], [5], [6], [7], [8], while in AI literature we see the growth of a subarea called *Qualitative Reasoning about Physical Systems* [9] that uses "qualitative simulation" [10] as a technique for deriving, and subsequently reasoning about, system behaviors. We now offer some initial observations based on the synthesis of AI and simulation research.

- A capability for intelligently reasoning about the evolution of a simulation is of benefit to simulation models.
- A capability for simulating the behavior of systems is of advantage to reasoning programs.
- It follows that integrated systems capable of both simulation and reasoning will provide leverage in dealing with a wide variety of complex systems.

Given this context, our central thesis is that a *multimodel* approach is an appropriate way to exploit the advantages of simulation models and reasoning techniques within a single system. We define a multimodel approach as one with a knowledge representation scheme composed of different types of knowledge coupled together. Each type of knowledge may provide the basis for a reasoning capability to be used for one aspect of the problem that the overall system addresses. For example, a bond graph is useful for modeling and simulating the energy flow characteristics of a physical system, but a functional model[1] may be more appropriate for reasoning about how the system components contribute to its overall functioning.

Any real world system can be understood at varying levels of abstraction. As one progressively moves to a more detailed view of the system, eventually, the complexity one

Paul A. Fishwick is Associate Professor with the Dept. of Computer and Information Sciences, University of Florida. E-mail: fishwick@cis.ufl.edu .

N. Hari Narayanan is a Visiting Research Scientist at the Advanced Research Laboratory of Hitachi, Ltd. E-mail: narayanan@harl.hitachi.co.jp .

Jon Sticklen is Associate Professor with the Dept. of Computer Science , Michigan State University. E-mail: sticklen@cpswh.cps.msu.edu .

Andrea Bonarini is Assistant Professor with the Dept. of Electronics Engineering at the Politecnico di Milano, Italy. E-mail: bonarini@ipmel2.elet.polimi.it .

[1] We use the word "functional" in this instance to refer to the functionality (or "purpose") of the device, and not the low-level *transfer functions* that may be a part of a mathematical model.

encounters is overwhelming. Although all real systems have this property, some systems present great complexity even at the most abstract level of understanding — systems such as ecological systems or engineered chemical refining plants. To reason about such complex organization, a problem solver might rely on more than one type of knowledge organization.

For example, ecologists pondering the cause-effect relations that might lead to global warming phenomena might first use "rule of thumb" heuristic knowledge organizations to obtain clues for what part of a large causal knowledge base might hold useful information. Then they might go on to explore (qualitatively) a causal-net like structure that could lead to a conjecture on what might happen should production of carbon dioxide increase by a factor of two. Finally, they might use this qualitative suggestion to initiate a large numerical simulation of the conjectured situation. Such a progression from compiled-level knowledge organizations, to model-based structures, to numerical calculations, is necessary since no single modeling method will serve to answer all questions about the system's behavior.

However, most traditional AI systems and simulation models involve a single paradigm or knowledge organization. For instance, an AI system may be based on semantic networks, fuzzy logic, or production rules; a simulation model may be based on a Markov model, differential equations, or a Petri net. We argue that a knowledge organization scheme composed of a variety of models, with each providing a reasoning or simulation capability for addressing one aspect of the problem, is the most effective way of tackling a complex problem at varying levels of abstraction.

II. Overview

While a multimodel approach can provide different perspectives on a problem and facilitate addressing the problem from different levels of abstraction, one should also be concerned with properly integrating different models (or types of knowledge organization). In this paper however, we have chosen to focus on the models themselves, and we concentrate on the following four areas:

- *Visuo-Spatial Reasoning*: If we have information in the form of a diagram depicting the spatial configuration of the components of a device, along with some basic knowledge regarding the components, can we derive the spatial behaviors of the device by a combination of predictive reasoning and visualizations (image transformations)?
- *Functional Modeling*: Suppose that we know the purposes (or goals) of a device and its component subdevices. Can the organization of our causal understanding of the device be used as a framework for integrating numerical relationships which govern the detailed behavior of the device? Moreover, can this integration of causally indexed, qualitative understanding with numerical relationships be used as a basis for numerical simulation?
- *Qualitative and Numerical Data in Simulation*: We may find that we have a well-defined, numerical model of a part of a complex system, and only incomplete, qualitative knowledge about another part of it. How can we simulate and reason about such systems?
- *Multimodel Integration*: With a multimodel scenario, the modeling task is complicated by the requirement of maintaining consistency across levels of abstraction. How can we design multi-level models so that models at one level serve as valid abstractions of models at lower levels?

A model for visuo-spatial reasoning, one which facilitates reasoning about spatial behaviors of device components from schematic diagrams, is described by Narayanan in Section 3. A mathematical model does not contain any *explicit* knowledge of the goal or purpose of the modeled system. In Section 4 Sticklen shows how the functional modeling technique addresses this drawback by organizing causal knowledge about a system around its known teleology and by incorporating mathematical relationships among state variables within this knowledge structure. Bonarini recognizes that integrating qualitative and numerical simulation models can mitigate limitations of each type of simulation while retaining their advantages. In Section V he describes such an integrated model. Fishwick describes the method of multimodel integration in Section VI. In the concluding section (Section VII) we will discuss how these four different models contribute to a multimodel approach toward reasoning and simulation. Their differences serve to enrich such a multimodel system in comparison with one that depends on a single modeling paradigm. Thus, we propose that the use of multiple types of knowledge in modeling complex real world systems has a tremendous advantage over the use of a reasoning system which is monolithic in nature.

The seeds of this collaboration among the four authors (whose backgrounds span Systems and Simulation Theory, AI, and Cognitive Science) were sown at the *AI, Simulation and Planning in High Autonomy Systems* conference held in April 1991 at Cocoa Beach, Florida [1]. There it became clear that both reasoning and simulation approaches were going to be necessary for addressing the challenges that complex systems pose. This motivated the preparation of a collaborative paper on multimodel reasoning and simulation.[2]

III. Visuo-Spatial Reasoning: A Visual Approach to Commonsense Spatial Reasoning (VSRM)

A capability to reason about how devices operate is clearly an important one for reasoning and simulation systems. Many devices contain mechanisms with spatial behaviors. Not surprisingly, the problem of reasoning about spatial behaviors of mechanisms has received much attention in AI [11], [12], [13]. However, reasoning qualitatively about the operation of a mechanism directly from its diagram without the benefit of metric information (*visuo-spatial reasoning*) is a commonsense spatial reasoning ca-

[2]Inquiries regarding this paper may be sent to the first author, while questions about any particular model should be addressed to the author of the corresponding section.

pability that has not yet been automated. Consider, for example, Fig. 1 from which the following questions can be easily answered:

[Figure 1 about here.]

(1) If gear-1 is turned clockwise, in which direction will gear-2 move? (2) Will the gear motions stop at some point? Thus it appears that a qualitative characterization of the spatial behaviors of a mechanism can, in principle, be generated directly from its diagrammatic representation, without creating and using more complex geometric or algebraic representations. There are however questions, such as what the angular velocity of the lower gear will be if the upper one is being turned at a certain velocity, which cannot be answered from a diagram alone without additional metric information. Nevertheless, a qualitative characterization as above can be very useful in guiding more detailed analyses of devices. Therefore, a model of visuo-spatial reasoning is presented in the following two sections.

A. Problem Solving Strategy

A cognitive strategy for solving visuo-spatial reasoning problems [14] involves the following activities. *Information extraction*: the diagram serves as a spatially organized representation of information, and information is spatially indexed and retrieved from it during problem solving. *Prediction generation*: predictions, based not only on information from the diagram, but also on conceptual information (e.g., rigidity of objects involved), are made about how spatial interactions (such as a collision) affect object behaviors. *Visualizations of spatial behaviors*: visualizations of predicted behaviors are carried out using the diagram to discover their effects, which in turn generates more predictions, and this drives the problem solving forward. This strategy may be viewed as repeated cycles of *analyze-visualize-detect-predict*, in which the diagram is analyzed, object behaviors are visualized, spatial interactions among objects are detected, and their effects are predicted.

Consider the example of Fig. 1 again. The first step of problem solving is to look at the diagram to comprehend the geometry of the gear configuration and to note areas of the diagram where objects are in contact or close proximity. This visual analysis takes place through visual operations such as scanning. Then, since the problem specification includes an initial motion of gear-1, a visualization of this motion is performed. This visualization can be carried out only until a spatial interaction is detected. This is because an interaction among objects can potentially change the way in which the configuration is evolving. So the visualization of gear-1's rotation cannot continue after it is seen to make contact with the lower gear.

We call object configurations with spatial interactions, detected during visualizations, *deliberative states*. Deliberative states contain object interactions which can affect the object configuration's evolution by imparting motions to previously stationary objects, arresting current motions, or changing the nature of current motions. In order to predict which of these changes, if any, will occur following a deliberative state, deliberation or reasoning is required — hence the name. The occurrence, elimination, or change in the nature of contacts between objects are typical interactions that signal a deliberative state. A deliberative state triggers the transition of the problem solving process from a visual phase (analysis and visualization) to a deliberative phase (predictive reasoning).

The goal of deliberation is to predict what happens as a result of the visualized object interaction. In the current example this goal is to predict how the two gears will behave as a result of their contact. The prediction that we will make at this stage is that this contact will result in the lower gear starting to turn counter-clockwise in tandem with the rotation of the upper one. Note that in order to make this prediction one requires three types of information. The first is information about the state of the gear configuration when their contact was visualized. This information is available from the visualization process involving the diagram. The second is conceptual information that is not available from the diagram (such as the fact that both gears are rigid). The third is information about how objects behave in the physical world, in this case, that a moving rigid object coming into contact with a stationary rigid object tends to push the stationary one in the direction of motion at the point of contact.

B. A Visuo-Spatial Reasoning Model (VSRM)

What kind of a model should a reasoning and simulation system use for visuo-spatial reasoning? In answering this question, we take a descriptive approach rather than a prescriptive one. In other words, instead of presenting a particular model, we describe desirable characteristics of such a model along with suggestions derived from a specific model that was recently developed [15], [14].

A VSRM needs to represent three types of knowledge. First, there is spatial knowledge explicit in the diagram of objects being reasoned about. Second, there is conceptual knowledge about objects and their parts; information that is relevant to reasoning, but which is not available from the diagram. Third, knowledge that allows the model to generate predictions regarding effects of spatial interactions detected during visualization needs to be represented.

Let us first consider how diagrammatic information can be represented. The structure of a representation should facilitate operations to be performed on it. Visuo-spatial reasoning requires three kinds of operations on diagrammatic information. Two kinds involve access (and modification) of diagrammatic information. One is spatially-indexed access in which the accessed element satisfies some spatial relation. An example of this is accessing any object that is in contact with a given object in a configuration. The other is object-oriented access in which the element satisfies some non-spatial relation (e.g., part-of). An example is accessing the protruding tooth of gear-2 from a representation of Fig. 1. The third kind of operation is visualization which transforms the representation to reflect the evolution of a configuration due to object motions.

An array of cells (two-dimensional for diagrams and

three-dimensional for 3D images) is a representational structure that is particularly amenable to spatially-indexed access. Each cell can contain one or more symbols representing the part of the diagram which the cell covers. Such arrays, called occupancy arrays, have recently been proposed as representations useful for computational imagery [16]. Computer analogues of visual operations that humans employ to achieve spatially-indexed access, such as scanning and boundary following, can easily be implemented on such arrays. An occupancy array facilitates visualization as well, since array procedures to simulate the effects of object motions can also be designed easily. However, any high resolution array representation of a diagram can have its individual cells represent only small parts of contours in the diagram being represented. This makes object-oriented access difficult since such access requires aggregation of subcontours in a diagram into meaningful parts of objects in the diagram. In other words, there is a mismatch between the grain size required by object-oriented access and the grain size of array cells. A solution to this problem is to overlay a descriptive part-of hierarchy on top of the array representation. Such a composite representation consisting of a descriptive hierarchy overlaid on a depictive array, as illustrated in Fig. 2, is a suitable diagrammatic representation scheme for a VSRM.

[Figure 2 about here.]

Conceptual (non-diagrammatic) information about objects may be represented using frames which are linked by pointers to the diagrammatic representation so that the reasoning system can access the conceptual properties of an object from its diagrammatic representation and vice versa.

Predictive knowledge —knowledge that is utilized to make predictions about spatial behaviors of objects subsequent to an interaction detected during visualization— is a critical part of a VSRM. It should combine both diagrammatic information about the nature of the detected object interaction and conceptual information about the objects involved, in order to arrive at a prediction. This is because spatial interactions between objects in diagrammatically identical configurations can have different effects depending on conceptual properties, such as rigidity, of objects involved. Furthermore, units of predictive knowledge should be specified at an abstract enough level that these have broad coverage. What constitutes predictive knowledge depends on the domain for which a VSRM is being developed. For a 2-dimensional blocks world domain [14], representational units called "visual cases" have been used to encode predictive knowledge. Fig. 3 shows a sample visual case. A visual case has five parts: the prototypical object configuration (POC) that the case represents, a description (ED), visual conditions (VC) verifiable from the diagrammatic representation, non-visual conditions (NVC) verifiable from conceptual knowledge, and predicted event or events (P).

[Figure 3 about here.]

The model in [14] contains about seventy such cases covering translation, rotation, sliding and tilting of objects.

Based on the aforementioned problem solving strategy, a four-stage reasoning process can be designed for a VSRM: analysis of diagrammatic representation, visualization of motions, detection of deliberative states, and prediction of spatial interaction effects. Fig. 4 shows the flow of control among these stages.

[Figure 4 about here.]

The first stage is analysis of the diagrammatic representation of the object configuration. One aim of this analysis is to verify the feasibility of current predictions. Feasibility verification is important because an effect, predicted in the deliberative phase, of an interaction may not in fact be feasible due to the influence of objects other than the interacting ones. For example, a collision, detected during visualization, between a moving object and a stationary one may generate a motion prediction for the stationary object in the deliberative phase. But the stationary object may already be obstructed by a third object, rendering the prediction infeasible. An analysis of the current configuration can reveal such situations. The second stage, visualization, involves transforming the diagrammatic representation in accordance with how the object configuration will change due to current motions.

Visualization is halted if and when an object interaction is detected. The configuration that the diagrammatic representation depicts at this point is called a deliberative state. Which types of object interactions are counted as denoting a deliberative state depends on the domain and class of problems for which a VSRM is being developed. The model in [14] considers the following types of interactions as denoting a deliberative state: a collision between objects, the removal of a previously existing inter-object contact, and a change in the nature of an inter-object contact. When detected, these interactions will trigger a transfer of control to the deliberative phase. In this phase, the type of detected interaction can be used to select applicable subsets of predictive knowledge contained in the VSRM. If visual cases are used to encode predictive knowledge, their visual and non-visual conditions can be used to select, match and apply relevant cases. If the application of predictive knowledge generates one or more predictions, the control loop is reentered at the visual analysis stage. The subsequent visualization will be guided by these new predictions. This control loop repeats until the evolution of the object configuration ceases or no more object interactions are predicted.

C. Summary

Visuo-spatial reasoning is a qualitative approach to diagrammatically representable spatial reasoning problems. It also provides a framework for integrating quantitative methods [17]. Quantitative information can be represented in a VSRM as part of conceptual information and quantitative methods can be invoked during the deliberative phase of reasoning. Two advantages of a VSRM, accruing from its use of diagrammatic representation of spatial

configurations, are that (1) it can quickly provide plausible predictions regarding spatial behaviors of objects in a configuration by directly manipulating its diagrammatic representation instead of having to construct and use complex algebraic or geometric representations, and that (2) such predictions can be used to guide the selective application of more complex spatial reasoning methods. However, this model of visuo-spatial reasoning has some limitations. Its reasoning strategy depends on the visualization using diagrammatic representation being able to detect —in the correct order— spatial interactions that may occur as an object configuration evolves. But this is not possible if the order depends on non-diagrammatic parameters such as mass or acceleration. Also, array-based representation and visualization of three-dimensional configurations can become quite time consuming.

IV. Functional Modeling (FM)

A. Functional Modeling for Capturing Quantitative Relationships

The goal of Functional Modeling (FM) is to make use of the known purpose (or goal) of a device, to use that knowledge to organize causal understanding of the device, and to provide a reasoning algorithm which can be used to simulate the device given starting conditions. The roots of FM lie in research by Sembugamoorthy and Chandrasekaran which set the initial representational concepts for the functional point of view [18]. Sticklen and Chandrasekaran applied and extended the initial work to include a simulation component to support diagnostic problem solving in a medical domain [19]. Goel has used a simulation viewpoint to attack problems of design problem solving [20]; Punch has likewise used an FM simulation point of view as a basis for the integration of Generic Tasks [21]. Allemang has recently reported an application of the methodology of functional representation to model computer programs [22]. Finally, Keuneke recently completed a research project in which she demonstrated that the functional representation is a valuable framework for the extraction of explanations of diagnostic conclusions [23].

Overall, the functional viewpoint centers on enumerating the proper primitives which can be used to organize causal device understanding, and on working out algorithms for utilizing the representation. Similarly motivated research has recently been reported by Chittaro et al in Italy [24], and by Franke at UT-Austin [25]. In this section, we characterize a recent extension of the FM methodology to encompass quantitative relationships between state variables of a modeled device. This extension will be described *via* our test bed example, the automotive cruise control device. In research carried out in this test bed —and fully described in [26]— we have found that the functional viewpoint can be characterized as providing a framework for organizing a series of quantitative calculations.

B. The Automatic Cruise Control System

The automatic cruise control (ACC) system is a hybrid system that automatically controls the cruising speed of the vehicle. It consists of electrical, electromagnetic, pneumatic, and mechanical components. At the top level, the ACC can be conceptualized as an equilibrium seeking system which tries to eliminate the difference between two control signals: the command-speed signal set by the driver of the vehicle, and a signal indicating the vehicle's true speed. The organization of the ACC is indicated in Fig. 5.

[Figure 5 about here.]

Representationally, the functional approach consists of two sublanguages for device description: one sublanguage for description of function, and one sublanguage for description of behavior (i.e., a language of state variable change). To solve the ACC representational problem, we extended both sublanguages. The function sublanguage is very simple, consisting of only three parts: a precondition, a postcondition, and a pointer to "implementing behaviors". Previously, we have represented the postcondition in terms of the primitive ToMake (i.e., the action of this primitive is to modify the value of a state variable of the device). In the ACC example, we need another primitive, one which will allow us to clearly indicate that the action of a function is going to be to fix a state variable based on the context of other state variable values at the point at which the function is invoked. We have called this new primitive of the function sublanguage *ToCalculate*. Corresponding to the *ToCalculate* primitive in the function sublanguage, we also required a similar new concept for the state sublanguage. The representation of a behavior in the FM approach consists of a graph structure in which the nodes (after the first level nodes) represent statements about changes of device state variables. Until our experience with the ACC, these statements about state variable changes were of two types: setting state variables to some stated value, and incrementing state variables by some set amount. To naturally represent the ACC we augmented our sublanguage for state by allowing "parameterized state change" in which a node in a behavior can be stated as a numerical calculation over other variables of the device which then sets a stated variable according to the result of the computation.

The first step of the FM methodology is device decomposition. To represent the ACC, no extensions to our previous work were needed to accomplish this step. The second and third steps of the methodology are to represent the abstractly stated functionality that is known for each device, which amount to listing its preconditions, postconditions, and listing a pointer to its implementing behavior(s). The third step of the methodology is to represent each behavior as a state change graph such as shown in Fig. 10.

Let us begin with the highest level behavior of the cruise control system, the adjust-speed behavior. As shown in Fig. 6, if the speed of the car does not match the speed the driver has set (i.e., if the error signal is not zero), then a number of causal consequences will follow in a set sequence.

[Figure 6 about here.]

Note that by looking at one graphic (Fig. 6), it is possible to grasp the overall operation of the cruise control system. The first causal consequence in Fig. 6 is that a new value of the "duty cycle" is calculated. The reason that causal consequence occurs can be ascertained by following the link *MakeThrottleControlSignal* function of the control electronics subsystem, which is shown in Fig. 7. Going to a yet deeper level of detail, Fig. 8 shows the implementing behavior make-duty-cycle-behavior.

[Figure 7 about here.]

[Figure 8 about here.]

There are several key points that should be emphasized about the representation of the ACC that we have shown. First, an FM representation is modular. Causality is represented in small chunks that chain together via annotations of why one causal chain follows another. Second, an FM representation "bottoms out" at a point that is appropriate for the problems the model must address. If our model of a cruise control were to be used to trouble shoot sub-chip level devices, then our representation would have to be extended. As it stands now, we would be able to model the ACC device to levels such as the OpAmp level in Fig. 9. Third, the chunks are organized around meaningful concepts: the known functions of the device. Fourth, the extensions which we have undertaken provide a highly organized, and meaningful way of capturing causal knowledge about the ACC device.

[Figure 9 about here.]

We now move to the reasoning algorithm we have developed to perform consequence finding over such an FM representation. Space here precludes a detailed discussion of our consequence finding engine, but it is fully described in [19]. In synopsis, the reasoning algorithm first finds the highest level function in the device which is applicable for the stated boundary conditions of the problem. From that high level behavior, a macro-expansion style of execution proceeds to build up what we term the PSD - the *particularized state diagram* - so called because it is particularized to the boundary conditions. Once one pass is completed of this macro expansion of functions and behaviors, then the process repeats until no functions of the system are applicable. Fig. 10 shows an example PSD after one cycle of execution; Fig. 11 shows the PSD of the second cycle of execution for the ACC.

[Figure 10 about here.]

Note that the vehicle speed has increased from the initial condition (60 MPH) and is approaching the set speed of 65 MPH. Because the error signal is still not 0, the simulator will remain active; it will repeat the same steps as before, and will produce the values shown in Fig. 11 at the end of the second invocation. It is important to note that the PSD graph is not simply behavior graphs with all annotations removed. Each behavior begins with a number of tests on state variables. During simulation, if a test on a state variable fails, then the behavior in which that test resides is not applicable, and will not become part of the PSD. Thus the PSD is calculated only for the specific case being examined (i.e., only for the stated initial conditions). Although we do not further discuss it here, it is interesting to note that after the third simulation, since the error signal is still not 0, the speed is over 65 MPH. In fact, proceeding with the simulation produces a damped oscillating behavior. Although within our current framework the FM simulator could not recognize and label the behavior as "damped and oscillating," it was very encouraging to observe this result.

[Figure 11 about here.]

C. Functional Modeling in the MBR Landscape

Functional Modeling is best understood as an intermediate type of reasoning between highly compiled level approaches and naive physics approaches. The tasks of compiled level problem solving is to relate associational knowledge directly to desired conclusions. For example, compiled level classification for medical diagnosis relates observations of patient states (signs and symptoms) to diagnostic categories. On the other hand, naive physics [27], [28], [29] addresses the task of deriving the large scale behavior of a device from the behaviors and connections of small scale devices. For example, in the medical area, Kuipers has developed a QSIM-based approach for modeling part of kidney function from known small scale behavior.

Functional Modeling is intermediate between association-based reasoning (compiled-level) and reasoning aimed at deriving large scale behavior from small scale behavior and connection (naive physics). The starting point for developing an FM model is knowing *a priori* what the functions (or purposes) of a device and its components are. ¿From that knowledge, FM provides a framework for reasoning towards particular performance of a device where "particular" means with stated boundary conditions.

We have shown that with proper extension, we can utilize FM as a framework for organizing numerical calculations about a device. An apt question would be "Why can't numerical calculations about the ACC be expressed easily in a simple Pascal program?" One reply to that question is that yes, all the numerical calculations required to carry out the solution of the ACC could have been done in Pascal, but how well would such an approach scale? Suppose we were representing a nuclear power plant. How difficult would it be to develop a similar Pascal program in that case? The reason that our FM approach to organizing numerical calculations will scale well is that the approach emphasizes (a) modularity, and (b) an organization based around known functionality of the device. In our approach, the numerical calculations are simply ways of determining the results of changes in state that are necessary to achieve known functionality. A crucial issue which any MBR technique must face is the issue of "model selection." One of the reasons that model selection is difficult is due to the multiple di-

mensions associated with the task. Along one dimension, we must select the level at which we want to represent our model. As Davis [30] points out, no model is complete.

The Functional Representation deals straightforwardly with this fact by including the ability to point to "world knowledge" as the reason for a given state variable transition (in a behavior). This gives an ability to the modeler to construct a model that "bottoms out" at whatever level is appropriate. The issue of the type of model we want to construct should be based on (a) the representational primitives offered by a particular type of model, and (b) the reasoning that a particular type of model enables. If the knowledge we have of a device to be modeled can be expressed in the primitives of a particular approach, and if the output of reasoning with that approach matches what we need to have in terms of output, then that particular type of modeling approach would be a good candidate. This statement may seem self-evident. Yet for the most part, MBR has not dealt explicitly with issues of types of models in these terms. We believe that one of the strongest arguments supporting the FM approach to MBR is the relative clarity of statement of the representational primitives of the approach, and of the reasoning methods that come with the approach.

D. Summary

Functional Modeling is a technique for organizing causal knowledge about devices. That causal knowledge has been extended (in the cruise control testbed) to include knowledge of quantitative relationships. In lineage, FM is largely an outgrowth and extension of the qualitative reasoning community, at least in terms of starting intuitions. However, with the realization that its real power lies in organizing device understanding for ease of use, new bridges to other segments of the modeling landscape have become possible. In particular, armed with the insight that the conceptual core of FM is *not* the distinction between qualitative and quantitative reasoning, it becomes again clear that to develop robust models, multiple modeling approaches are going to be necessary. In our own work at Michigan State, we are now investigating the integration of bond graph techniques and FM techniques. Our continuing goal is to develop methods that will allow engineers and scientists to more easily develop, and share their understandings of complex devices.

V. Integrating Qualitative and Numerical Simulation Models (QQSIM)

A. Motivations

Numerical simulations support many engineering tasks such as process monitoring and diagnosis, test of design parameters, identification of critical situations, planning, and scheduling. Usually, only specifically trained people can interpret the results of numerical simulations. Moreover, if the aspects of the system to be modeled are complex, both the development of models and the simulation run time are expensive in time and money.

To mitigate these problems, qualitative simulation [31] has been proposed as an alternative way of describing the behavior of a system. Qualitative variables may take a value which belongs to a limited set of mutually exclusive, rank-ordered symbols. The dynamics of variables is constrained by relationships. Some of them are analogous to the ordinary numerical ones, whereas others provide weaker constraints. Criteria for the successful application of a qualitative approach include:

- only incomplete knowledge about a system is available: therefore it is impossible either to identify quantitative equations among variables or to find numeric values for parameters and initial states;
- the user is interested in a whole class of experiments, instead of just the one produced by a numerical model;
- the generalization of raw data at a higher level of abstraction makes the decision-making process easier [32].

In many engineering applications, a numerical description of the behavior of the system is not needed: considerations about general trends can be successfully used to complete the given task. Moreover, the qualitative approach makes it possible to do (usually) quicker and cheaper simulations, producing qualitative descriptions of the states of the simulated system, which can be interpreted with little effort.

The well-known drawbacks of the qualitative approach arise from the impossibility to relate the simulation results to the numerical values of the parameters. This produces an exponential growth of the number of the possible states the simulated system reaches. If quantitative information is available, the evolutions of each variable are, in general, more constrained, possibly leading to a smaller set of system behaviors. At the other extreme, quantitative simulation produces just the behavior compatible with the available numbers and numerical relationships.

Qualitative simulation of complex systems may generate very large envisioning spaces. Here the advantages of qualitative simulation drop: the program runs for a long time and experts have to work hard to find the interesting behavior among the large number of possible states the system may reach — each one described by many variables. Moreover, since the complexity of qualitative simulation usually grows exponentially with the dimension of the model, some applications cannot even run on the available machines.

Here, we present a method to integrate these two modeling paradigms, enabling a reuse of existing numerical models. The method has been implemented in *QQSIM* [33], a system running in Common Lisp and CLOS.

B. Basic assumptions

We assume that the system to be modeled can be partitioned in two parts: one described qualitatively and the other one numerically. The quantitative model is defined in terms of a set $QT = \{Qt_1, ..., Qt_n\}$ of quantitative variables, and the qualitative model in terms of a set $QL = \{Ql_1, ..., Ql_m\}$ of qualitative variables, where the set $QTL = \{Qtl_1, ..., Qtl_p\}$ of the so-called *shared variables* is

bijectively mapped both onto a subset $\{Qt_l, ..., Qt_{l+p}\}$ of QT and onto a subset $\{Ql_t, ..., Ql_{t+p}\}$ of QL. Therefore, the variables of QTL admit a double representation: one on quantitative and the other on qualitative terms.

Both the models (qualitative and quantitative) are used for a parallel simulation, where the numerical evolution of the shared variables is qualitatively interpreted and used to prune the incompatible qualitative behaviors. As a consequence, the results from qualitative simulation contain also quantitative information about both time and variable values.

This approach can be used to answer questions such as:

- Which states will the system reach, given the initial conditions?
- What will be the value of variable X at time T?
- When will variable Y reach the value $\tilde{Y}$?

QQSIM may provide either qualitative or quantitative answers to the second and third questions above, depending on the ongoing simulation and the underlying models. For instance, if time T is exactly the time to which some of the shared variables reached a limit point, the answer to the second question will be the numerical or qualitative value of variable X at that time, depending to which of the two models X belongs. Otherwise, time T will be compared with the available time labels, and the returned values of X will be the ones of the states whose time labels are qualitatively compatible with T.

C. Definition of Models

Two models are used in QQSIM: a numerical model and a qualitative one. The numerical model is a standard model used for numerical simulation. We would like to stress the fact that our approach can be applied to already existing numerical models, to enrich them with qualitative considerations. Required properties for the numerical model are:

- the numerical simulation has to produce a temporally ordered sequence of values for the variables in the model;
- it should be possible to stop and restart the numerical simulation whenever the synchronization algorithm requires it;
- the sampling frequency should be high enough to avoid the presence of two limit points (i.e. maxima, zeroes, or minima) in the same interval. This requirement is usually weaker than those usually taken in numerical simulation, so we assume that it is always satisfied.

The qualitative model is expressed using the standard QSIM elements including qualitative variables and constraints. Qualitative variables, identified by a label, may take a qualitative value, which is a pair $< value, trend >$, where *value* is one element of the set {+, 0, -}, and *trend* is an element of the set {INC, DEC, STD}. Each pair $< value, trend >$ is associated to a landmark (or to an interval between landmarks) where it holds. Landmarks are represented by labels and form an ordered set. As in Q3 [34] [35], landmarks can be associated to actual numerical values. Constraints among variables state algebraic (ADD, MINUS, MULT), derivative (DERIV), and qualitative (M^+) relationships.

The simulation generates states that the system may reach. Each state consists of a set of values — one for each of the qualitative variables — and a time label, expressing a value belonging to the numerical simulation time axis when the state is reached. This label can be computed only when at least one of the shared variables reaches a limit point in the present state, since only in this case do we know exactly from the numerical simulation when this event happens.

The numerical simulation may also supply numerical values for landmarks present in the values of the shared variables.

D. Model Interfacing

The two models are integrated by the simulation process. First the initial conditions for both the models are checked for consistency. Then, the numerical model is started, producing values which are monitored for a qualitatively significant change. Numerical value sequences for the shared variables are translated into qualitative terms. When a shared variable reaches a limit point, the numerical simulation is halted.

The (*QTL-process*) is performed in two steps. During the first step, each qtl_i is analyzed in its trajectory, independently from that of the others. Its qualitative behavior is identified, along with the corresponding limit-points. This identification is based on the successive sampled numerical values as proposed by the quantitative simulation.

For instance, if at time T_1 variable qtl_1 takes on the value of 20 and at time T_2 it takes on the value of 21, its qualitative behavior will be $(< T_1, T_2 > (pos, inc))$, which has to be read "in time interval $< T_1, T_2 > qtl_1$ is positive and incrementing."

Inferential steps might be required: for instance, when a *(pos dec)* follows a *(pos inc)* interval, a *(pos std)* has to be interposed to preserve qualitative continuity. These inferences are based on a 9×9 table, mapping all the possible qualitative states for a variable onto subsequent variable states. Acceptable sequences, impossible ones and interpolations thus become identifiable. Having detected the behavior of adjacent intervals, a second table defines the behavior in the intermediate points.

The second step of the QTL-process involves a synchronization of the shared variables. With our assumptions on the sampling rate, variables having limit points in adjacent intervals may be actually related so as to have limit points formally coinciding. This is the case for a variable and its derivative, or the case of two variables differing by a constant. The exact location of maxima and minima for each sampled variable is not definite: a maximum is located on the basis of three points, the middle one with the highest of the three values (the lowest in case of minimum). It is therefore possible that the exact location of the maximum falls in the preceding or in the subsequent interval. This depends on the particular sampling used. The determination of the exact location of the limit point is arbitrary, since

it is only based on the sample points, but it is important for the qualitative states generated, since the distinctive time points are placed by QSIM in correspondence to the limit points. Therefore, the fact that two variables have limit points in the same interval or in different intervals corresponds to the generation (or acceptance) of a different number of qualitative states.

We devised an algorithm working on all the qtl_i, which resolves the attribution of the minima/maxima position to an interval. It is based on the assumption that the possibility to attribute limit points of different variables to the same interval could show a relationship among these variables. This relationship would be lost if we consider their trajectories as independent. For instance, in the case of a variable and its derivative, if we attributed the maximum of the variable to a time interval adjacent to the one of the zero of its derivative, then the constraint of qualitative derivation existing among them would not be satisfied.

However, our algorithm might synchronize unrelated variables: this does not mean that it forces a relationship (possibly non-existent) among the synchronized variables, since the attribution of the limit point to one of the two possible intervals is arbitrary. The synchronization algorithm is one way to solve the ambiguity of the assignment of the position in time of the limit point. Its effectiveness comes from our assumption that the duration of the simulation step is very short with respect to the characteristic times of the modeled variables, so that the probability for two unrelated limit points to be contemporary is negligible.

Now, let us continue the description of the simulation process. The qualitative values so obtained for the shared variables are used as additional constraints to generate the next states the system can reach from the already identified ones. Therefore, the number of new states is cut by information coming from quantitative simulation. Then, the numerical simulation is started again, and it runs until a new qualitative state is reached by a shared variable.

This process is iterated until no more new states can be generated, or the limits imposed by the user are reached.

E. Summary

The QQSIM approach is useful whenever:

- the numerical model of a complex system is already available;
- there is the need of making some modifications to this complex system, or to add parts to it;
- the modifications or the added subsystems can be represented by qualitative, constraint-based models;
- the user would like to have an envisioning tree describing all the possible states the whole system reaches given some initial conditions.

The numerical information is used as a filter for the envisioning activity, thus contributing to reduce the combinatorial explosion affecting this type of simulators.

The QTL-process defines a relationship between a landmark of a shared variable and the value it takes on at the limit point. Therefore, the envisioning tree may contain states enriched by information about the actual values of the shared variables.

Time quantification also has a special relevance, since the qualitative time axis is common to all the qualitative variables and a total order of the time points at which they reach their limit points is defined on it. QQSIM establishes a relationship between real and qualitative time for the states where shared variables undergo a variation. Therefore, it is possible to have an interval-valued time measure for the unshared variables. This is also a significant enhancement with respect to the standard QSIM approach.

VI. Multimodel Integration[3] (MI)

A. Overview

Simulation methodology has developed concepts to model complex systems over multiple levels of abstraction [37], [38], [39]. Ören [40] has developed the definition of *multimodel*[4] to formalize models containing several submodels, only one of which is put into effect at any time. Other groups in the AI community have also addressed the use of multiple models to support multi-level reasoning architectures [24], [41], [42]. Cellier [43] developed an approach to combined continuous/discrete event models implemented in a GASP language extension. Praehofer [44] extended the Discrete Event System Specification (DEVS) [8] to provide a formalism and a simulation environment for specifying combined continuous/discrete event models. In this section, we build on these developments by providing a methodology and formalism for developing multi-level, cooperative models of physical systems of the type studied in qualitative physics. The formalism should help to build reasoning and simulation systems that use multiple models at different levels.

We will use a system of boiling water to illustrate our methods (see Fig. 12.

[Figure 12 about here.]

Although at first glance it appears too simplistic, the boiling water system is appropriate for demonstrating a wide range of discrete and continuous behaviors as well as levels of abstraction. All models for computer simulation are constructed to answer a certain class of questions. With our multimodel approach, we are capable of answering a larger number of questions than with a single-level model. For instance, the question "How long will it take for the pot to boil over?" requires a numerical answer whereas "What is the next step after water starts heating?" involves a qualitative answer such as "If the system is in the phase *heating*, and the control knob is turned off then the next phase will be *cooling*. However, if the water temperature reaches 100 then the water starts *boiling*." We present a method that permits this kind of multi-level reasoning. Fishwick and Zeigler [2] have recently discussed a method

[3]The study described in this section is an extension of material drawn from two sources [36], [2].

[4]Our concept of *multimodel*, as in the title of this paper, includes multiple independent models as well as models containing submodels in a hierarchy. It is a more general notion than Ören's *multimodel*.

for linking the heterogeneous level coupling concept within a DEVS framework.

B. Combined Models

Taking the cross product of time and state in terms of two possible values ("discrete" and "continuous") suggests four possible model types. The *Discrete Event* model type has continuous time and a discretized state space. A *Discrete Time* model has a discrete time space with equal time intervals (the state space may be either discrete or continuous). A *Continuous* model has continuous time and space. Table I displays these combinations with example model formalisms for each. What about *continuous* events? In the simulation literature [45], [46], one finds reference only to discrete events. Continuous events might be defined in terms of the start and end of an arbitrary numerical integration interval. However, this concept is not adequate since it depends on a simulation process, and is not an intrinsic characteristic of the model. It seems that events, by their very nature, are discrete since they map to cognitive and linguistic concepts connected with the processes that we model. In the next section, we attempt to provide a conceptual framework for understanding discrete events.

[Table 1 about here.]

A *combined* model combines two or more of the above model types. For instance, a combined discrete event/continuous model has two distinct model types: a discrete event model and a continuous model. These two models are coupled with discrete events [43], [44].

C. Multimodels

Consider a pot of boiling water on a stovetop electric heating element. Initially, the pot is filled to some predetermined level with water. A small amount of detergent is added to simulate the foaming activity that occurs naturally when boiling certain foods. This system has one input or control – the temperature knob. the knob is considered to be in one of two states: on or off (on is $190°C$; off is α - ambient temperature). We make the following assumptions in connection with this physical system:

1. The input (knob turning) can change at any time. The input trajectories are piecewise continuous with two possible values (ON,OFF).
2. The liquid level (height) does not increase until the liquid starts to boil.
3. When the liquid starts to boil, a layer of foam increases in height until it either overflows the pot or the knob is turned off.
4. The liquid level decreases during the heating and overflow phases only.

To create a mathematical model, we must start with data and expert knowledge about the domain. If enough data can be gathered in a cost effective way then our model engineering process will be simplified since we will not have to rely solely on heuristics to identify the model. By analyzing a pot of boiling water we may derive simple causal models whose individual transitions may be $knob\_on \Rightarrow water\_getting\_hotter(1.0)$ or $water\_getting\_hotter \Rightarrow water\_boiling(0.75)$ where numbers in parentheses are certainty factors. An important facet of system modeling is that we choose certain modeling methods that require a categorization of informally specified system components. Key components of any system model are *input*, *output*, *state*, *event*, *time* and *parameter*. Different modeling methods include these components in different ways. For instance, an FSA focuses on state-to-state transitions with input being labeled on each arc. A dataflow model, on the other hand, focuses on the transfer function between input and output. Homogeneous model refinement [2], [3] is the process of refining models of the same type. For instance, we might represent the boiling water system using a hierarchy of finite state automata (see Fig. 13). The three levels are labeled FSA-1, FSA-2 and FSA-3.

[Figure 13 about here.]

Heterogeneous refinement takes homogeneous refinement a step further by loosening the restriction of equivalent model types. We might have a Petri net at the high abstraction level and we may choose to decompose each transition into a block graph so that when a transition fires within the Petri net, one may "drop down" into a functional block level. For the FSAs in Fig. 13 we choose to represent each state as a continuous model. Specifically, each state will define how three state variables, T (temperature), H_w (height of water), and H_f (height of foam on the top of the water) are updated. In all cases, $H_f \geq H_w$. The end result will eventually be a multi-level model that will be coordinated by the FSA hierarchy.

Fig. 14 displays a block diagram of *heating* within the *heating* state. Proper coupling is essential in heterogeneous refinements. That is, it must be made clear how components at one level match components at the higher level. Note, in Fig. 14, the transfer function taking the ON/OFF input detected by the FSA and converting these input values to temperature values for the block network. Specifically, the block labeled 'F' performs the mapping from 'ON/OFF' to real-valued temperatures $\alpha \leq T \leq 100$. Due to the latent heat effect, T of water cannot exceed 100 unless all the water has vaporized. After all of the water has turned to steam, the temperature increases beyond 100; however, the system passes to the underflow state in our model since $H_w = 0$.

[Figure 14 about here.]

The low-level continuous models $M_1, \ldots, M_6$ are defined as follows:[5]

1. (M_1) COLD: $T = \alpha$, $\dot{H_w} = 0$, $\dot{H_f} = 0$.
2. (M_2) HEATING: $\dot{T} = k_1(100 - T)$, $\dot{H_w} = 0$, $\dot{H_f} = 0$.
3. (M_3) COOLING: $\dot{T} = k_2(\alpha - T)$, $\dot{H_w} = 0$, $\dot{H_f} = -k_3$.
4. (M_4) BOILING: $T = 100$, $\dot{H_w} = -k_4$, $\dot{H_f} = k_5$.

[5]Models M_2 and M_3 exhibit first order exponential behaviors and are, therefore, rough approximations of the actual boiling water system.

5. (M_5) OVERFLOW: *same as BOILING* with constraint $H_f = H_t$.
6. (M_6) UNDERFLOW: $T = undefined$, $H_w = H_f = 0$.

The system phase is denoted by Φ and the state variables are:

- T: temperature of water.
- H_w: height of the water.
- H_f: height of the foam.

Note that the continuous models share a common set of state variables. However, in general state variables may be different for each M_i model.

There are also some constants such as H_t for the height of top of pot, H_s for the starting height of water when poured into the pot; and k_i rate constants. The initial conditions are: $\Phi = cold$, $T(0) = \alpha$, $H_w(0) = H_f(0) = H_s$ and $knob = OFF$. By including the functional block knowledge, we create one large model called COMBINED that is defined as the fully expanded FSA-3 level with each state containing a block model (as in Fig. 14).

D. Summary

The multimodel approach is one where models of different types are connected together in a "seamless" fashion. This approach yields more choices to the model designers who are now able to pick a model type to support a level of system abstraction. High levels of abstraction (in Fig. 13, for instance) can be used to reason about the system, whereas low levels (defined by differential equation sets) are useful for results requiring a smaller amount of granularity in terms of specification and explanation.

VII. Relating the Approaches

We have described four methods for promoting the modeling of dynamical systems and reasoning about them at different levels. Each approach addresses the multiple-model problem from a different perspective. These perspectives are outlined in table II so that individual approaches can be better compared and contrasted. Note the following terminology and abbreviations used in table II. The approaches (VSRM,FM,QQSIM,MI) appear in the same order presented within the paper. All physical phenomena can be studied with regard to their geometry (connectivity) or dynamics. Dynamic models all have the following components: (1) state/event space, (2) time space, (3) parameter space, (4) input/output space and (5) the type of relations or functions used to link the first four spaces together. This categorization is in accordance with the basic concept of system [47]. The components of a geometric model are the representational structure (struc) it uses and operations (oper) provided on this structure. By cross-correlating the four approaches with their dynamic and geometric facets, we can obtain a grasp of how the four methods are situated with respect to one another. The term *symbolic* refers to a nominal value such as a proposition in logic or natural language expression, *real* refers to a ratio variable implemented as a floating point value, *interval* refers to an interval with real-valued endpoints, and *fuzzy* refers to a fuzzy number.

[Table 2 about here.]

Now, let's discuss each approach and its unique contribution. VSRM tackles the problem of representing spatial dynamics and geometry. The geometry is represented in a symbolic array. The spatial dynamics is represented and reasoned about by transforming this array in accordance with predictions derived from production rules. FM advocates the use of causal graphs as the primary medium of dynamic representation since causal graphs often correspond to the purposes or goals that a system is designed to achieve. QQSIM represents two levels of abstraction: qualitative and quantitative. During simulation, ambiguity of relationships among state variables and parameters at the qualitative level is reduced by a search at the quantitative level. MI does not focus, particularly, on a given modeling method. Instead, the emphasis is in how to integrate existing models so that one can switch levels during reasoning and simulation.

What are the key differences among the four approaches? There are three items that reflect a distinction:

- *Measurement:* At what level is a variable known? Does a value come from a sensor or what a human communicates in natural language? If a variable can be physically measured then we need to know the level of granularity associated with the measurement. Often, the scales of nominal, interval and ratio suffice. Nominal scales are used in AI to represent propositional knowledge about devices assuming that physical measurements are not obtainable or too expensive. If data are available in the form of floating point values (often through sampling using an analog/digital interface) then we will use these. The issue, though, is not always one of expressiveness and measurement scale. It is often computationally more efficient to compute values through logical inference than through numerical methods. Computational efficiency is not always a result of using qualitative methods, though — when we make the functional relationships among state variables ambiguous (by saying, for instance, that a function is monotonically increasing or decreasing without any further specification) then we are forced to search over a pre-defined function space that is subject to combinatorial explosion. In any case, we have learned that one must be careful about the level chosen in representing variables; in determining what measurement scales are appropriate. VSRM uses a fairly high level scale – nominal – since it uses only symbolic knowledge. The other three approaches admit all three scales. Both VSRM and FM do not explicitly represent a time line[6], so trajectory information for states are not obtainable. The QQSIM and MI approaches have time bases. These focus more on *simulation* and less on *reasoning* unlike VSRM (which supports *reasoning* about processes) and FM (which supports *cal-*

[6] In FM, work in another application domain endows FM with the capability to reason over time [48].

culating new states from old ones).

- *Geometry & Dynamics:* Comprehensive approaches to systems study will eventually require representations that combine geometry and dynamics. VSRM is a push in this direction whereas the other approaches focus completely on dynamics. Multiple modeling approaches should attempt to incorporate both types of models in future since this enlarges the base of questions that can be asked of a system. We view the convergence in dynamical and geometric representations as being similar to the ongoing research in integrating computer aided design (CAD) and computer aided manufacturing (CAM) modeling techniques.
- *Perspectives & Levels:* QQSIM proves that symbolic and numerical perspectives need not be considered in exclusion of each other. To be most productive, future toolkits for analyzing systems must incorporate more than one level. The QQSIM and MI approaches address this issue of multiple levels by explicitly supporting more than one representation language, while the FM approach suggests one specific modeling language that admits multiple levels. The VSRM approach does not address this issue.

In a nutshell, VSRM shows how predictive knowledge about dynamics and a "depictive" representation of geometry can be used together for reasoning about spatial dynamics. FM promotes the organization, based on the notion of teleology, of knowledge about causal dynamics of engineered systems since this bears a close relationship to the way that humans design and reason about such systems. QQSIM shows how qualitative and quantitative levels of representation and simulation can be integrated. MI provides a formal definition of how to integrate existing model types together into a unified framework with multiple abstraction levels.

One lesson to be drawn from this paper is that building the model of a system to support both simulation and reasoning can be approached from two directions. One is top-down, in which a hierarchical and modular model is developed, whose levels reflect a system-subsystem decomposition organized around their intended functions and designed behaviors to achieve these functions (FM) or a state-space decomposition organized in terms of finite state automata (FSA) whose states expand into other FSAs or block models (MI). The other is bottom-up, in which spatial subsystems are represented in terms of their geometry and predictive rules that govern their behavior (VSRM) while non-spatial subsystems are represented in terms of relevant qualitative/quantitative variables and constraints (QQSIM). The top-down approach provides a framework for guiding simulation, controlling its level of detail, and interpreting simulation results in terms of intended functions and expected state transitions. The bottom-up approach is useful in detecting new behaviors resulting from unforeseen interactions (interactions among state variables or spatial interactions among components). Hence a multimodel that combines both will be well suited to simulating and reasoning about both designed and unexpected behaviors of a system.

We will now discuss two of the application examples mentioned in the paper to illustrate how the four approaches could be used together. Consider a variation of the gear example in Fig. 1, in which there are two pins between which the protruding tooth of gear-2 oscillates as the driving gear (gear-1) alternatingly rotates in opposite directions. It can be easily imagined as part of a more complex mechanical device such as a clock. The FM approach allows a model of such a device to be built, so that its purpose is linked to the functions of its components and mathematical computations (such as those involving gear ratios) are modularly organized within this teleological framework. Orthogonally, a hierarchical FSA model of this device can also be built as per the MI approach, which facilitates reasoning about state transitions of the device at different levels of abstraction. If a capability for detailed simulation is required however, the model also needs to represent how each component constrains the motions of others. Geometric constraints due to component shapes can be captured in a VSRM. But differential constraints among dynamic variables that take numerical values (e.g., torque, velocity, and acceleration) or qualitative values (e.g., the position of the protruding tooth of gear-2 may have only two qualitative values of interest: touching pin-1 or touching pin-2) are best represented using the QQSIM approach. These four approaches together thus allow a comprehensive model of the device to be built.

Consider the automobile cruise control (ACC) system of Fig. 5. Its components have intended purposes (functionalities). So the functional modeling approach (FM) can provide a model which links the functions of the system to the functions of its components and their behaviors. As discussed in Section 4, these behaviors can be further refined into mathematical equations, thereby providing a framework for organizing mathematical models. However, reasoning about the function of a system component with spatial behaviors —like the throttle actuator— requires a model such as the visuo-spatial reasoning model (VSRM). Similarly, there is sometimes a need to reason not only about the intended function of a system or a component, but also about all possible states it can reach from certain initial conditions. If such reachability information (or "envisionment") needs to be generated, qualitative and numerical simulation models, and their integration as discussed in Section 5 (QQSIM), should be made parts of the overall model. The multimodel integration approach in Section 6 (MI) provides an orthogonal modeling approach which supports user queries at different levels of abstraction. System components, like the control electronics module of the ACC, that involve both discrete and continuous processes can be modeled with this approach. Thus, a complex system such as the ACC can be modeled comprehensively by (1) developing an overall functional model (FM), (2) creating visuo-spatial models of spatial subsystems (VSRM), (3) specifying constraints relating numerical and qualitative parameters and state variables and developing a corresponding qualitative/quantitative simulation

model (QQSIM), and (4) using the multimodel integration approach to integrate a hierarchy of model abstractions (MI).

We have presented four different approaches to improve modeling of dynamic systems. These four models are drawn from different disciplines: Systems and Simulation Theory, Artificial Intelligence and Cognitive Science. This is an interdisciplinary effort that began at a joint conference. We have now taken the next logical step — constructing a paper that presents methods, in service of the common goal of modeling, from researchers in each discipline. As we collaborate in this manner, we find ourselves closely examining each others' terminology, methods and motivations. For the future, we hope that this kind of interdisciplinary effort will foster further collaborations among ourselves as well as other workers in the field.

Acknowledgments

Dr. Fishwick would like to acknowledge partial research support from a Florida High Technology and Industrial Council (FHTIC) grant entitled "Real Time Computer Animation for Interactive Visualization." Material in Section 3 is based on research conducted at the Laboratory for Artificial Intelligence Research, Ohio State University, and supported by grants to the laboratory from ARPA (under AFOSR contract F-49620-89-C-0110) and British Petroleum. Dr. Narayanan acknowledges Hitachi Advanced Research Laboratory for facilitating the preparation of this paper. The analysis and implementation of the cruise control system (in the section on Functional Modeling) was carried out under Dr. Sticklen's supervision by Mr. Ahmed Kamel. Dr. Sticklen's research at Michigan State is supported by ARPA (ARPA 8673), the NSF Center for High Speed, Low Cost Polymer Composites Processing at MSU, the McDonnell Douglas Research Foundation, the State of Michigan Research Excellence Fund, and generous equipment support from Apple Computer. Research of Dr. Bonarini (QQSIM) has been supported by the Italian National Research Council (CNR) as part of the Progetto Finalizzato Informatica e Calcolo Parallelo. The research has been done with Vittorio Maniezzo, and QQSIM has been implemented by Massimo Pianciamore, who also gave contributions to the theoretical aspects.

References

[1] P. A. Fishwick, J. W. Rozenblit, and B. P. Zeigler, eds., *Second Annual AI, Simulation and Planning in High Autonomy Systems Conference.* Los Alamitos, CA: IEEE Computer Society Press, 1991.

[2] P. A. Fishwick and B. P. Zeigler, "A Multimodel Methodology for Qualitative Model Engineering," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, no. 1, pp. 52 – 81, 1992.

[3] P. A. Fishwick, "An Integrated Approach to System Modelling using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, no. 4, pp. 307 – 330, 1992.

[4] P. A. Fishwick and R. B. Modjeski, eds., *Knowledge Based Simulation: Methodology and Application.* Springer Verlag, 1991.

[5] P. A. Fishwick and P. A. Luker, eds., *Qualitative Simulation Modeling and Analysis.* Springer Verlag, 1991.

[6] L. E. Widman, K. A. Loparo, and N. R. Nielsen, *Artificial Intelligence, Simulation and Modeling.* John Wiley, 1989.

[7] M. S. Elzas, T. I. Oren, and B. P. Zeigler, *Modelling and Simulation Methodology: Knowledge Systems' Paradigms.* North Holland, 1989.

[8] B. P. Zeigler, *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press, 1990.

[9] D. S. Weld and J. de Kleer, eds., *Readings in Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann, 1990.

[10] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, pp. 289–338, 1986.

[11] B. Faltings, "A symbolic approach to qualitative kinematics," *Artificial Intelligence*, vol. 56, pp. 139–170, 1992.

[12] A. Gelsey, "The use of intelligently controlled simulation to predict a machine's long-term behavior," in *Proceedings of the Ninth National Conference on Artificial Intelligence*, (Menlo Park, CA), pp. 880–887, AAAI press, 1991.

[13] L. Joskowicz and E. P. Sacks, "Computational kinematics," *Artificial Intelligence*, vol. 51, pp. 381–416, 1991.

[14] N. H. Narayanan, *Imagery, Diagrams and Reasoning.* PhD thesis, The Ohio State University, Columbus, OH, 1992.

[15] N. H. Narayanan and B. Chandrasekaran, "Reasoning visually about spatial interactions," in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, (Mountain View, CA), pp. 360–365, Morgan Kaufmann Publishers, 1991.

[16] J. I. Glasgow and D. Papadias, "Computational imagery," *Cognitive Science*, vol. 16, pp. 355–394, 1992.

[17] N. H. Narayanan and B. Chandrasekaran, "Integration of qualitative and quantitative methods in visual reasoning," in *Proceedings of the Second Conference on AI, Simulation, and Planning in High Autonomy Systems*, (Los Alamitos, CA), pp. 272–278, IEEE Computer Society Press, 1991.

[18] V. Sembugamoorthy and B. Chandrasekaran, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems," in *Experience, Memory, and Learning* (J. Kolodner and C. Reisbeck, eds.), Laurence Erlbaum Associates, 1986.

[19] J. Sticklen, B. Chandrasekaran, and W. Bond, "Distributed Causal Reasoning," *Knowledge Acquisition*, vol. 1, pp. 139–162, 1989.

[20] A. Goel and B. Chandrasekaran, "Functional Representation of Designs and Redesign Problem Solving," in *Proceedings of IJCAI-89*, 1989.

[21] W. Punch, K. Keller, and J. Sticklen, "Domain-Specific Software Architectures and their use in Developing Complex AI Systems," in *Proceedings of IEEE Workshop on AI Applications*, (Miami), 1991.

[22] D. Allemang, *Understanding Programs as Devices.* PhD thesis, The Ohio State University, 1990.

[23] A. Keunke, *Machine Understanding of Devices: Causal Explanation of Diagnostic Conclusions.* PhD thesis, The Ohio State University, 1989.

[24] L. Chittaro, C. Constantini, G. Guida, C. Tasso, and E. Toppano, "Diagnosis Based on Cooperation of Multiple Knowledge Sources," in *Proceedings of the Model Based Diagnosis International Workshop*, 1989.

[25] D. Franke, "Representing and Acquiring Teleological Descriptions," in *Proceedings of the 1989 Workshop on Model Based Reasoning (IJCAI 89)*, (Detroit), 1989.

[26] J. Sticklen, A. Kamel, and W. E. Bond, "Integrating Quantitative and Qualitative Computations in a Functional Framework," *Engineering Applications of Artificial Intelligence*, vol. 4, no. 1, pp. 1–10, 1991.

[27] J. deKleer and J. S. Brown, "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, vol. 24, pp. 7–83, 1984.

[28] K. Forbus, "Qualitative Process Theory," *Artificial Intelligence*, vol. 24, pp. 85–168, 1984.

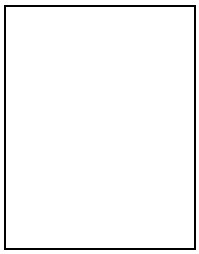
[29] B. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," *Artificial Intelligence*, vol. 24, pp. 169–203, 1984.

[30] R. Davis, "Model Based Trouble Shooting," in *Exploring Artificial Intelligence* (H. Shrobe, ed.), Morgan Kaufmann, 1987.

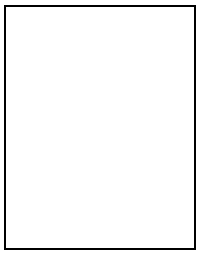
[31] D. Bobrow, ed., *Qualitative Reasoning on Physical Systems.* Cambridge, MA: MIT Press, 1985.

[32] F. E. Cellier and N. Roddier, "Qualitative State Spaces: a Formalization of the Naive Physics Approach to Knowledge-Based Reasoning," in *Second Annual AI, Simulation and Planning in High Autonomy Systems Conference* (P. Fishwick, B. Zeigler,

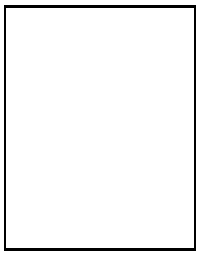
and J. Rozenblit, eds.), (Los Alamitos, CA), IEEE Computer Society Press, 1991.
[33] A. Bonarini and V. Maniezzo, "Integrating Qualitative and Quantitative Modeling," *International Journal of Expert Systems: Research and Applications*, vol. 4, no. 1, pp. 51 – 70, 1991.
[34] B. Kuipers and D. Berleant, "Using Incomplete Quantitative Knowledge in Qualitative Reasoning," in *Sixth National Conference on Artificial Intelligence*, (San Mateo, CA), Morgan Kaufmann, 1988.
[35] D. Berleant and B. Kuipers, "Combined Qualitative and Numerical Simulation with Q3," in *Qualitative Physics '90*, (IDSIA, Lugano, Switzerland), 1990.
[36] P. A. Fishwick, "Heterogeneous Decomposition and Coupling for Combined Modeling," in *1991 Winter Simulation Conference*, (Phoenix, AZ.), pp. 1120 – 1128, December 1991.
[37] P. A. Fishwick, *Hierarchical Reasoning: Simulating Complex Processes over Multiple Levels of Abstraction*. PhD thesis, University of Pennsylvania, 1986.
[38] P. A. Fishwick, "The Role of Process Abstraction in Simulation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, pp. 18 – 39, January/February 1988.
[39] P. A. Fishwick, "Abstraction Level Traversal in Hierarchical Modeling," in *Modelling and Simulation Methodology: Knowledge Systems Paradigms* (B. P. Zeigler, M. Elzas, and T. Oren, eds.), pp. 393 – 429, Elsevier North Holland, 1989.
[40] T. I. Oren, "Dynamic Templates and Semantic Rules for Simulation Advisors and Certifiers," in *Knowledge Based Simulation: Methodology and Application*, pp. 53 – 76, Springer Verlag, 1991.
[41] K. D. Forbus and B. Falkenhainer, "Self-Explanatory Simulations: An Integration of Qualitative and Quantitative Knowledge," in *AAAI*, pp. 380 – 387, 1990.
[42] S. Addanki, R. Cremonini, and J. S. Penberthy, "Reasoning about Assumptions in Graphs of Models," in *Eleventh International Joint Conference on Artificial Intelligence*, pp. 1432 – 1438, IJCAI, August 1989.
[43] F. E. Cellier, *Combined Continuous System Simulation by Use of Digital Computers: Techniques and Tools*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1979.
[44] H. Praehofer, "Systems Theoretic Formalisms for Combined Discrete Continuous System Simulation," *International Journal of General Systems*, vol. 19, no. 3, pp. 219–240, 1991.
[45] T. I. Oren, "Simulation Model Symbolic Processing: Taxonomy," in *Systems and Control Encyclopedia*, pp. 4377 – 4381, Pergammon Press, 1987.
[46] T. I. Oren, "Simulation: Taxonomy," in *Systems and Control Encyclopedia*, pp. 4411 – 4414, Pergammon Press, 1987.
[47] L. Padulo and M. A. Arbib, *Systems Theory: A Unified State Space Approach to Continuous and Discrete Systems*. Philadelphia, PA: W. B. Saunders, 1974.
[48] K. Patzer and J. Sticklen, "Causal Knowledge Sharing in Large Biological Projects: Leveraging the Functional Approach," in *AAAI-92 Workshop on Communicating Scientific Knowledge*, (San Jose, CA), 1992.



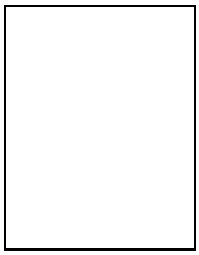
Paul A. Fishwick is an associate professor in the Department of Computer and Information Sciences at the University of Florida. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a senior member of the IEEE and the Society for Computer Simulation. He is also a member of the IEEE Society for Systems, Man and Cybernetics, ACM and AAAI. Dr. Fishwick was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the ACM Transactions on Modeling and Computer Simulation, IEEE Transactions on Systems, Man and Cybernetics, The Transactions of the Society for Computer Simulation, International Journal of Computer Simulation, and the Journal of Systems Engineering.



N. Hari Narayanan is a visiting research scientist at the Advanced Research Laboratory of Hitachi Ltd. His doctoral research was conducted at the Laboratory for Artificial Intelligence Research, Ohio State University. He also holds degrees in Computer Science, Automation, and Electrical Engineering from the University of Rochester, Indian Institute of Science, and Birla Institute of Technology and Science respectively. His main interest is in investigating the role of perceptual representations in reasoning and problem solving, from both computational and cognitive perspectives. His current work is on the problem of behavior hypothesis from device diagrams. He is also interested in qualitative spatial reasoning and model-based approaches to diagnosis and design. Dr. Narayanan organized the 1992 AAAI Spring Symposium on Reasoning with Diagrammatic Representations, and has guest-edited a forthcoming journal special issue on "computational imagery" (Computational Intelligence, Vol. 9, No. 3). He is currently editing a book on reasoning and problem solving with diagrams. He is a member of AAAI and ACM.



Jon Sticklen is Associate Professor of Computer Science and Director of the Intelligent Systems Laboratory, Michigan State University. Following his Ph.D. studies at the Ohio State University, Dr. Sticklen joined the faculty at MSU where he initiated an active research program in knowledge-based systems. Sticklen's major research foci are in integrative approaches to large grain task specific problem solving, function-based reasoning, and the theory of knowledge-based systems. Sticklen and his associates are currently engaged in a wide variety of domain projects including the design and fabrication of polymer composite materials, the modeling of landscape level ecological systems, and the development of decision support software for managing production agriculture in the lower Nile valley.



Andrea Bonarini was born in Milan in 1957. He received his Laurea (Master of Technology) in Electronics Engineering, in 1984, and his PhD in Computer Science in 1989, both from the Politecnico di Milano, Italy. He is Tenurial Assistant Professor at the Department of Electronics and Information of the Politecnico di Milano. Since 1984, he is member of the Politecnico di Milano Artificial Intelligence and Robotics Project. He is founding member of the AI*IA (the Italian Association for Artificial Intelligence), and co-founder of the AI*IA Special Interest Group on Qualitative Reasoning. He took part to several CEC Projects (within the ESPRIT and EUREKA programmes), and to several National Research Projects. His research interests are in the field of Uncertainty Representation, applied to Model-based Diagnosis of industrial plants, Simulation, and Control, with special emphasis on fuzzy-based technologies. He is also working on Reinforcement Learning techniques for the automatic synthesis of Fuzzy-based systems.

List of Figures

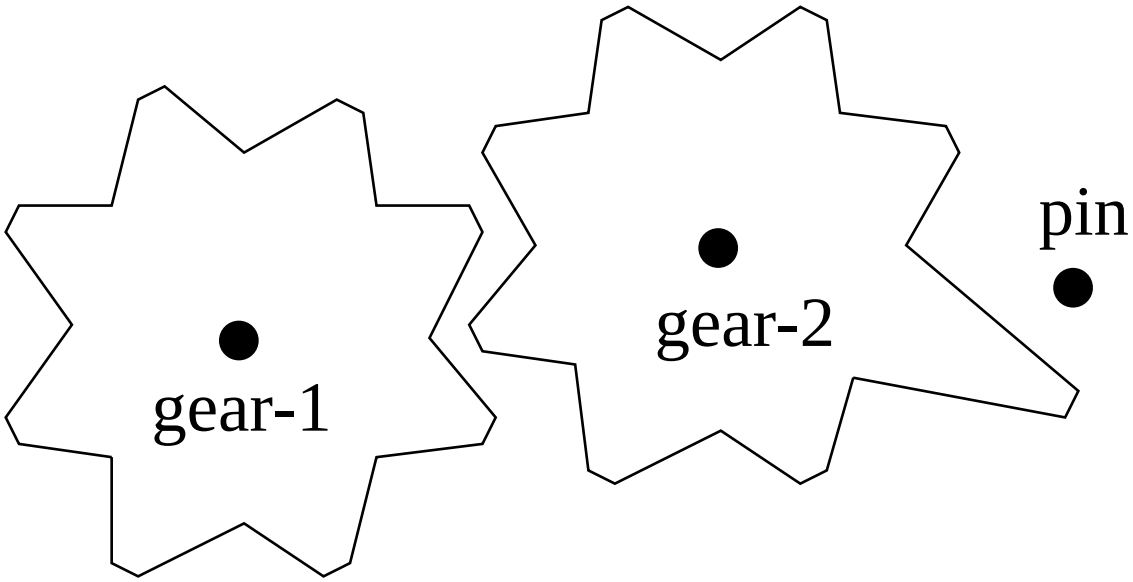


Fig. 1. A configuration of two interacting gears

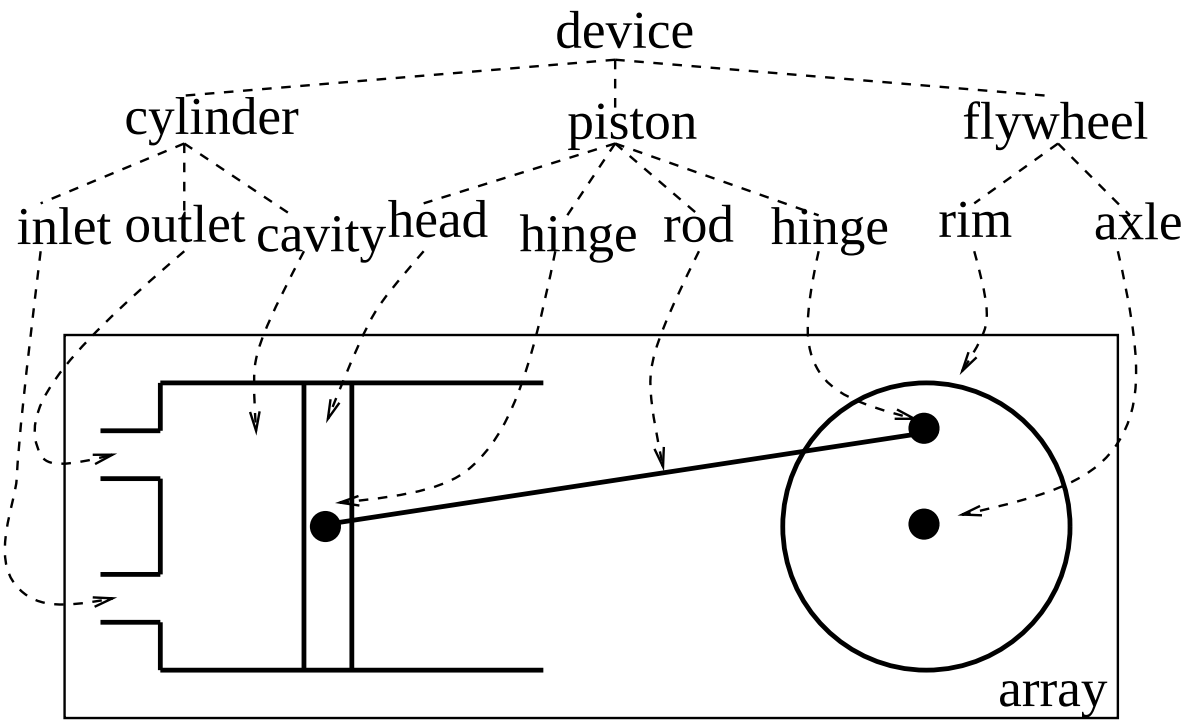


Fig. 2. A diagrammatic representation

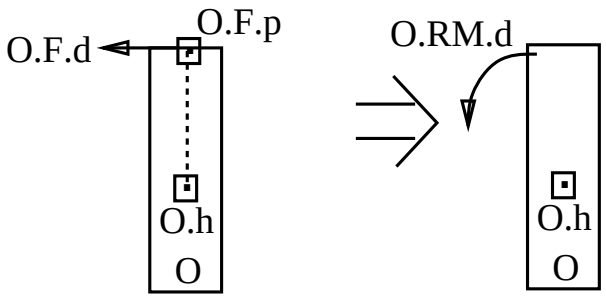
POC	
ED	An impetus acting on a rigid hinged object at a location such that the object's hinge is to the left of the impetus direction can produce a counterclockwise rotation of the object about its hinge.
VC	O.F(O.F.p, O.F.d); Impetus-Direction-Relative-To-Hinge(O.F.d, O.F.p, O.h)=Left;
NVC	Object(O); Rigid(O); Hinged(O);
P	O.RM such that O.RM.h=O.h and O.RM.d=Counterclockwise;

Fig. 3. A visual case

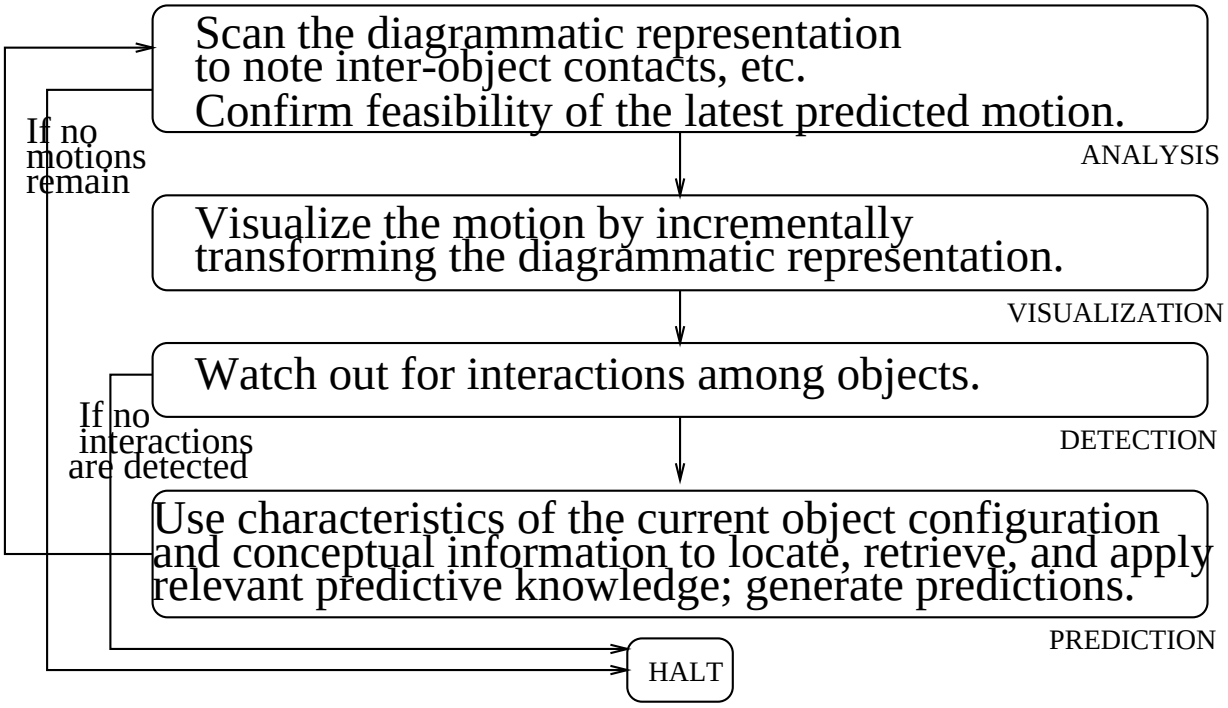


Fig. 4. Stages of visuo-spatial reasoning

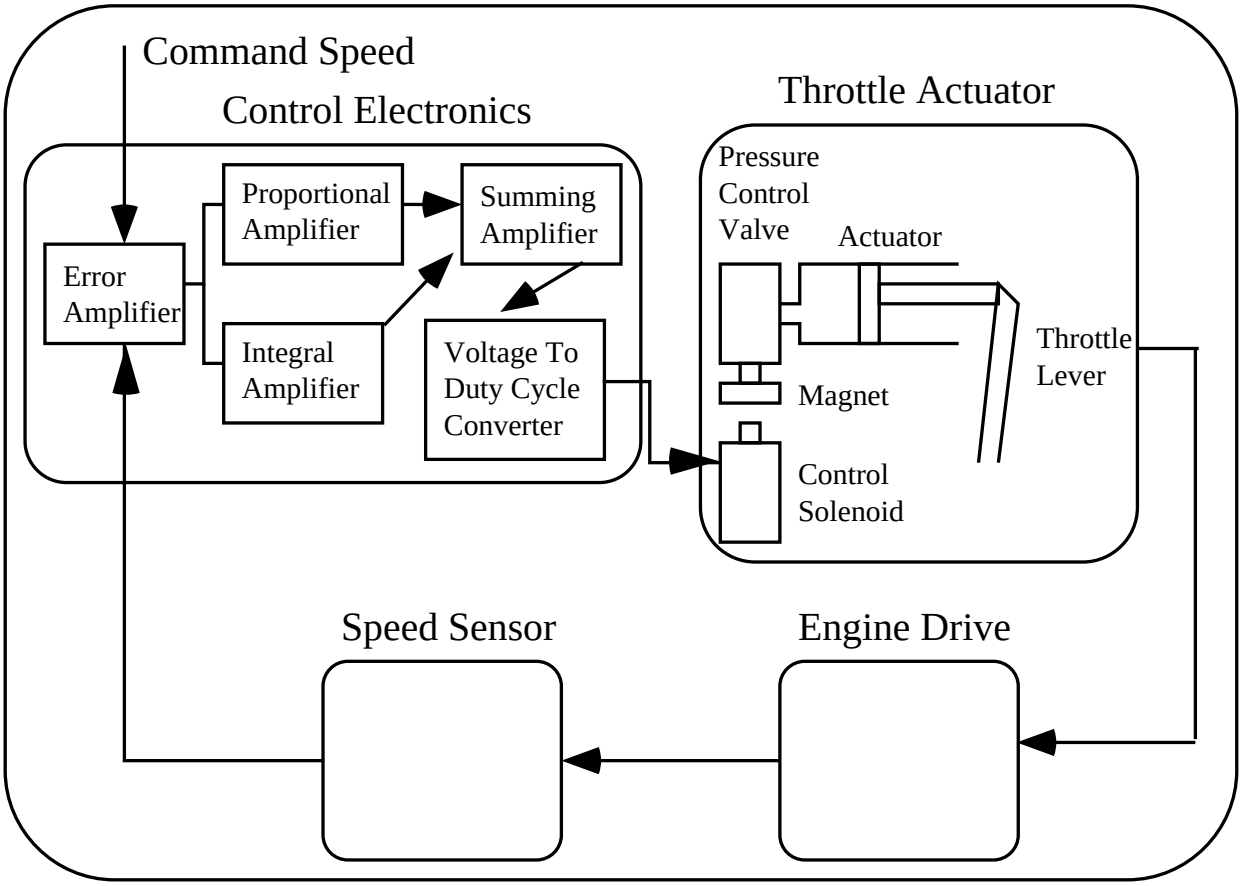


Fig. 5. Cruise control system schematic.

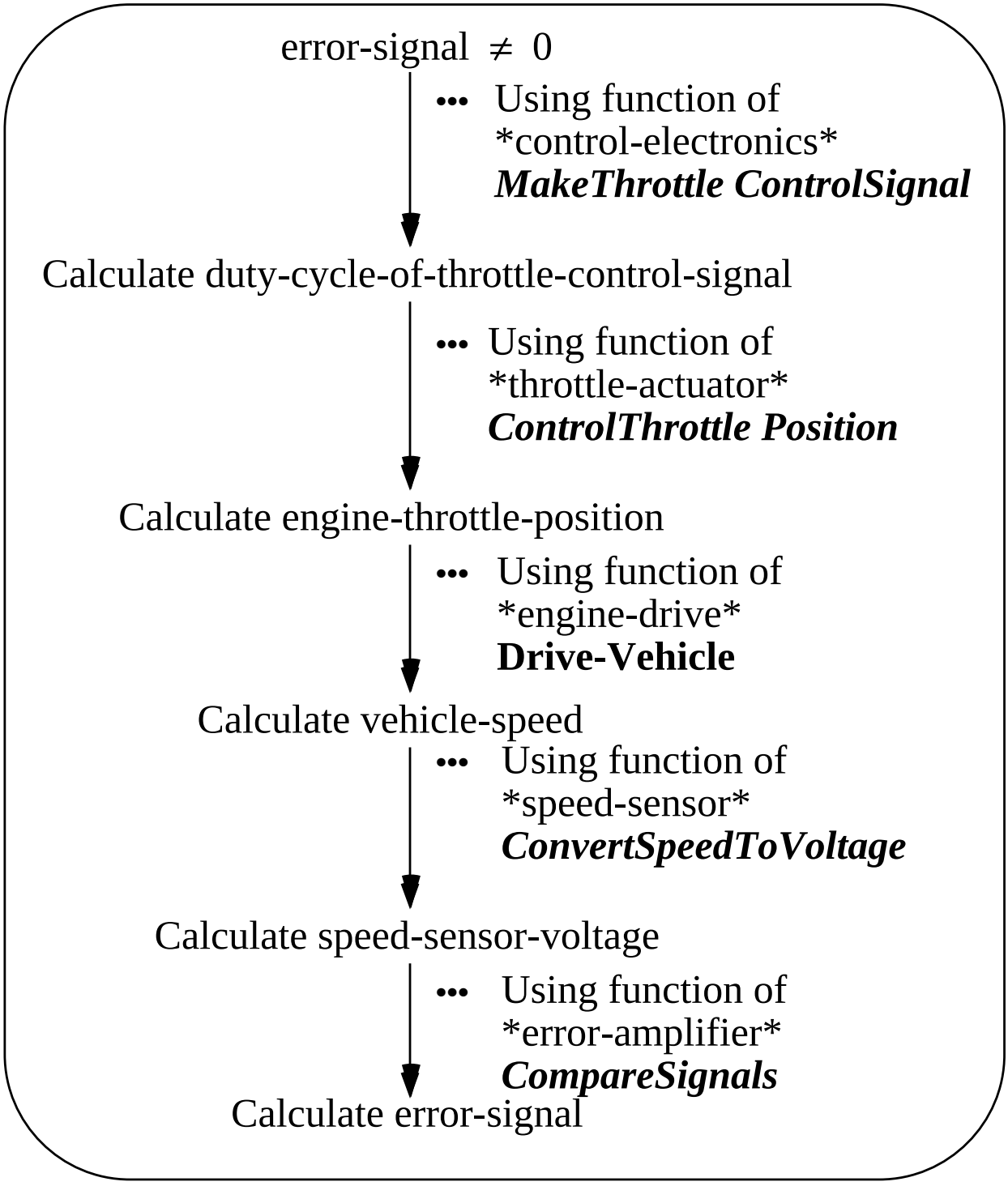


Fig. 6. A top level behavior of the cruise control system.

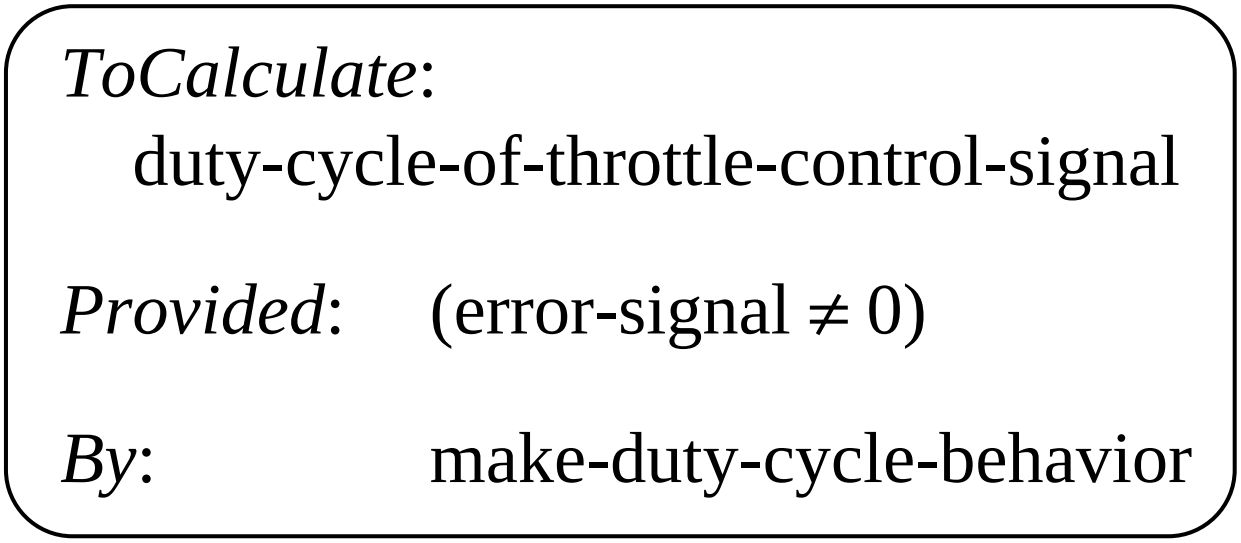


Fig. 7. A function - *MakeThrottleSignal*.

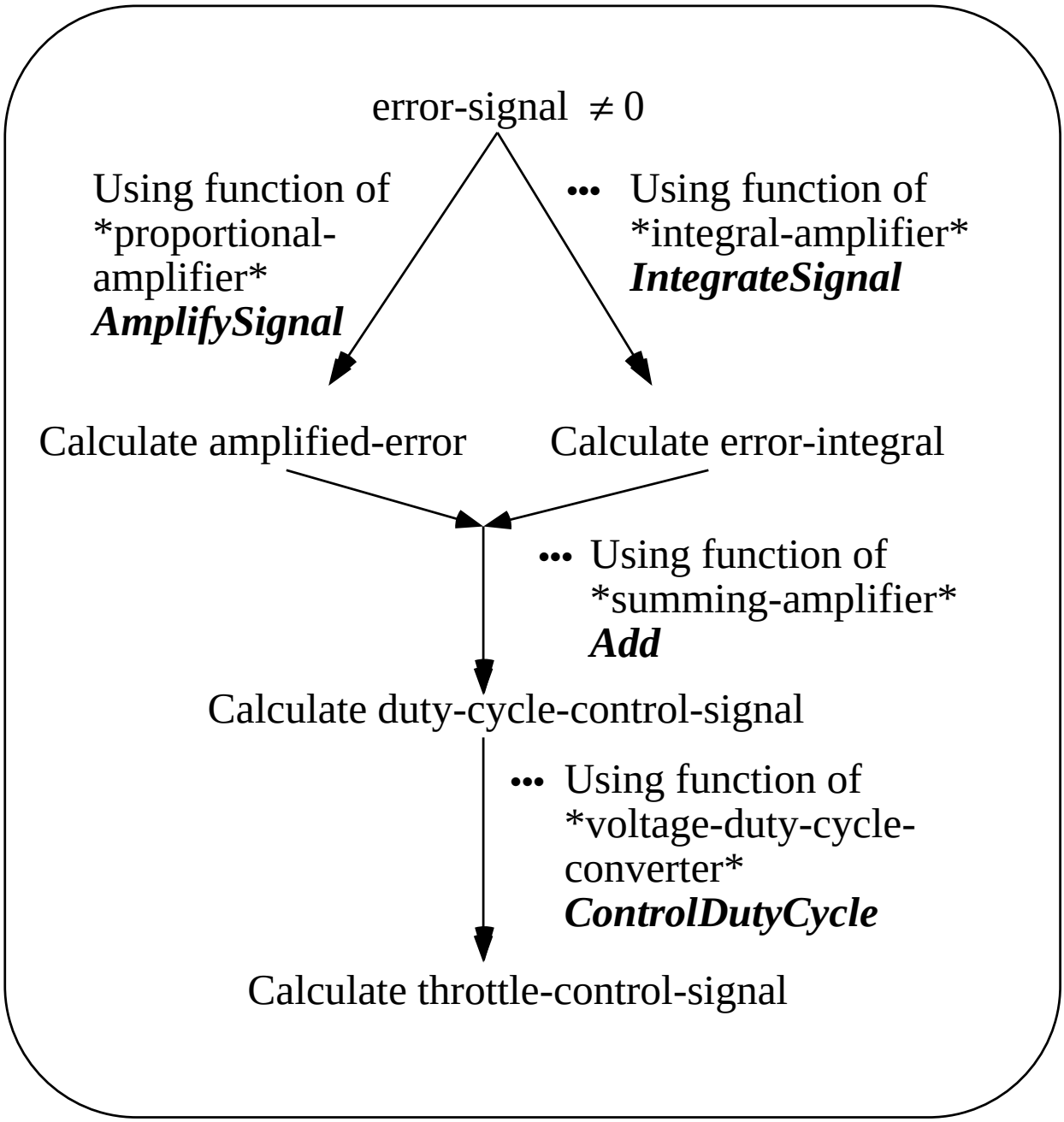


Fig. 8. A behavior - make-duty-cycle.

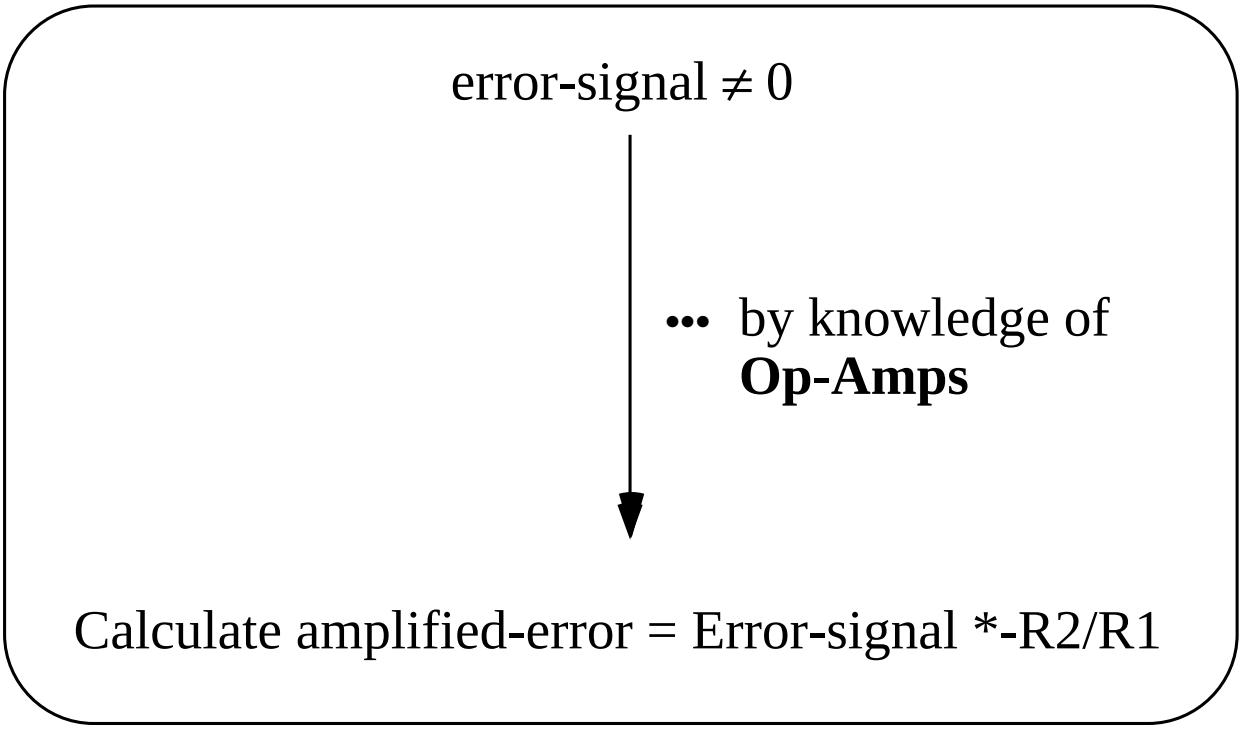


Fig. 9. A low level behavior - *amplify-behavior*.

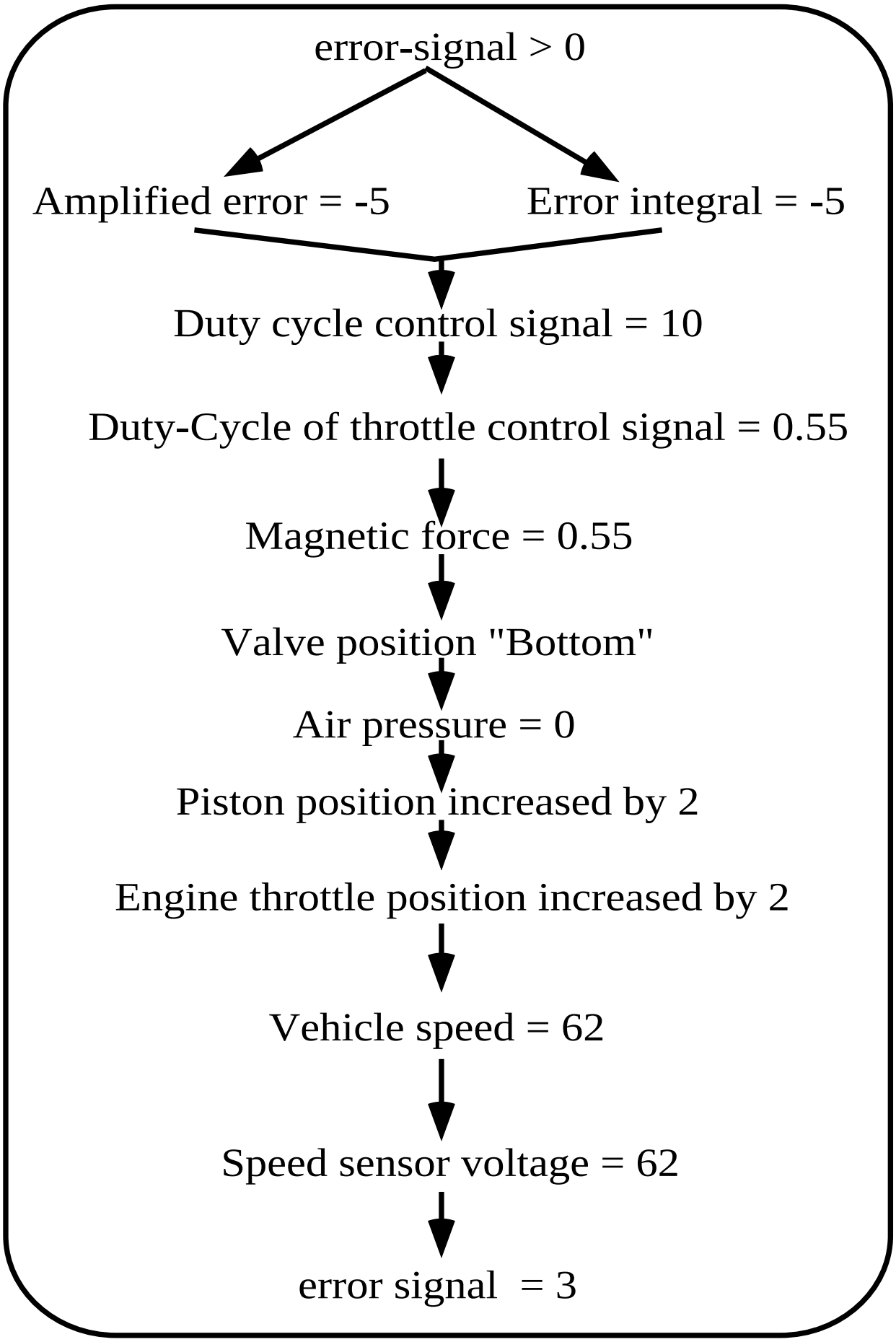


Fig. 10. State variables after one "invocation."

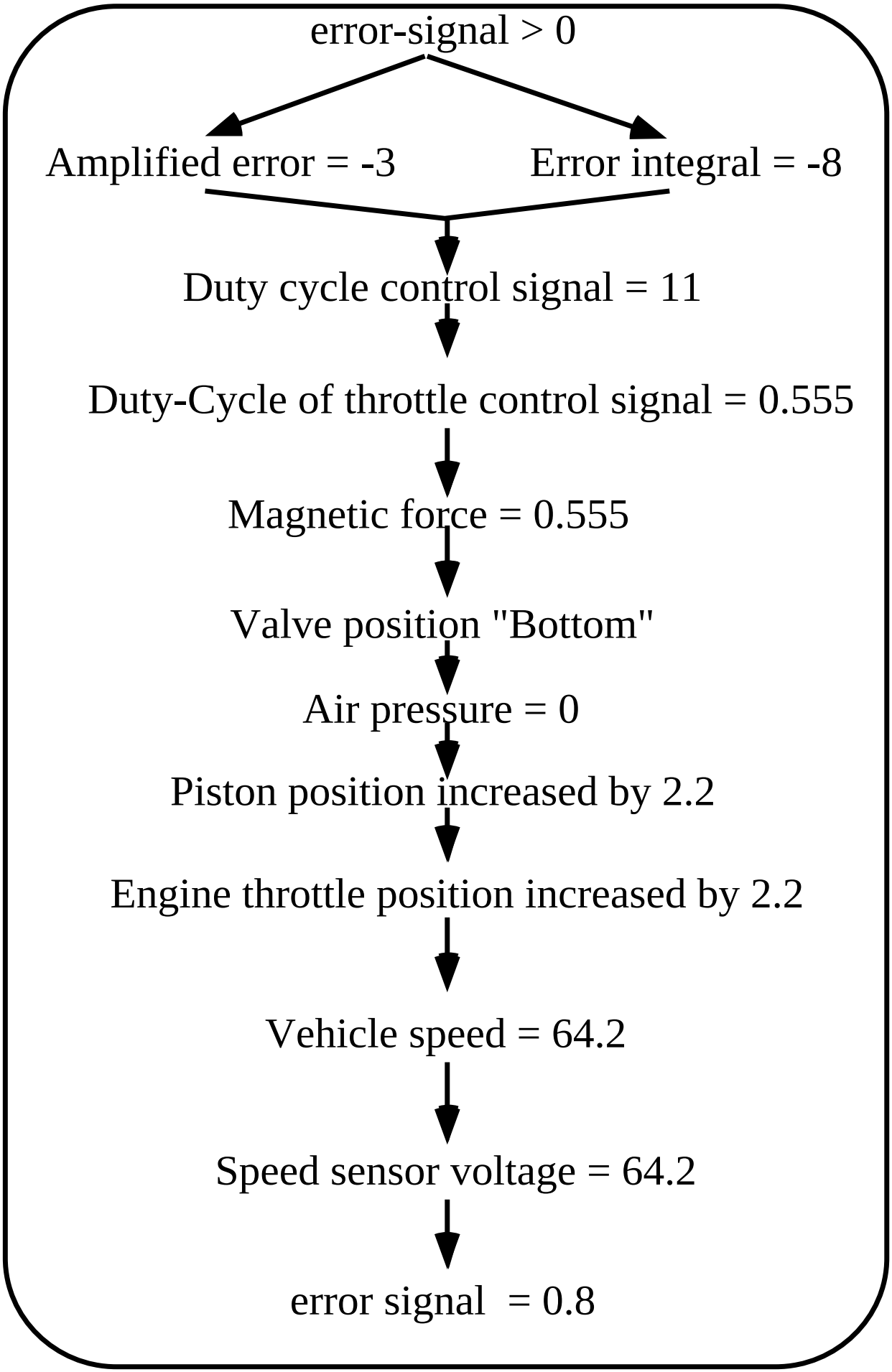


Fig. 11. State variables after two "invocations."

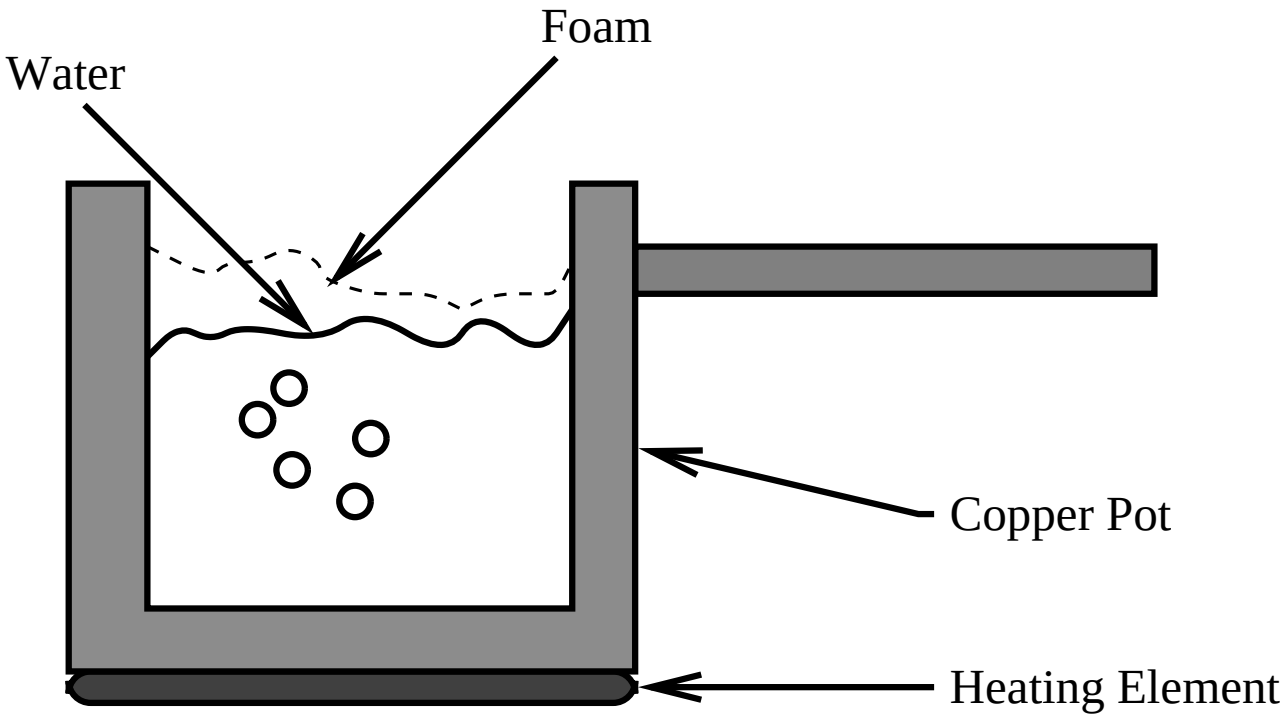


Fig. 12. A pot of boiling water.

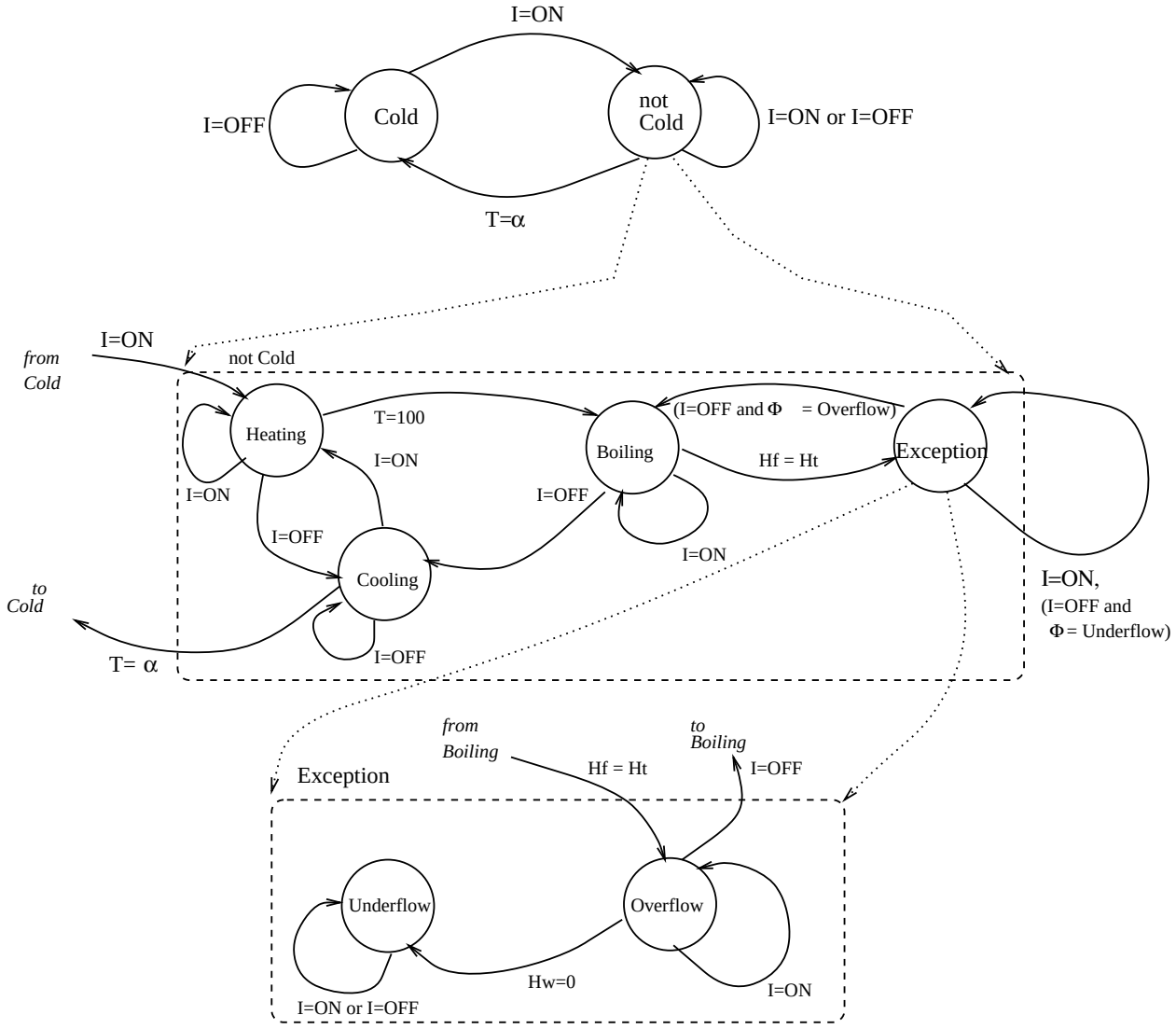


Fig. 13. Homogeneous FSA refinement.

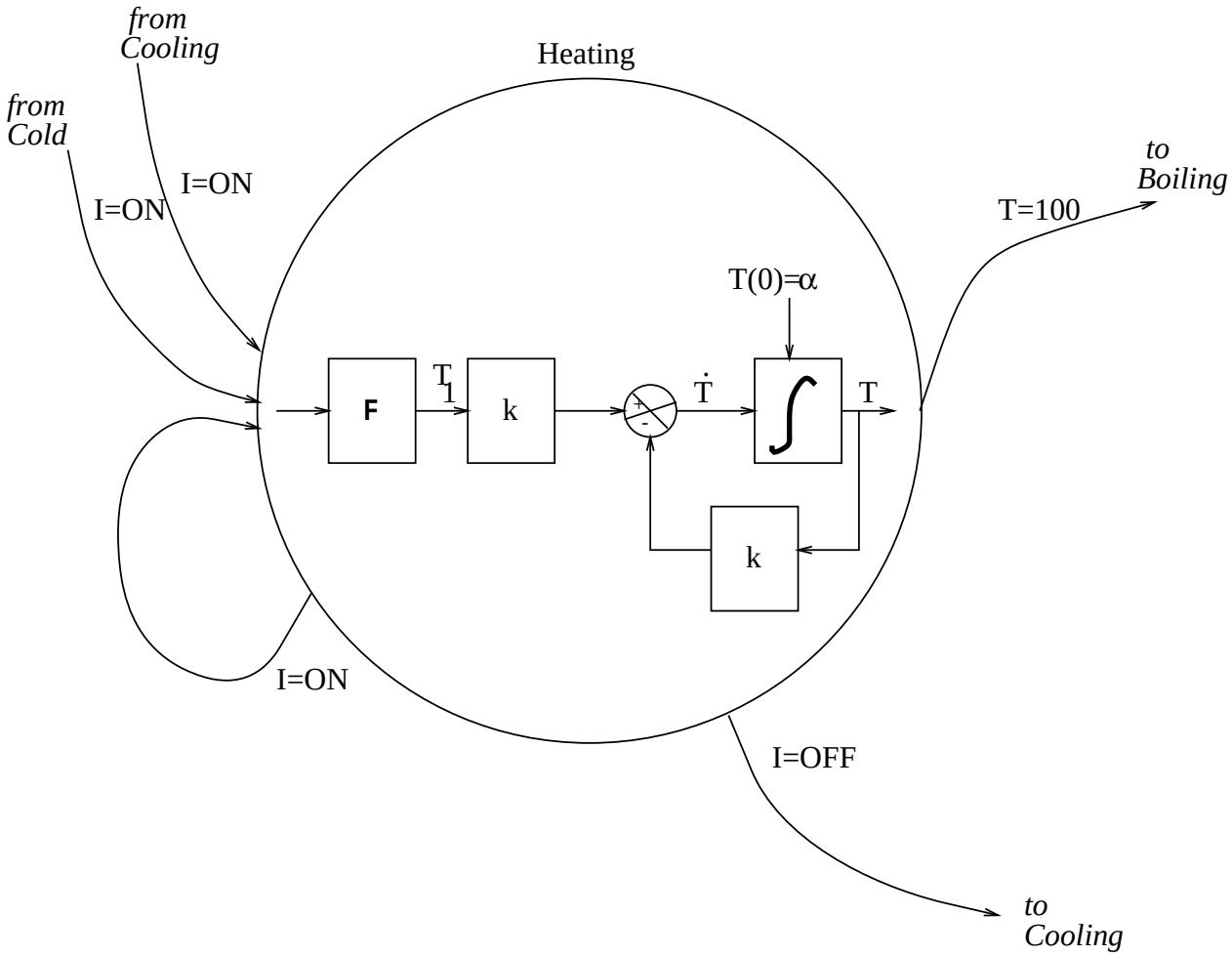


Fig. 14. Decomposition of *heating* state.

List of Tables

TABLE I
SIMULATION MODEL TYPES

	Discrete Space	Continuous Space
Discrete Time	*Discrete Time*	*Discrete Time*
	Difference Equations (with integer states) Cellular Automata Finite State Automata	Difference Equations (with real states)
Continuous Time	*Discrete Event*	*Continuous*
	Queuing Models Digital Logic Models	Differential Equations

TABLE II
CATEGORIZING FOUR APPROACHES

	Dynamic Model					Geometric Model	
Approach	State/Event	Time	Parameter	Relation	I/O	Struc	Oper
VSRM	symbolic	N/A	symbolic	symbolic	symbolic	image	symbolic
FM	real	N/A	real	causal	real	N/A	N/A
QQSIM	real	real	real	f. space	real	N/A	N/A
	interval		real	diffeq	interval	N/A	N/A
MI	real	real	real	automata	real	N/A	N/A
	fuzzy		fuzzy	diffeq	fuzzy	N/A	N/A