A Multifunctional Knowledge–Based System for Engineering

Eduardo F. Morales, L. Enrique Sucar

I.T.E.S.M. – Campus Morelos

Av. Paseo de la Reforma 182–A

Col. Lomas de Cuernavaca,

Temixco, Morelos, 62589, México

Phone: +52 (73) 29–80–84

To be published in: Expert Systems with Applications, Pergamon Press

A Multifunctional Knowledge–Based System for Engineering

Abstract

Knowledge–based systems are normally designed to perform a single task (e.g., diagnosis) for which they are competent, but their knowledge base is rarely used for other tasks, even when they are very similar and in the same application domain. An alternative approach is to build a *knowledge core* that has the basic knowledge for certain domain, on top of which different tasks could be built. In this paper, a multifunctional knowledge–based system for engineering domains, called *MF–KBS*, is described. MF–KBS has (i) a library of engineering components, such as pumps, valves, tanks, etc., associated with mathematical models at different abstraction levels, (ii) several reasoning mechanisms for these models, and (iii) *knowledge operators* capable of performing different tasks with the reasoning mechanisms, such as diagnosis, problem solving, tutoring, etc. Particular applications are build by joining components, producing a global model via composition, and selecting one or more of the knowledge operators. Experiments in a subsystem of an electric power plant with different knowledge operators are described.

1 Introduction

A knowledge–based system (KBS) can be defined as an information system capable of offering solutions to specific problems at a level comparable to that of experts in the field. They have been used in a wide range of applications with relative success [15, 2]. However, once a KBS has been built for a particular task, its knowledge base is very rarely used for other tasks or as bases for the construction of other systems in similar tasks. On the other hand, KBSs have been criticized for having very brittle knowledge, unable to solve problems which differ, even slightly, from the original problems that they were designed for. The inclusion of "deep" models [7] has helped to alleviate both problems but its potential has not been yet fully exploited.

An alternative approach is to build a *knowledge core* that has the basic knowledge for certain domain, on top of which different tasks could be built. This has the advantage that the common knowledge of the domain is integrated in a single knowledge–base that can be utilized for building specific subsystems to perform different tasks.

In this paper, a system called MF–KBS, which uses a knowledge core to perform a wide range of tasks, such as diagnosis or tutoring, is described. MF–KBS uses a library of engineering components, such as pumps, valves, tanks, etc., associated with models at different abstraction levels. Specific applications are built by joining instances of engineering components, creating global models at different abstraction levels via compositional modeling [7]. Once a model has been built, several reasoning mechanism can be applied to it, such as propagation of values and/or simulation, both of which can be qualitative, quantitative, causal or probabilistic. MF–KBS uses *knowledge operators* to perform different tasks using the same knowledge base. Each knowledge operator uses the model constructed via composition and the available reasoning mechanisms to perform a particular task. A knowledge operator can have, as well, its own reasoning mechanisms. For example, a knowledge operator for diagnosis can use propagation of values and ATMS

[5] to perform fault diagnosis. So far we have partial implementations of knowledge operators for diagnosis, problem solving, consulting, reliability analysis and tutoring, and we expect to add more in the future. With them, a user can perform, for instance, diagnosis on a device and learn about its function with a tutoring system without having to construct a different system for each task.

The goal of MF–KBS can be summarized as follows:

Given:

- a set of engineering components associated with models at different abstraction levels,
- a set of different reasoning mechanisms,
- a configuration of a device specified by the user (instances of components linked by the user which are part of the device).

Produce:

- different types of models (qualitative, quantitative, causal, probabilistic) for the device,
- capabilities to reason with each type of model, at different abstraction levels, to support multiple tasks (knowledge operators),
- capabilities to perform different tasks for the device, such as diagnosis, tutoring, consulting, reliability analysis, problem solving, etc.

In section 2 the architecture of MF–KBS is described. Section 3 shows experimental results on a subsystem of an electrical power plant. Section 4 presents related work and its contrast with our approach. Finally, section 5 gives some conclusions and future research directions.

2 MF–KBS

MF–KBS has three main components (see figure 1):

- **Knowledge Core**: generic knowledge about an engineering domain.
- **Reasoning Mechanisms**: different types of inference mechanisms – quantitative, qualitative, probabilistic, causal–, each of which could be static (constraint propagation) or dynamic (simulation).
- **Knowledge Operators**: specific subsystems to perform different tasks over the same knowledge base.

Next we describe each component.

Include figure 1 about here

2.1 Knowledge Core

The knowledge core is designed for engineering domains, such as power plants, chemical plants, etc. It is based on compositional modeling [7], that is, having different models for each basic component and joining them to build the complete model of a device. Each engineering component (pump, valve, tank, pipe, etc) is represented by a frame structure with the following information:

- general information,
- mathematical model(s).

A component frame has the following general information:

- name;

- is–a relation: creating a hierarchy of components;
- parts-of relation: for compositional reasoning;
- related concepts: used for explanations;
- connection constraints: restrictions on the connections of components;
- possible fault modes: ways in which a component can fail;
- fault rate: probabaility of failure of a component;
- connections to other components.

A component has, as well, one or more mathematical models involving state variables (e.g., temperature, pressure, flow, etc). We are considering four types of models: (i) quantitative, (ii) qualitative, (iii) causal, and (iv) probabilistic. Each component can have different types of models, and different models of the same type at different abstraction levels. For instance, a component can be associated with several qualitative and quantitative models at the same time.

Each model is associated with the following information:

- name;
- type: quantitative, qualitative, causal or probabilistic;
- parameters: constants associated to the model;
- variables: input and output variables and their valid ranges;
- constraints: functional applicability information and assumptions (e.g., only for liquids in non–turbulent flow without friction);
- formulation: equations (algebraic, differential, probabilistic) that describe the behavior of the component.

For example, an instance of a *valve* can be represented as follows:

---

name:	control–valve23
is–a:	valve
parts–of:	oil–cooling–system
related concepts:	[temperature, pressure, flow]
active mathematical model:	model1
connection constraints:	[tube]
possible fault models:	[closed, open, stuck]
fault rate:	0.0001036
components connected to:	[I:tube-water-flow32, O: cooling–system1]
model:	
name:	model1
type:	quantitative
parameters:	$\alpha = 11.2352, \beta = 328, \gamma = 0.2608$
variables:	I: W_I [0..100], T_I [20..350] O: W_O [0..100]
constraints:	liquids; non–turbulent
formulation:	$W_O = \alpha(\frac{\beta - T_I}{T_I} + \gamma)W_I$

MF–KBS has a library of components represented by graphical icons, so a device is modeled by selecting and joining components with a graphical editor. The components are organized in a class hierarchy (see figure 2). Once a new system has been configured, a simple validation process is applied using the allowable connections, and all the available consistent models are built[1]. In case the models are used for propagation of values, the composed models are transformed into a constraint or probabilistic net (more details are given below).

[1]So far, we only have one qualitative and one quantitative model for each component, so we do not have a combinatorial explosion of possible models.

The user is questioned about each applicable constraint and assumption for the device, and the system selects the models for each component that satisfy these requirements. If there are still several models that can be used, the user selects the appropiate model.

Include figure 2 about here

2.2 Reasoning Mechanisms

MF–KBS contemplates the use of two classes of reasoning mechanisms, each with different inference techniques according to the type of models:

1. Constraint propagation (static): quantitative, qualitative, probabilistic, or causal.

2. Simulation (dynamic): quantitative, qualitative, probabilistic or causal.

So far we have implemented the constraint propagation algorithms for quantitative, qualitative, and probabilistic models; and we are currently developing the simulation algorithms for quantitative and qualitative (using *QSIM* [10]) models.

For example, the following set of arithmetic constraints could be part of a model of a component (or several components):

$$i = ((a * b) + 1)(c + d)$$

$$h = (a * b) - (c + d)$$

which produces the constraint net shown in figure 3, where $e = a*b$, $f = c+d$, amd $g = e + 1$. With input values: $a = 1$, $b = 2$, $c = 2$, and $d = 3$, we can propagate through the constraints to obtain: $e = 2$, $g = 3$, $f = 5$, $h = -3$, and $i = 15$. Similarly, we can use the same constraint net but now with qualitative values[2]. If the initial values are: $a = std$, $b = dec$, $c = std$, $d = inc$ ($1 = std$), we obtain: $e = dec$, $g = dec$, $f = inc$, $h = dec$, and $i =?$.

Include figure 3 about here

For probabilistic reasoning, we build a model as a Bayesian network [12], and use probabilistic propagation techniques to obtain the posterior probability of the unknown variables in the model. Figure 4 shows an example of a Bayesian network, where each node represents a variable and the links represent probabilistic dependencies. Given the value of some variables (e.g., A, B, F, G), through probability propagation in the network [12] we obtain the posterior probability of the other variables (e.g., C, D, E). This technique is illustrated in more detail in the reliability analysis operator that is described in section 3.

Include figure 4 bout here

2.3 Knowledge Operators

At the top of the architecture is a set of knowledge operators. They are responsible for performing particular tasks using: (i) the knowledge core and (ii) the available reasoning mechanisms. They are "device independent", in the sense that they are able to perform their tasks as long as the device can be described with elements from the library of components. The definition of new knowledge operators increases the functionality of MF–KBS.

So far, we have implemented prototypes of knowledge operators for diagnosis, problem solving, consultation, reliability analysis, and tutoring, but

[2] Where: *std* = steady, *inc* = increassing, *dec* decreassing, and ? = unknown.

others, such as design, validation, and maintenance, could be incorporated in the future.

Each knowledge operator can have its own knowledge and reasoning tools that specify how to use and what to "see" in the generic knowledge base to perform its tasks. For example, a problem solving knowledge operator consults only knowledge related with mathematical models and constraints associated with the components and uses the available reasoning mechanisms of MF–KBS, whether, a diagnostic system looks as well, for instance, at fault modes of the components and uses an ATMS [5].

There can be different knowledge operator for a single task. We can have diagnosis using constraint propagation of qualitative values and ATMS or diagnosis using simulations and rule–based systems. The user is responsible for selecting the particular knowledge operator to use.

A knowledge operator can use another knowledge operator for its task. For example, the diagnosis knowledge operator uses the problem solver and both are used by the tutor.

2.4 Building specific applications

Specific applications are build by joining engineering components, producing instantiations of the integrated model of the system at different abstraction levels. Each application has access to the different reasoning mechanisms described in section 2.2.

Once an application has been built, we can use the generic tools to reason at different abstraction levels. The selection of an abstraction level depends on: (i) the particular task to perform, (ii) the quality of the available information, (iii) the problem solving constraints (e.g., time). Some tasks (e.g., reliability analysis) require certain type of reasoning mechanism and abstraction level, but for others (e.g., diagnosis) there could be several possibilities. In this case, the user selects the abstraction level. We are working on making this selection automatic (decided with simple if-then rules).

3 Experiments and results

MK–KBS uses the hierarchy of components illustrated in figure 2. Each component is associated with a qualitative and a quantitative model. No time considerations are being considered in this implementation, but we are planning to incorporate them in the future.

With this hierarchy, a user can select instances of each component and join them to reason about a particular device. Figure 5 shows an example of an oil lubrication system of an electric power plant that has been implemented with MF–KBS. It is important to mention that during the development of this application, several errors were detected and corrected in the original model extracted from a simulator, that is being used for training power plant operators [3].

Include figure 5 about here

When joining components, connectivity constraints are considered (e.g., a pump is not allowed to connect to another pump with inverted flux). At the reasoning mechanism level, we have implemented reasoning mechanisms to perform propagation of qualitative and quantitative values, as well as probability propagation. Once a set of components is joined a global mathematical model is constructed. This model is transformed into a constraint net where new values can be obtained from a set of given values.

Several prototypes of knowledge operators have been implemented and others can be implemented in the future. One of the key ideas of MF–KBS is that they all use the same information to perform different tasks. An schematic diagram of the current prototype is shown in figure 6. The knowledge core has the information of the components used in the oil lubrication system taken from the hierarchy of components shown in figure 2. The reasoning mechanisms implemented for this system are based on constraint propagation (static) and are: quantitative, qualitative and probabilistic. All these mechanism use the model of the device and associated information in the knowledge core. At the outer layer, a set of knowledge operators have been implemented. They use the previous reasoning mechanisms and information in the knowledge core. Some operators can use other knowledge operators, as well as their own reasoning mechanisms.

Include figure 6 about here

Next the prototype operators for problem solving, automatic example generation, diagnosis, reliability analysis and tutoring are briefly described.

3.1 Problem solver

MF–KBS uses a constraint net and the available reasoning mechanisms to solve problems pose by the user [14]. It accepts variable values from the user, checks for consistency within their allowable ranges and performs constraint propagation. In case the values fed by the user are not sufficient to solve the problem, MF–KBS shows a list of unknown variables and requests the user for more information. In case the information given by the user is contradictory, it warns the user, shows the contradiction, and allows him/her to change or remove some of the values. For example, the user may want to know the output flow of the control valve, given the flow of cooling water and its temperature. This knowledge operator can be used for tutoring, also.

3.2 Automatic example generator

Given the constraint net, MF–KBS randomly chooses a variable, identifies the required variables to obtain a value for the unknown variable, and poses a problem to the user. The selection of the variables whose values are generated, depends on the complexity of the problem, where complexity is defined for MF–KBS as the longest number of intermediate operations between a given value and the unknown. It then selects random values for variables within their valid ranges and asks the user for the value of an unknown variable [14]. With this knowledge operator, MF–KBS can automatically generate problems of increasing complexity. The automatic example generator uses the problem solver to verify the user's answer. For example, the automatic example generator can ask for the angular velocity of the auxiliary pump, given the pressure of the oil in the high pressure tube and the input and output flows in the oil tank. This knowledge operator can be used for tutoring, as well.

3.3 Fault diagnosis

We have implemented a multiple fault diagnostic system [11] that uses a modified ATMS [5], and that is based on GDE [6]. In ATMS, each variable is associated with three data: value, dependency list, and environment (context). The interpretation of these data is that a variable has a particular value obtained in a particular context which depends on the objects mentioned in the dependency list.

If a conflict (difference) is found between a given (measured) value and a predicted (propagated) value, the dependency list of the predicted value represents a set (conflict set) of possible faulty components. The candidate space of possible faults is defined as the power set of the components. Since the size of the candidate space grows exponentially with the number of components, we need an efficient search mechanism to locate the faulty components. ATMS searches this space by considering the minimum conflict set that explains the behavior.

In MF–KBS, when a system is configured, a constraint's network is constructed, from the mathematical model, and used for the propagation (and diagnosis). However, each component keeps the information of the variables used in its model. Although our propagation is through arithmetic constraints, they all belong to a particular component. One of the main differences with an ATMS is that the dependency list uses a set of components. That is, we are using a higher abstraction level. For instance, in the oil lubrication system, the output heat of the oil tank has 18 mathematical operations associated with it. An ATMS would involved 18 elements in its dependency list, MK–KBS uses only 5 (the components associated with the tank) 4 pumps and 1 valve, which simplifies the analysis.

To test the fault diagnosis knowledge operator, values from an initial condition of a simulator [3] were used. Propagating these values were considered as normal behavior. Expected values under faulty conditions were used and all the possible faults considered in the simulator were successfully detected.

It should be pointed out that the fault diagnosis knowledge operator can use the qualitative and quantitative reasoning mechanisms indistinctively, producing a qualitative or quantitative analysis.

3.4 Intelligent tutoring

MF–KBS has a prototype of an intelligent tutoring system[16]. It uses the automatic example generator, the problem solver, and the diagnostic system to teach the functioning of a particular device. In its current implementation, all the tutoring knowledge needs to be manually placed in the knowledge operator once a device has been configured. It uses an overlay–model [16] to characterize what it is known by a student and a simple rule–based system to suggest what information to teach next. It follows a simple tutoring strategy where all the knowledge of a subsystem of the device is first presented to the user. The automatic example generator is then used to produce two examples that are solved first by the problem solver and five more examples that are posed to the student. If the student solves correctly most of the problems (4), the system proceeds to another subsystem. The device is divided into subsystems by the user. We are planning to include tutoring information into the description of the components to automate, as much as possible, the construction of the tutor.

3.5 Reliability analysis

In some engineering domains, such as power plants, it is important to predict the reliability of the system given its components and structure. We have developed a method for obtaining the system reliability with a Bayesian network representation [13]. For this, MF–KBS requires a reliability block diagram of the system (e.g., a power plant), from which a Bayesian network model is constructed automatically. The block diagram is composed by the user from the component library. With the Bayesian network, the reliability of the system is obtain form the individual reliability of each component (fault rate in the component frame) via probability propagation.

Figure 7 shows a simplified block diagram of a turbine's oil lubrication system (see figure 5). From this structure, MF–KBS builds the Bayesian network model shown in figure 8. The top nodes correspond to the components, the intermediate nodes are internal variables that represent parts of the system, and bottom node corresponds to the complete device. Each node has associated a conditional probability matrix that is generated by MF–KBS according to the structure in the block diagram (parallel or serial combination) and used for probability propagation. The probability propagation reasoning mechanism combines the prior probability of the components (obtained from the fault rate) and the conditional probabilities to give the reliability of the system (posterior probability value for node OLS).

Include figure 7 about here

Include figure 8 about here

The current Bayesian network representation is equivalent to the fault tree technique used traditionally for reliability analysis [13], but can be easily extended to incorporate other aspects, such as dependencies between components, which are not possible with fault trees. Another future development is to automate the acquisition of the reliability block diagram from the functional structure.

4 Related work

Our work is closely related to *compositional modeling* [7], where a model is composed to answer a query about a system's behavior. Falkenhainer and Forbus define their model formulation problem as follows: given a library of model fragments and rules constraining their use, a *scenario* description (physical structure, initial conditions, range–limits, etc.), and a query about the behavior of a device, produce the most useful (simplest) and coherent model capable of answering the query. Each model fragment is associated with the structural configurations to which the model applies, assumptions of the model, operating conditions delimiting its scope, and qualitative or quantitative relations.

In MF–KBS the models are associated with specific components. The information associated with each model fragment is similar to ours, however, we do not need to include structural configurations to which the model applies (the user configures the device), and they do not need information about fault modes or probabilistic models due to the reasoning mechanisms that they employ. Our compositional modeling process is much simpler than theirs, since they need to compose a model to answer a particular query, whether in MF–KBS, the user configures a system (joining components) and a model is composed using the models of the components used. Once a model has been composed, they use it only to answer a particular query, whether we can use it for a wider range of tasks (simulation, problem solving, tutoring, reliability analysis, etc).

Probably the work that is closest to ours is the *Device Modeling Environment* (DME) that is being developed at Stanford University [8], [9]. DME integrates model formulation support, qualitative and quantitative simulation, and explanation generation for supporting the design of engineering devices. DME uses a compositional modeling approach for assembling a mathematical model from its library of engineering components. It has a simulation engine that can perform numerical and qualitative simulations, including the capability to change models during simulation (when parameters cross the limit points), so model formulation and simulation are interleaved. The equations from the model fragments and a causal ordering, that is inferred from the models, are combined to produce explanations of the device behavior [8], that support, mainly, the design process.

Although MF–KBS and DME are both focused on engineering domains and based on compositional modeling, there are some important differences. DME is, basically, made for supporting the design of engineering devices, while MF–KBS is multifunctional and does not have any preference for a particular task. MF–KBS supports several tasks, such as diagnosis, tutoring and reliability analysis, using the same knowledge base and reasoning tools, and can be extended to support additional tasks, such as design. In contrast, DME is specifically used for design and it is not clear how to extend it to perform other tasks. Both include different reasoning mechanisms, but MF–KBS has, in principle, a wider variety, including qualitative and quantitative simulations, contraint propagation and probabilistic reasoning. On the other hand, DME's simulation and explanation capabilities are more sophisticated that the ones currently implemented in MF–KBS.

A group at University of Texas at Austin ([1], [4]) is also developing multifunctional knowledge–based systems. They have built large scale knowledge–bases for botany and distributed computer systems. Their representation is based on semantic networks and frames, and it represents the "general" knowledge about certain physical domain. They have developed, among others, tools for assembling a representation from components [4] and extracting different viewpoints from the knowledge base [1]. The main task of their system is to answer questions, describe concepts and generate explanations about a physical domain.

While the work at Austin and MF–KBS have in common their multi-functionality, they differ in the representation and reasoning mechanisms, and also in the type of domains they are designed for. They use a frame type representation with reasoning mechanisms based on logic and graph theory, while we use mathematical models with constraint propagation and simulation. Their representation is appropiate for domains where precise models are difficult to obtain, such as botany; while ours is designed for engineering domains, where mathematical models of the devices can be built.

5 Conclusions and future work

The construction of knowledge–based systems has always been difficult and time–consuming. Once an applications has been built, its knowledge base is very rarely used to build other applications, even when they are from the same domain. In this paper we have described a system, called MF–KBS, capable of generating different applications using the same core of knowledge. MF–KBS has a library of components, each associated with models at different abstraction levels, a set of reasoning mechanisms that act over these models, and a set of knowledge operators that perform different tasks. The user selects instances of components of a particular device, and joins them together through a graphical interface. MF–KBS constructs a global model for the device via composition of simpler models. Once a model has been built, MF–KBS can use its reasoning mechanisms and any of its knowledge operators to construct a particular application. Each knowledge operators has a particular "view" of the knowledge core and uses the existing reasoning mechanisms to perform a particular task. A knowledge operator can use, as well, other knowledge operators and other reasoning mechanisms.

The definition of a new knowledge operator produces a new functionality for all the devices that can be configured by the system. Adding components and/or models to the library of components widens the applicability of MF–KBS, allowing an incremental development.

In addition to refining the definition of some of the knowledge operators and including simulations in the reasoning mechanisms, there are several research issues that are being currently considered. In particular, we would like to provide the knowledge operators with an adequate selection criterion of the reasoning tools to use. In cases where more than one knowledge operator exist for a single task, we would like to provide the user with adequate information about the operator capabilities to make a better decision. We are studying alternative methods to compose models in the presence of multiple models of the same type (e.g., qualitative) at different abstractions levels. Although a component can be associated with different models at different abstractions levels, some models can be obtained from more detailed models through abstraction and simulation. In particular, a quantitative model can be used, in principle, for the construction of a qualitative (through abstractions), causal (additional abstractions) or probabilistic model (using the causal model and simulations).

Acknowledgements
The authors would like to thank Bruce Porter and his research group for helpful discussions during the development of this work. Iliana Parra, Gerardo Torres, Graciela Mojica, Ma. de Jesús Vergara, and Pablo de Buen contributed in the development of the system. This research was partially supported by CONACyT project No. 5357–A.

References

[1] Acker, L. & Porter, B. (1994). Extracting Viewpoints from Knowl-

edge Bases. In *Twelfth National Conference on Artificial Intelligence, (AAAI–94)* (pp. 547–552). Menlo Park: AAAI Press.

[2] Cantú–Ortíz, F.J. (1991). *Operational Expert System Applications in Mexico*. Oxford: Pergamon Press.

[3] C.F.E. (1992). *Simulador Parcial Rodado de Turbina; Condiciones Iniciales Preprogramadas*, Technical Report: Comisión Federal de Electricidad, Centro de Adiestramiento de Operadores, M'exico.

[4] Clark, P. & Porter, B. (1996). *Building Domain Representations from Components*, Tecnical Report AI96-241, Dept. of Computer Science, University of Texas at Austin.

[5] De Kleer, F. (1986). An Assumption based Truth Maintenance System. *Artificial Intelligence*, **28**:127-162, 1986.

[6] De Kleer, J. & Williams, B. (1987). Diagnosing Multiple Faults. *Artificial Intelligence*, **32**:97-130.

[7] Falkenhainer, B. & Forbus, K.D. (1991). Compositional modeling: finding the right model for the job. *Artificial Intelligence*, **51**: 95–143.

[8] Gautier, P. O. & Gruber, T. R. (1993). Generating Explanations of Device Behavior Using Compositional Modeling and Causal Ordering. In *Eleventh National Conference on Artificial Intelligence (AAAI–93)* (pp. 000–000). Menlo Park: AAAI Press.

[9] Iwasaki, Y. & Low, C.M. (1993). Model Generation and Simulation of Device Behavior with Continuous and Discrete Changes. *Intelligent Systems Engineering*, **1**(2):000–000.

[10] Kuipers, B. (1994). *Qualitative Reasoning: modeling and simulation with incomplete knowledge*. Cambridge: MIT Press.

[11] Parra, I. & Morales, E. (1995). Diagnóstico de sistemas físicos a través de un mecanismo de propagación de restricciones. In *XII Reunión Nacional de Inteligencia Artificial* (pp. 114–120). México: Noriega.

[12] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA: Morgan-Kaufmann.

[13] Torres, J.G. & Sucar, L.E. (1995). Un Enfoque de Redes Bayesianas en Confiabilidad de Sistemas. In *XII Reunión Nacional de Inteligencia Artificial* (pp. 47–53). México: Limusa.

[14] Vergara, M. de J. (1997). Generacióm Automática de Problemas para Procedimientyos de Ingeniería Civil, *Master Thesis*, ITESM–Campus Morelos, México.

[15] Walker, T.C. & Miller, R.K. (1991). *Expert Systems 1990, An assessment of Technology and Applications*, SEAI Technical Publications.

[16] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems, Computational and Cognitive Approaches to the Communication of Knowledge.* San Mateo, CA: Morgan–Kaufman.

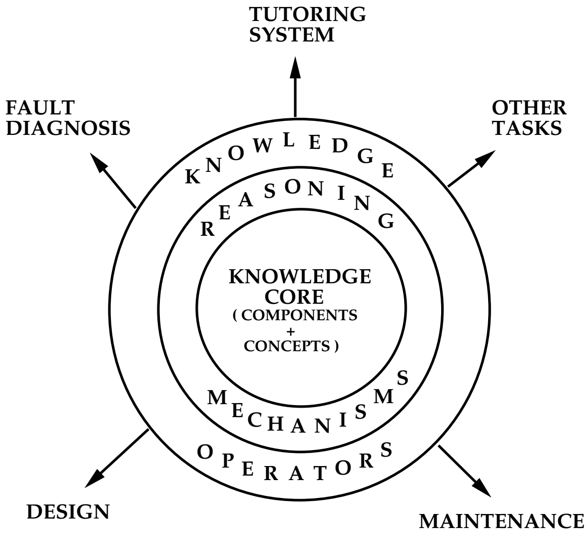

Figure 1: MF–KBS components.

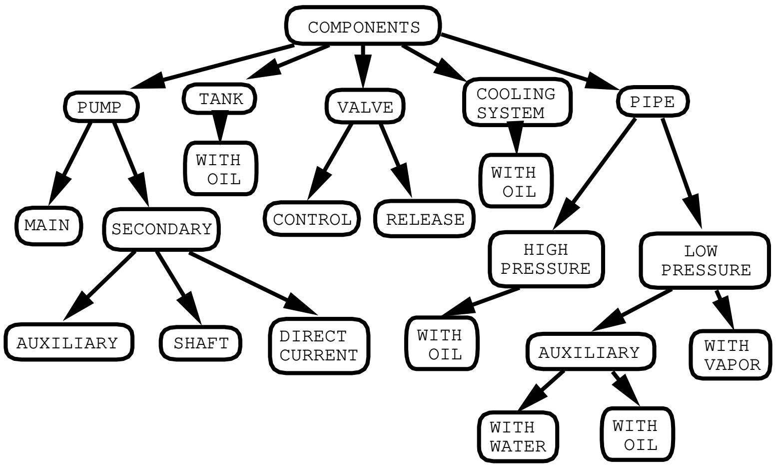

Figure 2: Hierarchy of components for an oil lubrication system of a power plant.

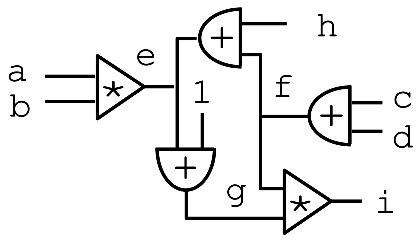

Figure 3: An example constraint net.

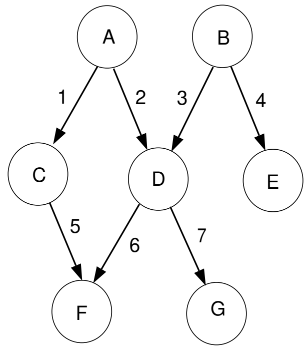

Figure 4: An example of a Bayesian network.

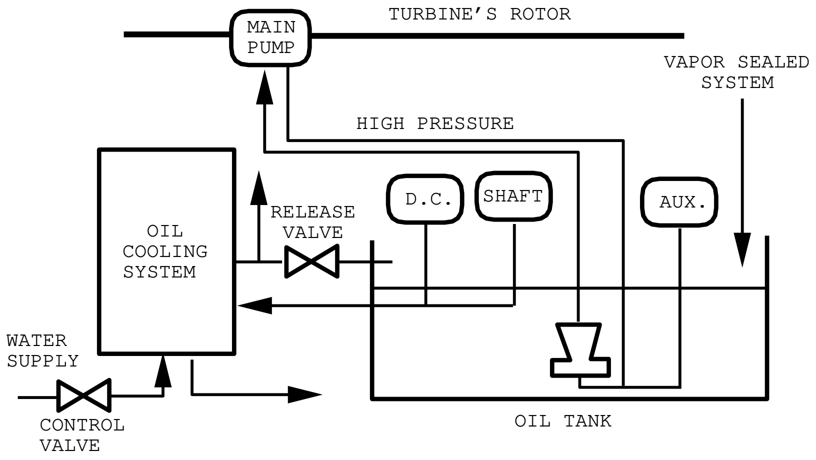

Figure 5: Turbine's oil lubrication system.

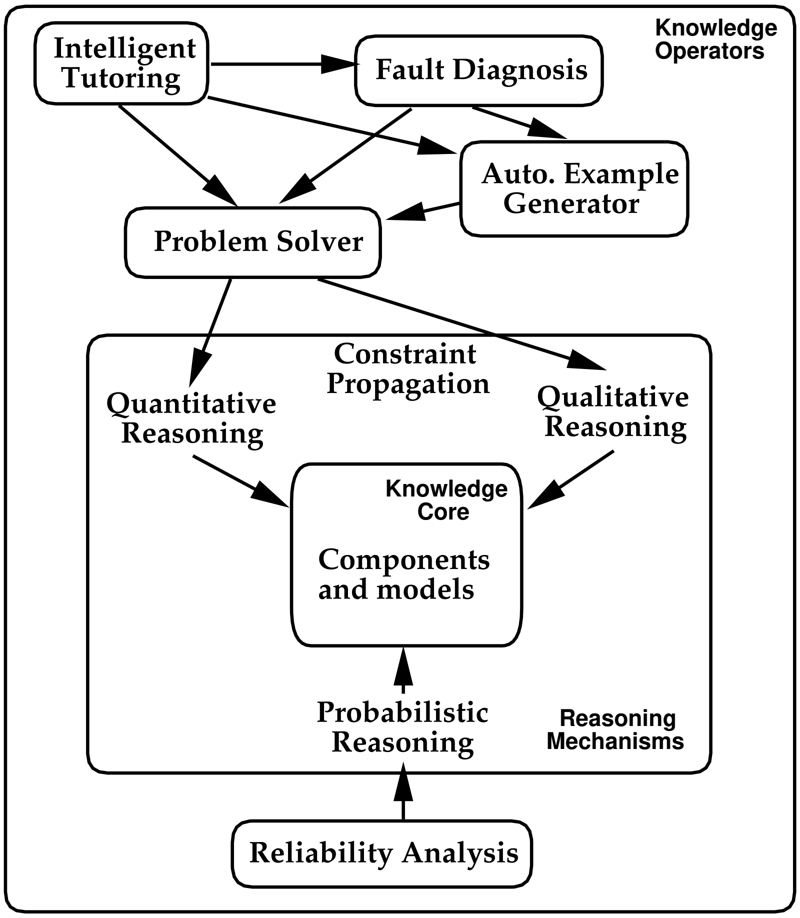

Figure 6: MF–KBS: current implementation.

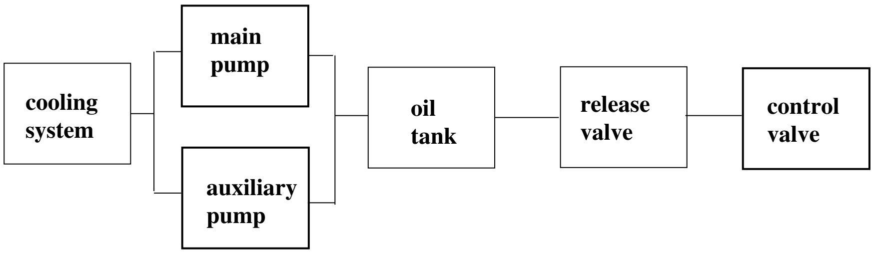

Figure 7: Simplified reliability block diagram for the oil lubrication system.

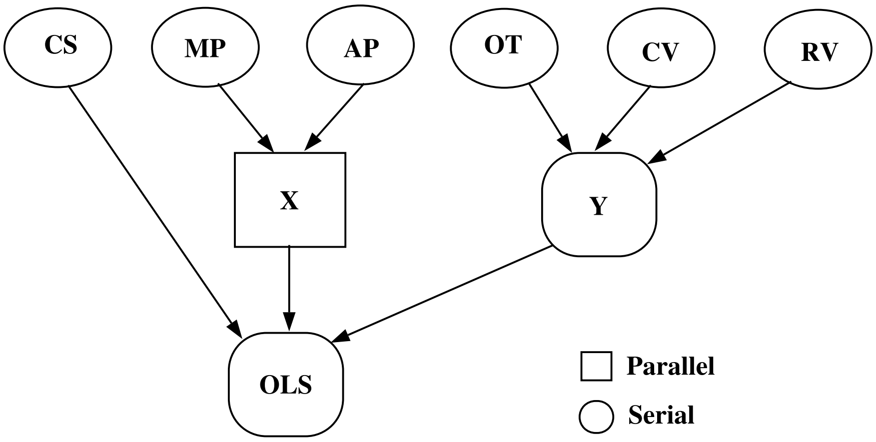

Figure 8: Bayesian network for reliability analysis obtained from the block diagram. (Notation: CS: cooling system; MP: main pump; AP: auxiliary pump; OT: oil tank; RV: release valve; CV: control valve; X,Y: auxiliary variables; OLS: oil lubrication system.)